Lecture Notes in Computer Science 3572

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Clelia De Felice Antonio Restivo (Eds.)

Developments in Language Theory

9th International Conference, DLT 2005
Palermo, Italy, July 4-8, 2005
Proceedings

Springer

Volume Editors

Clelia De Felice
Università di Salerno
Dipartimento di Informatica ed Applicazioni "Renato M. Capocelli"
84081 Baronissi, Italy
E-mail: defelice@dia.unisa.it

Antonio Restivo
Università degli Studi di Palermo
Dipartimento di Matematica ed Applicazioni
Via Archirafi 34, 90123 Palermo, Italy
E-mail: restivo@math.unipa.it

Library of Congress Control Number: 2005928167

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

ISSN 0302-9743
ISBN-10 3-540-26546-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26546-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11505877 06/3142 5 4 3 2 1 0

Preface

DLT 2005 was the 9th Conference on Developments in Language Theory. It was intended to cover all important areas of language theory, such us grammars, acceptors and transducers for strings, trees, graphs, and arrays; efficient text algorithms; algebraic theories for automata and languages; combinatorial and algebraic properties of words and languages; variable-length codes; symbolic dynamics; decision problems; relations to complexity theory and logic; picture description and analysis; polyominoes and bidimensional patterns; cryptography; concurrency; and DNA and quantum computing.

DLT 2005 was held at Mondello (Palermo, Italy) during July 4–8, 2005 and was sponsored by the Department of "Matematica e Applicazioni," University of Palermo, the Department of "Informatica e Applicazioni – R.M. Capocelli," University of Salerno, and the MIUR Project *"Formal Languages and Automata: Methods, Models and Applications."* The conference was also under the auspices of EATCS. We are grateful to these organizations.

Previous DLTs were held in Turku (1993), Magdeburg (1995), Thessaloniki (1997), Aachen (1999), Vienna (2001), Kyoto (2002), Szeged (2003) and Auckland (2004). Since 2001, a DLT conference takes place in every odd year in Europe, and in every even year in another continent.

The Program Committee selected 29 papers from 73 submitted papers. The papers came from the following countries: Austria, Belgium, Canada, Czech Republic, Finland, France, Germany, India, Italy, Portugal, Moldova, Russia, Spain, and the UK. Each submitted paper was evaluated by at least three members of the Program Committee, who were often assisted by their referees. All 29 selected papers are contained in this volume together with 6 invited presentations.

We would like to thank the members of the Program Committee for the evaluation of the submissions and the numerous referees who assisted in this work. The list of the referees is as complete as we could achieve and we apologize for any omissions and errors. We are grateful to the contributors to DLT 2005, in particular to the invited speakers, for the realization of a very successful conference. We also thank the members of the Organizing Committee and Chiara Epifanio and Marinella Sciortino in particular whose work behind the scenes contributed to this volume.

April 2005 Clelia De Felice and Antonio Restivo

Organization

Invited Lecturers

Jean Paul Allouche (France)
Aldo de Luca (Italy)
Dominique Perrin (France)
Jarkko Kari (Finland)
Cenk Sahinalp (Canada)
Howard Straubing (USA)
Mikhail V. Volkov (Russia)

Program Committee

A. Bertoni (Milan)
V. Bruyère (Mons)
C. Calude (Auckland)
C. Choffrut (Paris)
C. De Felice (Salerno, co-chair)
Z. Esik (Szeged)
H.J. Hoogeboom (Leiden)
O. Ibarra (Santa Barbara)
J. Karhumäki (Turku)
W. Kuich (Vienna)
J.E. Pin (Paris)
A. Restivo (Palermo, co-chair)
J. Sakarovitch (Paris)
P.V. Silva (Porto)
W. Thomas (Aachen)
T. Yokomori (Tokyo)

Organizing Committee

A. Restivo (Palermo, co-chair)
C. De Felice (Salerno, co-chair)
F. Mignosi (Palermo)
S. Mantaci (Palermo)
M. Sciortino (Palermo)
C. Epifanio (Palermo)
F. Burderi (Palermo)
G. Castiglione (Palermo)
A. Gabriele (Palermo)
L. Giambruno (Palermo)

Referees

J.-P. Allouche
M. Anselmo
P. Arnoux
E. Asarin
J.-M. Autebert
M.-P. Béal
D. Beauquier
M. ter Beek
B. Berard
A. Berlea
J. Berstel
V. Berthé
A. Bertoni
D. Besozzi
J.-C. Birget
V. Blondel
S.L. Bloom
Y. Boukhad
R. Brijder
V. Bruyère
C.S. Calude
G. Castiglione
D. Caucal
M. Cavaliere
C. Choffrut
T. Colcombet
B. Courcelle
S. Crespi Reghizzi
E. Csuhaj-Varju
E. Czeizler
F. D'Alessandro
Z. Dang
C. De Felice
V. Diekert
J. Engelfriet
M. Faella
S. Ferenczi
P. Frisco
A. Frosini
Z. Fulop
D. Giammarresi
M. Goldwurm
P. Grabner
V. Halava
T. Harju
M. Hirvensalo
J. Holub
J. Honkala
H.J. Hoogeboom
O.H. Ibarra
C. Imreh
N. Jonoska
M. Jurdziński
J. Justin
J. Karhumäki
J. Kari
W. Karianto
D. Kirsten
F. Klaedtke
J. Kleijn
S. Kobayashi
W. Kuich
M. Kunc
S. La Torre
P. Lecomte
E.L. Leiss
M. Lewenstein
V. Liskovets
K. Lodaya
C. Löding
S. Lombardy
V. Lonati
S. Margolis
P. Massazza
J. Mazoyer
C. Mereghetti
J.F. Michon
M. Mishna
D. Niwinski
S. Okawa
A. Okhotin
B. Palano
M. Parente
A. Paun
G. Pighizzini
H. Petersen
J.-E. Pin
L. Polak
B. Ravikumar
E. Remila
A. Restivo
G. Richomme
S. Rinaldi
J. Rutten
N. Sabadini
Y. Sakakibara
J. Sakarovitch
K. Salomaa
K. Sato
P. Schnoebelen
M. Sciortino
P. Seebold
G. Senizergues
O. Serre
J. Shallit
P.V. Silva
S. Szabo
N. Tanida
P. Tesson
D. Thérien
W. Thomas
S. Tini
S. Vagvolgyi
G. Vaszil
S. Varricchio
R. van Vliet
H. Vollmer
C. Wartena
B. Watson
P. Weil
S. Wöhrle
Hsu-Chun Yen
T. Yokomori
S. Yu
R. Zizza

Table of Contents

Restricted Towers of Hanoi and Morphisms

Jean-Paul Allouche[1,⋆] and Amir Sapir[2]

[1] CNRS, LRI, Université Paris Sud, Centre d'Orsay, F-91405 Orsay Cedex, France
allouche@lri.fr
[2] Department of Computer Science, Ben-Gurion University, Beer-Sheva 81405, Israel
amirsa@cs.bgu.ac.il

Abstract. The classical towers of Hanoi have been generalized in several ways. In particular the second named author has studied the 3-peg Hanoi towers with all possible restrictions on the permitted moves between pegs. We prove that all these Hanoi puzzles give rise to infinite morphic sequences of moves, whose appropriate truncations describe the transfer of any given number of disks. Furthermore two of these infinite sequences are actually automatic sequences.

1 Introduction

The towers of Hanoi have been introduced by the famous French number-theoretist Lucas, see [17]. Actually Lucas even invented in 1883 under the pseudonym Claus (anagram of Lucas) a legend about these towers, see e.g., http://www.cs.wm.edu/∼pkstoc/toh.html

We recall briefly that this puzzle consists of three pegs and N distinct disks numbered $1, 2, \ldots, N$. The disks are all placed on peg 1 in increasing order of size (smallest disk on top). A move consists of taking the topmost disk on some peg and placing it on the top of some other peg, with the requirement that no disk should be covered by a larger one. The purpose is to move all the disks from peg 1 to some other peg in a minimal number of moves.

While the original game has been extensively studied, several variations have been proposed: cyclic towers of Hanoi, more than three pegs, colored disks, ... The reader can read in particular the bibliographies of [2, 5, 14], and the 207-item bibliography by Stockmeyer available at http://www.cs.wm.edu/∼pkstoc/biblio.ps

Several papers on Hanoi towers deal with links between the Hanoi puzzle (or related algorithms) and other objects from mathematics or computer science, e.g., finite automata [5], morphisms of free monoids [4], Toeplitz sequences [3, 5], Pascal's triangle [15], Stern's diatomic sequence [16].

In [18] the second named author has studied optimal algorithms for the 3-peg Hanoi puzzle where some moves from a given peg to a given peg are forbidden: there are five non-isomorphic possibilities (the case where the pegs are aligned and only moves between adjacent pegs are permitted goes back to [19]; the case

⋆ Supported by MENESR, ACI NIM 154 Numeration

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 1–10, 2005.
© Springer-Verlag Berlin Heidelberg 2005

where the pegs lie on a circle and only clockwise moves are permitted goes back to [7]). We will prove in this paper that these five possibilities give rise to infinite sequences of moves that are *morphic* (a definition is given in the next section) and such that appropriate prefixes (truncations) of these infinite sequences yield (possibly up to a small number of final moves) a minimal sequence of moves for transferring any given number of disks. This generalizes the case of classical and cyclic Hanoi towers, see [4, 5].

2 Morphic Sequences and Automatic Sequences

We give a short reminder about morphic and automatic sequences (see for example [6] for a more complete account).

Let Σ be an alphabet (finite set) and let Σ^* be the free monoid generated by Σ (i.e., the set of finite words on Σ equipped with the concatenation operation). Let $h : \Sigma^* \to \Sigma^*$ be a morphism (i.e., a homomorphism of monoid). The morphism h is said to be *prolongable* on a if there exists a letter $a \in \Sigma$ and a word $x \in \Sigma^*$ such that $h(a) = ax$, and, for all $k \geq 0$, $h^k(x) \neq \emptyset$. It is then clear that the following limit exists

$$h^\infty(a) := \lim_{k\to\infty} h^k(a) = ah(a)h^2(a)h^3(a)\cdots$$

A sequence $\mathbf{u} = (u_n)_{n\geq 0}$ with values in Σ is said to be an *iterative fixed point* of the morphism h if h is prolongable on u_0. It is then clear that

$$\mathbf{u} = h^\infty(u_0) := \lim_{k\to\infty} h^k(u_0) = u_0h(u_0)h^2(u_0)h^3(u_0)\cdots$$

Of course the sequence $\mathbf{u}$ is then a fixed point of (the extension by continuity of) h (to infinite sequences).

Definition 1. *A sequence* $\mathbf{u} = (u_n)_{n\geq 0}$ *on the alphabet* Σ *is said to be* pure morphic *if it is an iterative fixed point of some prolongable morphism on* Σ.

A sequence $\mathbf{v} = (v_n)_{n\geq 0}$ *on the alphabet* Δ *is said to be* morphic *if it is a coding of a pure morphic sequence on some alphabet* Σ*, i.e., if there exist an alphabet* Σ*, a morphism from* Σ^* *to itself, a map* τ *from* Σ *to* Δ*, and an infinite sequence* $\mathbf{u} = (u_n)_{n\geq 0}$ *with values in* Σ *such that* $\mathbf{u}$ *is an iterative fixed point of* h *and for each* $n \geq 0$, $v_n = \tau(u_n)$*. If, furthermore, the morphism* h *has* constant length ℓ *(i.e., for each letter* $a \in \Sigma$*, the length, i.e., the number of letters, of* $h(a)$ *is equal to* ℓ*) the sequence* $\mathbf{v}$ *is said to be* ℓ-automatic.

The following result shows how to solve the Hanoi puzzle in the classical and cyclic cases by using morphic or automatic sequences. Let a, resp. b, c, be the move that takes the topmost disk from peg 1 and put it on top of peg 2, resp. from peg 2 to peg 3 and from peg 3 to peg 1 (see Figure 1), and let $\overline{a}, \overline{b}, \overline{c}$ be the corresponding reverse moves (e.g., $\overline{a}$ takes the topmost disk from peg 2 and put it on top of peg 1).

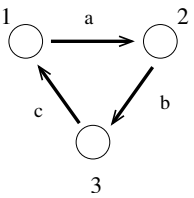


Fig. 1. The direct moves for the towers of Hanoi

Theorem 1 ([4, 5]).
(i) There exists a sequence $\mathbf{u} = (u_n)_{n\geq 0}$ *on the alphabet* $\{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$ *that is* 2*-automatic, and such that its prefixes of length* $2^N - 1$ *describe a minimal set of moves in the* classical *Hanoi puzzle to transfer* N *disks from peg* 1 *to peg* 2 *if* N *is odd and from peg* 1 *to peg* 3 *if* N *is even. The sequence* $\mathbf{u}$ *is the iterative fixed point of the morphism of length* 2 *defined on the alphabet* $\{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$ *by*

$$\begin{array}{llll} a \to a\overline{c}, & b \to c\overline{b}, & c \to b\overline{a} \\ \overline{a} \to ac, & \overline{b} \to cb, & \overline{c} \to ba \end{array}$$

(ii) There exists an infinite sequence on the alphabet $\{a, b, c\}$ *that is the common limit of the finite minimal sequences of moves given by Atkinson for the* cyclic *towers of Hanoi that permit to transfer* N *disks from peg* 1 *to peg* 2 *or from peg* 1 *to peg* 3*. Furthermore this sequence is morphic: it is the image under the map* $\varphi : \{f, g, h, u, v, w\} \to \{a, b, c\}$ *where* $\varphi(f) = \varphi(w) := a$, $\varphi(g) = \varphi(u) := c$, $\varphi(h) = \varphi(v) := b$ *of the iterative fixed point of the morphism* s *defined on* $\{f, g, h, u, v, w\}$ *by*

$$\begin{array}{lll} s(f) = fvf, & s(g) = gwg, & s(h) = huh, \\ s(u) = fg, & s(v) = gh, & s(w) = hf \end{array}$$

3 3-Peg Towers of Hanoi with Forbidden Moves

3.1 Known Results

All the possible restrictions of moves under which the three-peg Hanoi puzzle can be solved are given in [18]. Up to "isomorphism" there are only five cases: these cases are given in Figure 2:

- the *complete* puzzle uses all possible moves in $\{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$;
- the *three-in-a-row* puzzle ("lazy" puzzle) only uses the moves in $\{a, b, \overline{a}, \overline{b}\}$;
- the *cyclic* puzzle only uses the moves in $\{a, b, c\}$;
- the *complete*$--$ puzzle only uses the moves in $\{a, b, c, \overline{a}, \overline{b}\}$;
- the *cyclic*$++$ puzzle only uses the moves in $\{a, b, c, \overline{a}\}$.

In [18] Sapir gives minimal recursive algorithms to solve these five Hanoi problems. Furthermore he shows that the number of moves for N disks has order

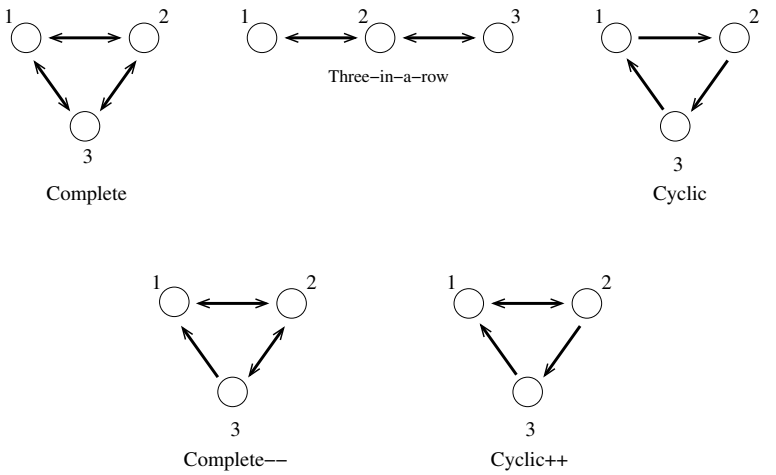


Fig. 2. The five 3-peg Hanoi puzzles

of magnitude λ^N, where λ is respectively equal to 2, 3, $1+\sqrt{3}$, $(1+\sqrt{17})/2$, and the largest root of the polynomial x^3-x^2-4x+2 in each of the five cases above. We will show that these algorithms give more: they permit to construct infinite morphic sequences of moves from which the moves for N disks can be easily deduced for any N.

3.2 Morphic Sequences for the Restricted 3-Peg Hanoi Puzzle

We prove the following theorem.

Theorem 2. *1. The five* 3*-peg Hanoi puzzles with restricted moves give rise to infinite sequences, obtained as limits of the sequence of moves for N disks when N goes to infinity. Let us call these sequences the "complete Hanoi sequence", the "three-in-a-row Hanoi sequence", the "cyclic Hanoi sequence", the "complete−− Hanoi sequence", and the "cyclic++ Hanoi sequence".*

2. The five restricted Hanoi sequences are morphic. Furthermore the complete Hanoi sequence is 2*-automatic and the three-in-a-row Hanoi sequence is* 3*-automatic. The morphisms and codings can be given explicitly:*

– *the* complete Hanoi sequence *is the iterative fixed point of the morphism*

$$a \to a\overline{c},\ \ b \to c\overline{b},\ \ c \to b\overline{a}$$
$$\overline{a} \to ac,\ \ \overline{b} \to cb,\ \ \overline{c} \to ba$$

on the alphabet $\{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$*;*

– *the* three-in-a-row Hanoi sequence *is the iterative fixed point of the morphism*

$$a \to a\ b\ a,\quad \overline{a} \to a\ b\ \overline{a},\quad b \to \overline{b}\ \overline{a}\ b,\quad \overline{b} \to \overline{b}\ \overline{a}\ \overline{b}.$$

– *the* cyclic Hanoi sequence *can be obtained as the coding under the map* $\varphi : \{f, g, h, u, v, w\} \to \{a, b, c\}$ *where* $\varphi(f) = \varphi(w) := a$, $\varphi(g) = \varphi(u) := c$, $\varphi(h) = \varphi(v) := b$ *of the iterative fixed point of the morphism* s *defined on* $\{f, g, h, u, v, w\}$ *by*

$$\begin{array}{lll} s(f) = fvf, & s(g) = gwg, & s(h) = huh, \\ s(u) = fg, & s(v) = gh, & s(w) = hf \end{array};$$

– *the* complete−− Hanoi sequence *can be obtained as the coding under the map* $\varphi : \{x, s, z, d, e, f, a, b, c, \overline{a}, \overline{b}\} \to \{a, b, c, \overline{a}, \overline{b}\}$ *defined by*

$$\begin{array}{lll} \varphi(x) = a, & \varphi(s) = a, & \varphi(z) = \overline{a}, \\ \varphi(d) = b, & \varphi(e) = c, & \varphi(f) = \overline{b} \\ \varphi(a) = a, & \varphi(\overline{a}) = \overline{a}, & \\ \varphi(b) = b, & \varphi(\overline{b}) = \overline{b}, & \\ \varphi(c) = c & & \end{array}$$

of the iterative fixed point of the morphism θ *defined by*

$$\begin{array}{lll} \theta(x) = s\ b\ a\ f, & \theta(s) = s\ b\ a\ e\ b\ s, & \theta(z) = d\ \overline{a}\ e, \\ \theta(d) = z\ b\ s\ b, & \theta(e) = f\ c\ z, & \theta(f) = e\ \overline{b}\ x, \\ \theta(a) = a, \quad \theta(b) = b, & \theta(\overline{a}) = \overline{a}, \quad \theta(\overline{b}) = \overline{b}, & \theta(c) = c. \end{array}$$

– *the* cyclic++ Hanoi sequence *can be obtained as the coding under the map* $\psi : \{x, r, z, t, u, s, a, b, c, \overline{a}\} \to \{a, b, c, \overline{a}\}$ *defined by*

$$\begin{array}{lll} \psi(x) = a, & \psi(r) = a, & \psi(z) = \overline{a}, \\ \psi(t) = b, & \psi(u) = c, & \psi(s) = c \\ \psi(a) = a, & \psi(\overline{a}) = \overline{a}, & \\ \psi(b) = b, & \psi(\overline{b}) = \overline{b}, & \\ \psi(c) = c & & \end{array}$$

of the iterative fixed point of the morphism η *defined by*

$$\begin{array}{lll} \eta(x) = r\ b\ a\ s\ a, & \eta(r) = r\ b\ a\ u\ b\ r, & \eta(z) = t\ \overline{a}\ u, \\ \eta(t) = z\ b\ r\ b, & \eta(u) = s\ a\ c\ z\ c, & \eta(s) = s\ a\ c\ t\ a\ s, \\ \eta(a) = a, \quad \eta(b) = b, & \eta(\overline{a}) = \overline{a}, \quad \eta(\overline{b}) = \overline{b}, & \eta(c) = c. \end{array}$$

Proof (sketch). Our proof will follow the lines of [1, 4, 5]. We define families of words that describe the restricted Hanoi algorithms. These words converge to the infinite Hanoi sequences. Since they are *locally catenative* (roughly speaking there exists a fixed δ such that the nth word depends on the words indexed by $n-1$, $n-2$, ..., $n-\delta$) we can construct morphisms and codings proving that the sequences are morphic. Note that a general result of Shallit (see [20]) proves that locally catenative sequences satisfying mild properties are morphic.

Proofs for complete and cyclic Hanoi sequences can be found in [4, 5] so that it suffices to address the three remaining cases. We are sketching the proofs (inspired by [5], see also [4] and [20]) for the three remaining cases.

– Consider the algorithm given in [18] for the three-in-a-row Hanoi puzzle. Let X_N be the sequence of moves on the alphabet $\{a, b, \overline{a}, \overline{b}\}$ that takes N disks from peg 1 to peg 2, and Y_N the sequence of moves that takes N disks from peg 1 to peg 3, in the minimal algorithm given in [18]. Then, defining the map g by $g(a) := \overline{b}$, $g(b) := \overline{a}$, $g(\overline{a}) := b$, $g(\overline{b}) := a$, a straightforward consequence of the algorithm in [18] is that

$$X_{N+1} = Y_N a g(X_N) \quad \text{and } Y_{N+1} = Y_N a g(Y_N) b Y_N$$

with $X_1 = a$, $Y_1 = ab$. We first note that the two sequences of words $(X_N)_{N\geq 1}$ and $(Y_N)_{N\geq 1}$ converge to a same infinite sequence (since Y_{N+1} begins with Y_N and X_{N+1} begins with Y_N). Let us denote by Y_∞ this common limit.

Let us define $E_N := Y_N a$, $F_N := Y_N \overline{a}$, $G_N := g(E_N)$, and $H_N := g(F_N)$. We see that $E_1 = a\ b\ a$, $F_1 = a\ b\ \overline{a}$, $G_1 = \overline{b}\ \overline{a}\ \overline{b}$, and $H_1 = \overline{b}\ \overline{a}\ b$. Furthermore

$$\begin{aligned} E_{N+1} &= E_N\ H_N\ E_N \\ F_{N+1} &= E_N\ H_N\ F_N \\ G_{N+1} &= G_N\ F_N\ G_N \\ H_{N+1} &= G_N\ F_N\ H_N \end{aligned}$$

and the sequence of words $(E_N)_{N\geq 1}$ clearly converges to Y_∞. Now let σ be the morphism defined on the alphabet $\{a, b, \overline{a}, \overline{b}\}$ by

$$\begin{aligned} \sigma(a) &= a\ b\ a \\ \sigma(\overline{a}) &= a\ b\ \overline{a} \\ \sigma(b) &= \overline{b}\ \overline{a}\ b \\ \sigma(\overline{b}) &= \overline{b}\ \overline{a}\ \overline{b}. \end{aligned}$$

we easily see by induction on N that the following four relations simultaneously hold

$$\sigma(E_N) = E_{N+1}, \quad \sigma(F_N) = F_{N+1}, \quad \sigma(G_N) = G_{N+1}, \quad \sigma(H_N) = H_{N+1}.$$

Hence, for all $N \geq 1$, we have $\sigma^N(a) = E_N$. This implies that $Y_\infty = \sigma^\infty(a)$.

– Case of the complete $--$ Hanoi puzzle. The permitted moves are the elements of $\{a, b, c, \overline{a}, \overline{b}\}$. Let us define the following words on this alphabet:

X_N is the word given by the algorithm in [18] to transfer N disks from peg 1 to peg 2,

Y_N is the word given by the algorithm in [18] to transfer N disks from peg 1 to peg 3,

Z_N is the word given by the algorithm in [18] to transfer N disks from peg 2 to peg 1,

D_N is the word given by the algorithm in [18] to transfer N disks from peg 2 to peg 3,

E_N is the word given by the algorithm in [18] to transfer N disks from peg 3 to peg 1,

F_N is the word given by the algorithm in [18] to transfer N disks from peg 3 to peg 2.

A rephrasing of the algorithm given in [18] yields that:

$$\begin{aligned} X_{N+1} &= Y_N a F_N \\ Y_{N+1} &= Y_N a E_N b Y_N \\ Z_{N+1} &= D_N \overline{a} E_N \\ D_{N+1} &= Z_N b Y_N \\ E_{N+1} &= F_N c Z_N \\ F_{N+1} &= E_N \overline{b} X_N \end{aligned}$$

together with $X_1 = a$, $Y_1 = ab$, $Z_1 = \overline{a}$, $D_1 = b$, $E_1 = c$, and $F_1 = \overline{b}$. It is not hard to see that Y_N ends in b for any $N \geq 1$. We define S_N by $Y_N := S_N b$; hence $S_1 = a$. The relations above can then be written as

$$\begin{aligned} X_{N+1} &= S_N b a F_N \\ S_{N+1} &= S_N b a E_N b S_N \\ Z_{N+1} &= D_N \overline{a} E_N \\ D_{N+1} &= Z_N b S_N b \\ E_{N+1} &= F_N c Z_N \\ F_{N+1} &= E_N \overline{b} X_N \end{aligned}$$

In particular there exist infinite sequences X_∞, Z_∞ and E_∞ on $\{a, b, c, \overline{a}, \overline{b}\}$ such that

$$\begin{aligned} \lim_{N\to\infty} X_N = \lim_{N\to\infty} Y_N = \lim_{N\to\infty} S_N = X_\infty \\ \lim_{N\to\infty} Z_N = \lim_{N\to\infty} D_N = Z_\infty \\ \lim_{N\to\infty} E_N = \lim_{N\to\infty} F_N = E_\infty. \end{aligned}$$

Now, taking the morphism θ and the map φ given in the statement of Theorem 2, we get by an easy simultaneous induction that $\varphi(\theta^N(w)) = W_N$ where w is any of the letters x, s, z, d, e, f and W is the corresponding capital letter in $\{X, S, Z, D, E, F\}$. In particular $\varphi(\theta^\infty(s)) = X_\infty$.

– Case of the cyclic++ Hanoi puzzle. The permitted moves are the elements of $\{a, b, c, \overline{a}\}$. Let us define the following words on this alphabet:

X_N is the word given by the algorithm in [18] to transfer N disks from peg 1 to peg 2,

Y_N is the word given by the algorithm in [18] to transfer N disks from peg 1 to peg 3,

Z_N is the word given by the algorithm in [18] to transfer N disks from peg 2 to peg 1,

T_N is the word given by the algorithm in [18] to transfer N disks from peg 2 to peg 3,

U_N is the word given by the algorithm in [18] to transfer N disks from peg 3 to peg 1,

V_N is the word given by the algorithm in [18] to transfer N disks from peg 3 to peg 2.

A rephrasing of the algorithm given in [18] yields that:

$$\begin{aligned} X_{N+1} &= Y_N a V_N \\ Y_{N+1} &= Y_N a U_N b Y_N \\ Z_{N+1} &= T_N \overline{a} U_N \\ T_{N+1} &= Z_N b Y_N \\ U_{N+1} &= V_N c Z_N \\ V_{N+1} &= V_N c T_N a V_N \end{aligned}$$

together with $X_1 = a$, $Y_1 = ab$, $Z_1 = \overline{a}$, $T_1 = b$, $U_1 = c$, and $V_1 = ca$. It is not hard to see that Y_N ends in b for any $N \geq 1$. We define R_N by $Y_N := R_N b$; hence $R_1 = a$. We also see that V_N ends in a for any $N \geq 1$. We define S_N by $V_N := S_N a$; hence $S_1 = c$. The relations above can then be written as

$$\begin{aligned} X_{N+1} &= R_N b a S_N a \\ R_{N+1} &= R_N b a U_N b R_N \\ Z_{N+1} &= T_N \overline{a} U_N \\ T_{N+1} &= Z_N b R_N b \\ U_{N+1} &= S_N a c Z_N \\ S_{N+1} &= S_N a c T_N a S_N \end{aligned}$$

In particular there exist infinite sequences X_∞, Z_∞ and S_∞ with values in the alphabet $\{a, b, c, \overline{a}\}$ such that

$$\begin{aligned} \lim_{N\to\infty} X_N &= \lim_{N\to\infty} R_N = X_\infty \\ \lim_{N\to\infty} Z_N &= \lim_{N\to\infty} T_N = Z_\infty \\ \lim_{N\to\infty} U_N &= \lim_{N\to\infty} S_N = S_\infty. \end{aligned}$$

Now, taking the morphism η and the map ψ given in the statement of Theorem 2, we get by an easy simultaneous induction that $\psi(\eta^N(w)) = W_N$ where w is any of the letters x, r, z, t, u, s and W is the corresponding capital letter in $\{X, R, Z, T, U, S\}$. In particular $\psi(\eta^\infty(r)) = X_\infty$.

4 Conclusion

Our Theorem 2 above somehow means that all restricted Hanoi puzzles belong to the same class of "regularity" formed by the morphic sequences. For example, computing the ith term of a morphic sequence can be done in time at most polynomial in $\log i$ (see [21] where more is proved). On the other hand automatic sequences form a (strict) subclass of morphic sequences and can be seen as "more regular". The computation of the ith term of an automatic sequence can be done in linear time in $\log i$ (this is a consequence of the possible computation by finite automata with output function, as proved in [10]). Note that a sequence can well be both morphic with a non-uniform morphism and d-automatic for some $d \geq 2$: for example taking the celebrated Thue-Morse sequence and counting the number of 1's between two consecutive zeros, one obtains the sequence

$$2\,1\,0\,2\,0\,1\,2\,1\,0\,1\,2\,0\,1\,2\,0\,\cdots$$

This sequence is both (see [8]) the iterative fixed point of the morphism

$$2 \to 210, \quad 1 \to 20, \quad 0 \to 1$$

and the image under the map $x \to x \bmod 3$ of the iterative fixed point of the 2-morphism

$$2 \to 21, \quad 1 \to 02, \quad 0 \to 04, \quad 4 \to 20.$$

What can be said about the infinite sequences associated with restricted Hanoi puzzles? We have seen that the complete (classical) and three-in-a-row Hanoi sequences are respectively 2-automatic and 3-automatic. We also know that the cyclic Hanoi sequence is not d-automatic for any d (see [1]). Proving that a given sequence is not d-automatic for any d is not an easy task and can be done by using various criteria (frequencies, repetitions, subsequences of certain types, etc.). We hope to finish up the proof that the only restricted Hanoi sequences that are d-automatic for some d are the complete and the three-in-a-row Hanoi sequences, by proving the non-automaticity of the cyclic++ and the complete−− Hanoi sequences. A hint is that the dominant eigenvalues of the transition matrices of the morphisms entering the picture are respectively $\lambda_1 := (1+\sqrt{17})/2$ and λ_2 the maximal root of the polynomial $X^3 - X^2 - 4X + 2$. A conjecture of Hansel (see [11–13] for several results toward a proof) asserts that if the iterative fixed point of a morphism with dominant eigenvalue λ is also d-automatic and not ultimately periodic, then $(\log \lambda)/(\log d)$ must be rational: this is a generalization of a theorem due to Cobham for d- and d'-automatic sequences [9]. Under this conjecture it would suffice to prove that $(\log \lambda_j)/(\log d)$ is irrational for $j = 1, 2$, and to prove that the images of the fixed points of the Hanoi morphisms with dominant eigenvalues λ_j under the corresponding codings are still non-automatic (in other words that these codings are not "trivial"). A small step is given in the proposition below.

Proposition 1. *Let $\lambda_1 := (1+\sqrt{17})/2$ and λ_2 be the maximal root of the polynomial $X^3 - X^2 - 4X + 2$. Then, for any integer $d \geq 1$, the real numbers $(\log \lambda_j)/(\log d)$ with $j = 1, 2$ are irrational.*

Proof. It suffices to show that for any integer $n \geq 1$ the real numbers λ_j^n are not integers.

For λ_1: this is clear since for any $n \geq 1$ there exist integers $s_n > 0$ and $t_n > 0$ such that $2^n \lambda_1^n = (1+\sqrt{17})^n = s_n + t_n\sqrt{17}$.

For λ_2: if α were an integer such that $\lambda_2^n = \alpha$, then the polynomial $X^3 - X^2 - 4X + 2$ would divide $X^n - \alpha$; hence there would exist a polynomial $Q(X)$ with integer coefficients such that

$$(X^3 - X^2 - 4X + 2)Q(X) = X^n - \alpha.$$

Reducing modulo 2 gives

$$X^2(X-1)Q(X) \equiv X^n - \alpha \bmod 2$$

which would imply that α is both congruent to 0 and to 1 modulo 2, hence the desired contradiction.

References

1. J.-P. Allouche, Note on the cyclic towers of Hanoi, in Number theory, combinatorics and applications to computer science (Marseille, 1991), Theoret. Comput. Sci. **123** (1994) 3–7.
2. J.-P. Allouche, D. Astoorian, J. Randall, J. Shallit, Morphisms, squarefree strings, and the Tower of Hanoi puzzle, Amer. Math. Monthly **101** (1994) 651–658.
3. J.-P. Allouche, R. Bacher, Toeplitz sequences, paperfolding, Towers of Hanoi and progression-free sequences of integers, Enseign. Math. **38** (1992) 315–327.
4. J.-P. Allouche, J. Bétréma, J. Shallit, Sur des points fixes de morphismes d'un monoïde libre, RAIRO Inform. Théor. Appl. **23** (1989) 235–249.
5. J.-P. Allouche, F. Dress, Tours de Hanoï et automates, RAIRO Inform. Théor. Appl. **24** (1990) 1–15.
6. J.-P. Allouche, J. Shallit, Automatic sequences. Theory, applications, generalizations, Cambridge University Press, Cambridge, 2003, xvi+571 pp.
7. M. D. Atkinson, The cyclic towers of Hanoi, Inform. Process. Lett. **13** (1981) 118–119.
8. J. Berstel, Sur la construction de mots sans carré, Séminaire de Théorie des Nombres de Bordeaux, 1978–1979, Exposé 18, 18-01–18-15.
9. A. Cobham, On the base-dependence of sets of numbers recognizable by finite automata, Math. Systems Theory **3** (1969) 186–192.
10. A. Cobham, Uniform tag sequences, Math. Systems Theory **6** (1972) 164–192.
11. F. Durand, A generalization of Cobham's theorem, Theory Comput. Syst. **31** (1998) 169-185.
12. F. Durand, A theorem of Cobham for non-primitive substitutions, Acta Arith. **104** (2002) 225–241.
13. F. Durand, Combinatorial and dynamical study of substitutions around the theorem of Cobham, in Dynamics and randomness. Lectures given at the conference, Universidad de Chile, Santiago, Chile, December 11-15, 2000, A. Maass et al. (eds.), Kluwer, Nonlinear Phenom. Complex Systems **7** (2002) 53–94.
14. A. M. Hinz, The Tower of Hanoi, Enseign. Math. **35** (1989) 289–321.
15. A. M. Hinz, Pascal's triangle and the Tower of Hanoi, Amer. Math. Monthly **99** (1992) 538–544.
16. A. M. Hinz, S. Klavžar, U. Milutinović, D. Parisse, C. Petr, Metric properties of the Tower of Hanoi graphs and Stern's diatomic sequence, European J. Combin. **26** (2005) 693–708.
17. É. Lucas, Le calcul et les machines à calculer, Assoc. Française pour l'Avancement des Sciences; Comptes Rendus **13** (1884) 111–141.
18. A. Sapir, the tower of Hanoi with forbidden moves, The Computer Journal **47** (2004) 20–24.
19. R. S. Scorer, P. M. Grundy, C. A. B. Smith, Some binary games, Math. Gazette **280** (1944) 96–103.
20. J. Shallit, A generalization of automatic sequences, Theoret. Comput. Sci. **61** (1988) 1–16.
21. J. Shallit, D. Swart, An efficient algorithm for computing the i'th letter of $\varphi^n(a)$, Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1999, pp. 768–775.

Collapsing Words: A Progress Report*

Dmitry S. Ananichev, Ilja V. Petrov, and Mikhail V. Volkov

Department of Mathematics and Mechanics,
Ural State University, 620083 Ekaterinburg, Russia
Dmitry.Ananichev@usu.ru, ilja_petrov@pisem.net, Mikhail.Volkov@usu.ru

Abstract. A word w over a finite alphabet Σ is *n-collapsing* if for an arbitrary DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, the inequality $|\delta(Q, w)| \le |Q| - n$ holds provided that $|\delta(Q, u)| \le |Q| - n$ for some word $u \in \Sigma^+$ (depending on $\mathscr{A}$). We overview some recent results related to this notion. One of these results implies that the property of being n-collapsing is algorithmically recognizable for any given positive integer n.

Introduction

Let $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ be a deterministic finite automaton (DFA, for short) with the state set Q, the input alphabet Σ, and the transition function $\delta : Q \times \Sigma \to Q$. The action of the letters in Σ on the states in Q defined via δ extends in a natural way to an action of the words in the free Σ-generated monoid Σ^*; the latter action is still denoted by δ. When we deal with a fixed DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, then, for any $w \in \Sigma^*$ and $Q' \subseteq Q$, we set $Q' \,.\, w = \{\delta(q, w) \mid q \in Q'\}$. We call the difference $\mathrm{df}_{\mathscr{A}}(w) = |Q| - |Q \,.\, w|$ the *deficiency of the word w with respect to $\mathscr{A}$*.

Let n be a positive integer. A DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is said to be *n-compressible* if there is a word $w \in \Sigma^*$ with $\mathrm{df}_{\mathscr{A}}(w) \ge n$. The word w is then called *n-compressing with respect to $\mathscr{A}$*. We say that a word $w \in \Sigma^*$ is *n-collapsing* if w is n-compressing with respect to every n-compressible DFA whose input alphabet is Σ. In other terms, a word $w \in \Sigma^*$ is n-collapsing if, for any DFA $\mathscr{A}$, we have $\mathrm{df}_{\mathscr{A}}(w) \ge n$ whenever $\mathscr{A}$ is n-compressible. Thus, such a word is a kind of a 'universal tester' whose action on the state set of an arbitrary DFA with a fixed input alphabet exposes whether or not the automaton is n-compressible.

The concept of an n-collapsing word arose (under a different name) in the beginning of the 1990s with original motivations coming from combinatorics and abstract algebra (cf. [17, 20]). In fact, the notion appears to be fairly natural from the automata theory point of view as it perfectly fits in Moore's classic approach of 'Gedanken-experiments' [14]. Over the last few years, automata/language-theoretic connections of n-collapsing words have been intensely studied, see [3–6, 12, 18], and a few new applications have been found, see [1, 2, 19]. In the present paper we try to summarize these recent developments and also discuss some new results.

* The authors acknowledge support from the Federal Education Agency of Russia, grants 49123 and 04.01.437, the Russian Foundation for Basic Research, grant 05-01-00540, and the Federal Science and Innovation Agency of Russia, grant 2227.2003.01

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 11–21, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The very first problem related to n-collapsing words is of course the question of whether such words exist for every n and over every Σ. We address this question in Sect. 1 where we survey a few constructions for such words and some bounds on their length. In Sect. 2 we present and discuss a new result which allows one to recognize, given a word $w \in \Sigma^*$ and a positive integer n, whether or not w is n-collapsing. Finally, in Sect. 3 we briefly explain how n-collapsing words are used in algebra and exhibit their new application to computational complexity issues in finite semigroup theory.

1 Constructing Collapsing Words

It is easy to see that a word w over a singleton alphabet is n-collapsing if an only if $|w| \geq n$. (Here and below $|w|$ stands for the length of the word w.) Therefore, in the sequel we assume that the size t of our alphabet Σ is at least 2. In this case even the existence of n-collapsing words is not completely obvious, to say nothing about constructing such words in any explicit way. Indeed, in the above definition of an n-collapsing word, the alphabet Σ is fixed but one imposes absolutely no restriction to the number of states of n-compressible automata under consideration. Clearly, there are infinitely many n-compressible automata and they all should be 'served' by the same word: if $w \in \Sigma^*$ is 3-collapsing, say, then w should bring each state of any 3-compressible DFA with 4 states to one particular state and the same w should send the state set of any 3-compressible DFA with 1000000 states to a 999997-element subset, etc.

Sauer and Stone [20] who were arguably the first to introduce n-collapsing words (under the name '*words with property* Δ_n') proved their existence for each n via the following inductive construction. To start with, observe if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is 1-compressible, then there is a letter $a \in \Sigma$ with $\mathrm{df}_{\mathscr{A}}(a) \geq 1$. (Otherwise each transformation $\delta(\_, a)$ would be a permutation and the DFA $\mathscr{A}$ could not be 1-compressible.) Since the deficiency of any word is no less than the maximum deficiency of its letters, we conclude that any word involving all letters in $\Sigma = \{a_1, \ldots, a_t\}$ is 1-collapsing. Having used this observation as the induction basis, Sauer and Stone let $w_1 = a_1 \cdots a_t$ and then proceeded by defining

$$w_{n+1} = w_n \prod_{0 \leq |v| \leq 3 \cdot 2^{n-2}+1} (v w_n). \tag{1}$$

Thus, the right-hand side of (1) is an alternating product of all words from Σ^* of length at most $3 \cdot 2^{n-2}+1$ (in some fixed order) and the corresponding number of copies of the word w_n. Then, for each n, the word w_n is n-collapsing [20, Theorem 3.3]. Of course, the length of these words grows very fast with n (it is a doubly exponential function of n).

It turns out that the same idea can yield a series of much shorter n-collapsing words. Namely, in [12, Theorem 3.5] it is shown that one can restrict the above alternating product to words of length at most $n+1$. More precisely, the result says that if $u_1 = w_1 = a_1 \cdots a_t$ and

$$u_{n+1} = u_n \prod_{0 \le |v| \le n+1} (vu_n) \tag{2}$$

then, for each n, the word u_n is n-collapsing ('*witnesses for deficiency* n' in the terminology of [12]). The proof relies on examining a certain configuration in combinatorics of finite sets and follows Pin's approach in [15]. It can be easily calculated that $|u_n| = O(t^{\frac{n^2-n}{2}})$; for $t \ge 5$, the word u_n is the shortest n-collapsing word known so far.

Another existence proof suggested in [12] exploits tight relations between n-collapsing words and the famous *Černý conjecture* in automata theory. Recall that a DFA $\mathscr{A}$ is called *synchronizable* if there exists a *reset* word whose action 'resets' $\mathscr{A}$, i.e. brings all its states to a particular one. The Černý conjecture [9] claims that each synchronizable automaton with m states possesses a reset word of length at most $(m-1)^2$. The conjecture is open in general; the best estimation of the size of the shortest reset word known so far is due to Pin [16]. It is shown in [12, Section 2] how Pin's estimation implies that any word having among its factors all words over Σ of length $\frac{1}{6}n(n+1)(n+2)-1$ is n-collapsing. The minimum length of such an n-collapsing word is equal to $t^{\frac{n(n+1)(n+2)}{6}-1} + \frac{1}{6}n(n+1)(n+2) - 2$. It is worth mentioning that this alternative construction also depends on a rather involved result from combinatorics of finite sets – a theorem conjectured in [16] and proved by Frankl [10].

We denote by $c(n,t)$ the minimum length of n-collapsing words over t letters. The construction (2) provides an upper bound for $c(n,t)$. As for a lower bound, for $n > 2$ the only known bound follows from an observation made in [12]. We call a word *n-full* if it has all words of length n over Σ among its factors.

Proposition 1 ([12, Theorem 4.2]). *Any n-collapsing word is n-full.*

The shortest n-full word over Σ has the length $t^n + n - 1$ whence $c(n,t) \ge t^n + n - 1$ for all n and t. For $n = 2$, a better lowed bound has been recently found in [18, Theorem 2]: $c(2,t) \ge 2t^2$.

At present, only two exact values of the function $c(n,t)$ are known: $c(2,2) = 8$ [20] and $c(2,3) = 21$ [6, Proposition 3.3]. The words

$$W_8 = aba^2b^2ab \quad \text{and} \quad W_{21} = aba^2c^2bab^2acbabcacbcb \tag{3}$$

are two concrete examples of 2-collapsing words of minimum length over 2 and respectively 3 letters. Observe that these two words can be used to improve the upper bounds for $c(n,2)$ and $c(n,3)$: one gets shorter n-collapsing words over 2 and 3 letters by starting the recursion (2) with $n = 2$ and with the word W_8 or respectively W_{21} in the role of u_2. This gives, for instance, the estimations $c(3,2) \le 162$ and $c(3,3) \le 963$. It is also known that $c(2,4) \le 58$ – this follows from an example of 2-collapsing word of length 58 that has been constructed by Martjugin (unpublished). Again, using Martjugin's word in the role of u_2 in (2) one obtains a series of shorter n-collapsing words over 4 letters.

As for lower bounds, we know that $c(3,2) \ge 33$; this follows from [6, Proposition 3.4]. The reader sees that the gaps between upper and lower estimates for

$c(n,t)$ remain quite large not only in the general case but also for small concrete values of n and t. Thus, even in a short distance ahead there is plenty that needs to be done.

2 Recognizing Collapsing Words

Given a word $w \in \Sigma^*$ and a positive integer n, how to decide whether or not w is n-collapsing? The answer is easy to find for $n = 1$: a word is 1-collapsing if and only if it involves all letters in Σ. In [3], the question has been answered for the first non-trivial case $n = 2$. The solution proposed in [3] is based on a reduction of the initial question to a problem concerning finitely generated subgroups of free groups which can be efficiently solved by certain classic methods of combinatorial group theory. A more geometric version of this idea has been developed in [4]. It results in an algorithm that, given a 2-full word $w \in \Sigma^*$, produces a finite bunch of inverse automata such that w is **not** 2-collapsing if and only if at least one of these inverse automata can be completed to a 2-compressible DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with $|Q| < |w|$ and $\mathrm{df}_{\mathscr{A}}(w) = 1$. (If w is not 2-full, it cannot be 2-collapsing by Proposition 1.) The algorithm involves an exhaustive search through all subsets of certain sets of factors of the word w, and therefore, fails to be polynomial for $t = |\Sigma| > 2$. Still the algorithm can be implemented efficiently enough in order to check 2-collapsability of considerably long words, see [6] for a brief survey of our computer experiments in the area. For instance, the word W_{21} in (3) is the first (in the lexicographical order) in the list of all shortest 2-collapsing words over $\{a, b, c\}$ computed by a program implementing the algorithm from [4] (up to renaming of the letters there are 80 such words).

So far we have not succeeded in extending the ideas from [3, 4] to n-collapsing words with $n > 2$. Therefore, we have focused our efforts on a more modest goal of showing that the language $\mathcal{C}_n$ of all n-collapsing words over Σ is decidable in principle, i.e. is a recursive subset of Σ^*. For this, it suffices to find, for each positive integer n, a computable function $f_n : \mathbb{N} \to \mathbb{N}$ such that a word $w \in \Sigma^*$ is n-collapsing provided $\mathrm{df}_{\mathscr{A}}(w) \geq n$ for every n-compressible DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with $|Q| < f_n(|w|)$. Indeed, if such a function exists, then, given a word w, we can calculate that value $m = f_n(|w|)$ and then check the above condition through all automata with at most $m - 1$ states. Since there are only finitely many such automata with a fixed input alphabet, the procedure will eventually stop. If in the course of the procedure we encounter an n-compressible DFA $\mathscr{A}$ with $\mathrm{df}_{\mathscr{A}}(w) < n$, then the word w is not n-collapsing by the definition. If no such automaton is found, then w is n-collapsing by the choice of the function f_n.

From the results of [4] it follows that, for $n = 2$, a function f_2 with the desired property does exist; in fact, one may set $f_2(\ell) = \max\{4, \ell\}$. Recently, the second-named author has managed to find a suitable function for an arbitrary n. The result can be stated as follows.

Theorem 1. *For every word $w \in \Sigma^*$ which is not n-collapsing, there exists an n-compressible DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with $|Q| \leq 3|w|(n-1) + n + 1$ such that $\mathrm{df}_{\mathscr{A}}(w) < n$.*

As discussed above, this implies that the language $\mathcal{C}_n$ is recursive. In fact, we can say more because Theorem 1 shows that the role of the function f_n can be played by the linear (in $|w|$) function $3|w|(n-1)+n+2$. This immediately leads to a non-deterministic linear space and polynomial time algorithm recognizing the complement of the language $\mathcal{C}_n$: the algorithm simply makes a guess consisting of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with $|Q| \leq 3|w|(n-1)+n+1$ and then verifies that $\mathscr{A}$ is n-compressible (this can be easily done in low polynomial time) and that w is not n-compressing with respect to $\mathscr{A}$. By classical results of formal language theory (cf. [13, Sections 2.4 and 2.5]), this implies that the language $\mathcal{C}_n$ is context-sensitive.

The proof of Theorem 1 is rather technical and cannot be reproduced here in full. Instead we outline its main underlying ideas.

Thus, fix a word $w \in \Sigma^*$ which is not n-collapsing. We may assume that w is n-full – otherwise w is not n-compressing with respect to a synchronizing DFA with $n+1$ states (see the proof of [12, Theorem 4.2]). Now fix an n-compressible DFA $\mathscr{B} = \langle B, \Sigma, \beta\rangle$ such that $\mathrm{df}_{\mathscr{B}}(w) < n$. Without any loss we may assume that $\mathrm{df}_{\mathscr{B}}(w) = n-1$. Indeed, if $\mathrm{df}_{\mathscr{B}}(w) = k < n-1$ we can add $n-k$ new states $q_1, \ldots, q_{n-k}$ to B and extend the transition function β to these new states by letting $\beta(q_i, a) = q_1$ for all $i = 1, \ldots, n-k$ and all $a \in \Sigma$. Clearly, we obtain an n-compressible DFA and the deficiency of the word w with respect to it is precisely $n-1$.

Now assume that some of the states of the DFA $\mathscr{B}$ are covered by tokens and the action of any letter $a \in \Sigma$ redistributes the tokens according the following rule: a state $q \in B$ will be covered by a token after the action of a if and only if there exists a state $q' \in B$ such that $\beta(q', a) = q$ and q' was covered by a token before the action. In more 'visual' terms, the rule amounts to saying that tokens slide along the arrows labelled a and, whenever several tokens arrive at the same state, all but one of them are removed. The following picture illustrates the rule: its right part shows how tokens distribute over the state set of a DFA after completing the action of the letter a on the distribution shown on the left.

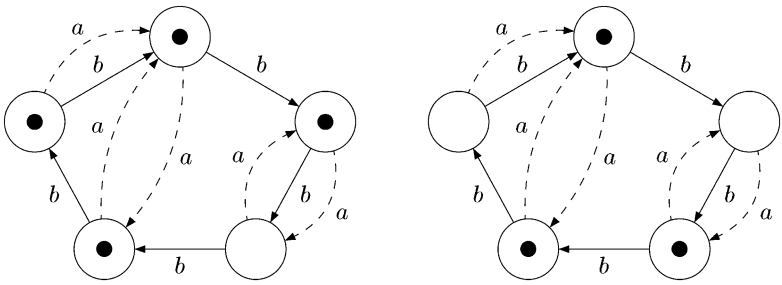


Fig. 1. Redistributing tokens under the action of a letter

Now suppose that we have covered all states in B by tokens and let a word $v \in \Sigma^*$ (that is, the sequence of its letters) act according the above rule. It is

easy to realize that, after completing this action, tokens will cover precisely the set $B \,.\, v$.

Let $\ell = |w|$ and, for $i = 1, \ldots, \ell$, let $w[i] \in \Sigma$ be the letter occupying the i-th position from the left in the word w. Let $w_i = w[1] \cdots w[i]$ be the prefix of length i of w. We cover all states in B by tokens and let the letters $w[1], \ldots, w[\ell]$ act in succession. On the i-th step of this procedure we mark all elements of the following sets of states:

$$\begin{aligned} \mathrm{PES}(i) &= (B \setminus B \,.\, w_{i-1}) \,.\, w[i]; \\ \mathrm{CES}(i) &= B \setminus B \,.\, w_i; \\ \mathrm{NES}(i) &= (B \setminus B \,.\, w_i) \,.\, w[i]. \end{aligned}$$

The meaning of these three sets can be easily explained in terms of the distribution of tokens before and after the action of the letter $w[i]$. It is convenient to call a state *empty* if it is not currently covered by a token. Then $\mathrm{PES}(i)$ is the set of all 'post-empty' states to which the letter $w[i]$ brings states that had been empty before the action of $w[i]$. The set $\mathrm{CES}(i)$ consists of current empty states and $\mathrm{NES}(i)$ contains all 'next-to-empty' states that can be achieved from $\mathrm{CES}(i)$ by an extra action of the letter $w[i]$.

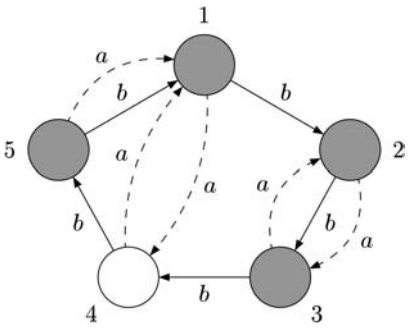


Fig. 2. Marking induced by the transition shown in Fig. 1

For example, assume that the transition shown in Fig. 1 represents the i-th step of the above procedure (so that $w[i] = a$). Then all but one states get marks as shown on Fig. 2. Indeed, $\mathrm{PES}(i) = \{2\}$ because 3 was the only empty state before the action of a and $\beta(3, a) = 2$. Further, $\mathrm{CES}(i) = \{2, 5\}$ and $\mathrm{NES}(i) = \{1, 3\}$.

Let $C = \bigcup_{1 \le i \le \ell} (\mathrm{PES}(i) \cup \mathrm{CES}(i) \cup \mathrm{NES}(i))$ be the set of all states that get marks during the described process. This set forms a core of the 'small' n-compressible DFA $\mathscr{A}$ whose existence is claimed in Theorem 1.

Proposition 2. $|C| \le 3\ell(n-1)$.

Proof. Since $\mathrm{df}_{\mathscr{B}}(w) = n - 1$, at most $n - 1$ states of $\mathscr{B}$ can be empty after the action of each of the letters $w[1], \ldots, w[\ell]$. This implies that each of the sets

PES(i), CES(i), NES(i) contains at most $n-1$ states whence their union C has at most $3\ell(n-1)$ states. □

Now we consider the incomplete automaton $\mathscr{C}$ whose state set is C and whose (partial) transition function $\gamma : C \times \Sigma \to C$ is the restriction of the function β in the following sense: for each $q \in C$ and each $a \in \Sigma$

$$\gamma(q,a) = \begin{cases} \beta(q,a) & \text{if } \beta(q,a) \in C, \\ \text{undefined} & \text{if } \beta(q,a) \notin C. \end{cases}$$

Our next aim is to complete the automaton $\mathscr{C}$ to a DFA by appending new arrows to it.

We call any triple $(q,a,q') \in B \times \Sigma \times B$ such that $\beta(q,a) = q'$ *a transition labelled* a. Let $\overline{C} = B \setminus C$. A transition (q,a,q') is said to be *ingoing* (respectively *outgoing*) if $q \in \overline{C}$ and $q' \in C$ (respectively if $q \in C$ and $q' \in \overline{C}$). We need the following result.

Proposition 3. *For each $a \in \Sigma$, the number of ingoing transitions labelled a does not exceed the number of outgoing transitions labelled a.*

Proof. Any letter $a \in \Sigma$ occurs in the word w (because w is n-full); let i, $1 \le i \le \ell$, be such that $w[i] = a$. By the definition of the set CES($i-1$), all states that become empty after the action of the word w_{i-1} belong to the set C whence, before the action of $w[i]$, all states in the set $\overline{C}$ are covered by tokens. Therefore the number of tokens brought from $\overline{C}$ to C by the action of $w[i] = a$ is equal to the number of ingoing transitions labelled a. Now, arguing by contradiction, suppose that the number of ingoing transitions labelled a exceeds the number of outgoing transitions labelled a. Then the number of tokens arriving at $\overline{C}$ under the action of $w[i]$ is strictly less than the number of tokens leaving $\overline{C}$. This means that after the action of w_i some state $q \in \overline{C}$ becomes empty, that is, $q \in B \setminus B \,.\, w_i = \mathrm{CES}(i) \subseteq C$. We have found a state that belongs to both C and $\overline{C}$, a contradiction. □

By Proposition 3, for each letter $a \in \Sigma$, there exists a one-to-one mapping φ_a from the set of all of ingoing transitions labelled a to the set of all outgoing transitions labelled a. We use these mappings to complete the automaton $\mathscr{C}$ in the following way. If for some $q \in C$, the state $\gamma(q,a)$ is not defined, then $q' = \beta(q,a)$ belongs to $\overline{C}$ so that (q,a,q') is an outgoing transition. If this transition does not lie in the range of φ_a, we define $\gamma'(q,a) = q$; in other words, we append to $\mathscr{C}$ a new loop labelled a at the state q. If $(q,a,q') = \varphi_a((r,a,r'))$ for some (uniquely determined) ingoing transition (r,a,r'), then we define $\gamma'(q,a) = r'$; in other words, we append to $\mathscr{C}$ a new arrow from q to r' labelled a. Now we define a complete transition function $\delta : C \times \Sigma \to C$ by letting

$$\delta(q,a) = \begin{cases} \gamma(q,a) & \text{if } \gamma(q,a) \text{ is defined}, \\ \gamma'(q,a) & \text{if } \gamma(q,a) \text{ is undefined}. \end{cases}$$

We define the DFA $\langle C, \Sigma, \delta \rangle$ by $\mathscr{D}$. The main property of the DFA is contained in the next proposition whose (relatively long) proof is omitted.

Proposition 4. $\mathrm{df}_{\mathscr{D}}(w) = n - 1$.

However, the automaton $\mathscr{D}$ is not yet the DFA from the formulation of Theorem 1 because in general $\mathscr{D}$ may fail to be n-compressible. On the other hand, the DFA $\mathscr{B}$ is n-compressible, that is, there exists a word $u \in \Sigma^*$ with $\mathrm{df}_{\mathscr{B}}(u) \geq n$. Then, of course, $\mathrm{df}_{\mathscr{B}}(wu) \geq n$ as well whence in the set $B \,.\, w$ there are two different states p and r such that $\beta(p, u) = \beta(r, u)$. We fix such a pair $(p, r) \in B \,.\, w \times B \,.\, w$ and assume that u is the shortest word whose action brings p and r to one state q, say. Then we have

Proposition 5. *The state q belongs to the set C of marked states.*

Proof. Let a be the last letter of the word u. If we decompose u as $u = u'a$, then, by the choice of u, the states $p' = \beta(p, u')$ and $r' = \beta(r, u')$ are still different while $\beta(p', a) = \beta(q', a) = q$.

Now recall that the word w is assumed to be n-full and hence w contains the word a^n a factor, say, $w[i]w[i+1]\cdots w[i+n-1] = a^n$. Since $\mathrm{df}_{\mathscr{B}}(w) = n-1$, we must have $\mathrm{df}_{\mathscr{B}}(a^n) \geq n-1$. This means that the decreasing sequence $B \supseteq B \,.\, a \supseteq B \,.\, a^2 \supseteq \ldots$ stabilizes after at most $n-1$ steps whence a acts on the set $B \,.\, a^{n-1}$ as a permutation. If $q \notin B \,.\, a^{n-1}$, then $q \in B \setminus B \,.\, w_{i-1}a^{n-1} = \mathrm{CES}(i+n-2) \subseteq C$. If $q \in B \,.\, a^{n-1}$, then at least one of the states p' and r' lies outside $B \,.\, a^{n-1}$ (because a acts on this set as a permutation and $\beta(p', a) = \beta(q', a) = q$). Therefore, $q \in \mathrm{PES}(i + n - 1) \subseteq C$. □

Let $|u| = k$ and, for $0 \leq j < k$, let u_j be the prefix of u of length j. The rest of the proof splits into several cases depending on the position of the states $p_j = \beta(p, u_j)$ and $r_j = \beta(r, u_j)$ with respect to the sets C and $B \,.\, w$. Clearly, if all the states p_j, r_j $(j = 0, \ldots, k-1)$ belong to the set C, then already the DFA $\mathscr{D}$ is n-compressible. We may also assume that for no $j > 0$ the pair (p_j, r_j) belongs to $B \,.\, w \times B \,.\, w$ because otherwise we could have used (p_j, r_j) instead of (p, r). Using this observations and also the facts that $|B \setminus B \,.\, w| = \mathrm{df}_{\mathscr{B}}(w) = n-1$ and $B \setminus B \,.\, w = \mathrm{CES}(\ell) \subseteq C$, we can enlarge $\mathscr{D}$ by adding at most $n + 1$ new states to an n-compressible DFA $\mathscr{A}$ retaining the property that the word w is not n-compressing with respect to $\mathscr{A}$. Again, we omit technicalities which are interesting but rather cumbersome.

In conclusion, we formulate two open questions related to Theorem 1.

1. As discussed above, the theorem implies the existence of a non-deterministic polynomial time algorithm to recognize that a given word is **not** n-collapsing. In other words, this means that the problem of recognizing n-collapsing words belongs to the complexity class co-NP. Therefore a natural question to ask is the following: is the problem of recognizing n-collapsing words co-NP-complete?

2. We have deduced from Theorem 1 that the language $\mathcal{C}_n$ of all n-collapsing words is context-sensitive. We expect that $\mathcal{C}_n$ is not context-free for $n > 1$. So far this conjecture has been proved only for the language $\mathcal{C}_2$ over two letters [18, Theorem 1]. Is it true in general?

3 Applications of Collapsing Words

Let S be a finite semigroup, Σ a finite alphabet and $\varphi : \Sigma \to S$ an arbitrary mapping. Then the DFA $\mathscr{C}_\varphi(S) = \langle S, \Sigma, \delta \rangle$ where $\delta(s, a) = s\varphi(a)$ for all $s \in S$ and $a \in \Sigma$ is called the *right Cayley graph of S with respect to φ*. The following fact (first observed in [17] for the Sauer–Stone words (1)) underlies all algebraic applications of collapsing words:

Proposition 6. *Let S be a semigroup with n elements, $\varphi : \Sigma \to S$ an arbitrary mapping and w an $(n-1)$-collapsing word over Σ. Then, for every $u \in \Sigma^*$, the word uw acts on the set $S \,.\, w$ in the right Cayley graph $\mathscr{C}_\varphi(S)$ as a permutation.*

Proof. Clearly $(S \,.\, w) \,.\, uw \subseteq S \,.\, w$. In order to show that the two sets coincide, denote the cardinality of the set $(S \,.\, w) \,.\, uw = S \,.\, wuw$ by k. Then the deficiency of the word wuw with respect to $\mathscr{C}_\varphi(S)$ is equal to $n-k$ whence the DFA $\mathscr{C}_\varphi(S)$ is $(n-k)$-compressible. It is easy to see that each $(n-1)$-collapsing word is also $(n-k)$-collapsing word for all $k = 1, \ldots, n$. Therefore the deficiency of w with respect to $\mathscr{C}_\varphi(S)$ is at least $n-k$ whence $|S \,.\, w| \leq k$. Thus, the action of uw just permutes the elements of $S \,.\, w$. □

Using some basic facts from the theory of finite semigroups, one can easily obtain that, under the condition of Proposition 6, $\varphi(w\Sigma^* w)$ is a subgroup of the semigroup S. Observe that the universal nature of collapsing words reflects in the fact that the latter claim does not depend on any structural property of S (only the cardinality of S is important). This makes collapsing words be a powerful device in reducing certain questions of finite semigroup theory to similar questions concerning groups. Various concrete examples of such usage of collapsing words can be found in [1, 17]; here we present a new application dealing with computational complexity issues.

Let S be a semigroup, Σ an finite alphabet and $u, v \in \Sigma^+$. We say that S *satisfies the identity* $u = v$ if $\varphi(u) = \varphi(v)$ for every mapping $\varphi : \Sigma \to S$. The *identity checking problem* for a finite semigroup S, ID-CHECK(S), is a combinatorial decision problem with:

INSTANCE: A semigroup identity $u = v$.
QUESTION: Does S satisfy the identity $u = v$?

Observe that the size of an instance $u = v$ of ID-CHECK(S) is just $|u| + |v|$; the semigroup S is not a part of the input, and therefore, $|S|$ (and any function of $|S|$) should be treated as a constant.

Recently, the idea of classifying finite semigroups with respect to the computational complexity of their identity checking problem has attracted a considerable attention. (Indeed, the problem is quite natural by itself and also is of interest from the computer science point of view – we refer to [7, Section 1] for a brief discussion of its relationships to formal specification theory.) So far the most complete answers have been found for the group case: it has been shown that ID-CHECK(G) is coNP-complete for each non-solvable group G [11] but is decidable in polynomial time whenever G is nilpotent or dihedral [8].

The next result reduces the identity checking problem for a certain group to the identity checking problem for a given finite semigroup S.

Theorem 2. *Let S a semigroup with n elements and G the direct product of all its maximal subgroups. Further, let Σ be a finite alphabet, w an $(n-1)$-collapsing word over Σ, and $\pi(a) = w^{n!}aw^{n!}$ for each letter $a \in \Sigma$. Then, for all $u, v \in \Sigma^+$, G satisfies the identity $u = v$ if and only if S satisfies the identity $\pi(u) = \pi(v)$.*

As discussed in Section 1, for each finite alphabet Σ, there exists an $(n-1)$-collapsing word w over Σ whose length is a polynomial of $|\Sigma|$. If one takes such a word w for constructing the mapping π in Theorem 2, then $|\pi(u)| + |\pi(v)|$ is bounded by a polynomial of $|u|+|v|$, and Theorem 2 provides a polynomial time reduction of ID-CHECK(G) to ID-CHECK(S). As an immediate consequence of this reduction and [11], we obtain that the problem ID-CHECK(S) is coNP-complete whenever S contains a non-solvable subgroup.

References

1. J. Almeida and M. V. Volkov. Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety. *J. Algebra Appl.* **2** (2003) 137–163.
2. J. Almeida and M. V. Volkov. Subword complexity of profinite words and subgroups of free profinite semigroups. *Int. J. Algebra Comp.*, accepted.
3. D. S. Ananichev, A. Cherubini, and M. V. Volkov. Image reducing words and subgroups of free groups. *Theor. Comput. Sci.*, 307, no.1 (2003), 77–92.
4. D. S. Ananichev, A. Cherubini, and M. V. Volkov. An inverse automata algorithm for recognizing 2-collapsing words. In M. Ito, M. Toyama (eds.), *Developments in Language Theory* [*Lect. Notes Comp. Sci.* **2450**] Springer, Berlin, 2003, 270–282.
5. D. S. Ananichev and M. V. Volkov. Collapsing words vs. synchronizing words. In W. Kuich, G. Rozenberg, A. Salomaa (eds.), *Developments in Language Theory* [*Lect. Notes Comput. Sci.* **2295**], Springer-Verlag, Berlin-Heidelberg-N.Y., 2002, 166–174.
6. D. S. Ananichev and I. V. Petrov. Quest for short synchronizing words and short collapsing words. *WORDS. Proc. 4th Int. Conf.*, Univ. of Turku, Turku, 2003, 411–418.
7. C. Bergman and G. Slutzki. Complexity of some problems concerning varieties and quasi-varieties of algebras. *SIAM J. Comput.* **30** (2000) 359–382.
8. S. Burris and J. Lawrence, Results on the equivalence problem for finite groups. Dept. Pure Math., Univ. of Waterloo, preprint.
9. J. Černý. Poznámka k homogénnym eksperimentom s konecnými automatami. *Mat.-Fyz. Cas. Slovensk. Akad. Vied.* **14** (1964) 208–216 [in Slovak].
10. P. Frankl. An extremal problem for two families of sets. *Eur. J. Comb.* **3** (1982) 125–127.
11. J. Lawrence. The complexity of the equivalence problem for nonsolvable groups. Dept. Pure Math., Univ. of Waterloo, preprint.
12. S. W. Margolis, J.-E. Pin, and M. V. Volkov. Words guaranteeing minimum image. *Int. J. Foundations Comp. Sci.* **15** (2004) 259–276.
13. A. Mateesku and A. Salomaa. Aspects of classical language theory. In G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages, Vol. I. Word. Language, Grammar*, Springer-Verlag, Berlin-Heidelberg-N.Y., 1997, 175–251.

14. E. Moore. Gedanken-experiments with sequential machines. In C. E. Shannon, J. McCarthy (eds.), *Automata Studies* [*Ann. Math. Studies* **34**], Princeton Univ. Press, Princeton, N.J., 1956, 129–153.
15. J.-E. Pin. Utilisation de l'algèbre linéaire en théorie des automates. *Actes du 1er Colloque AFCET-SMF de Mathématiques Appliquées*, AFCET, 1978, Tome II, 85–92 [in French].
16. J.-E. Pin. On two combinatorial problems arising from automata theory. *Ann. Discrete Math.* **17** (1983) 535–548.
17. R. Pöschel, M. V. Sapir, N. Sauer, M. G. Stone, and M. V. Volkov. Identities in full transformation semigroups. *Algebra Universalis* **31** (1994) 580–588.
18. E. V. Pribavkina. On some properties of the language of 2-collapsing words. *This volume.*
19. N. R. Reilly and S. Zhang. Decomposition of the lattice of pseudovarieties of finite semigroups induced by bands. *Algebra Universalis* **44** (2000) 217–239.
20. N. Sauer and M. G. Stone. Composing functions to reduce image size. *Ars Combinatoria* **31** (1991) 171–176.

Locally Consistent Parsing and Applications to Approximate String Comparisons

Tuğkan Batu and S. Cenk Sahinalp

School of Computing Science, Simon Fraser University
{batu,cenk}@cs.sfu.ca

Abstract. Locally consistent parsing (LCP) is a context sensitive partitioning method which achieves partition consistency in (almost) linear time. When iteratively applied, LCP followed by consistent block labeling provides a powerful tool for processing strings for a multitude of problems. In this paper we summarize applications of LCP in approximating well known distance measures between pairs of strings in (almost) linear time.

1 Introduction

Locally consistent parsing (LCP) is a method to partition a string S from an alphabet σ to short substrings in a "consistent" fashion. Consistency is achieved in the following manner: identical substrings are partitioned identically with the possible exception of their margins. We give a more precise definition of consistency later in the paper.

LCP has been introduced more than 10 years ago. Since then, it has been applied to a multitude of problems involving strings such as data structures problems, pattern matching applications, data compression, and string embeddings. Below, we give a brief history of the development of LCP and some of its key applications.

LCP is based on the Deterministic Coin Tossing (DCT) procedure of Cole and Vishkin [5], which was introduced to deterministically partition a ring of n processors, each with a unique ID, to blocks of size 2 or 3 in a distributed fashion. The procedure was iteratively applied to the representative processors (say, the leftmost one) in each block to obtain a hierarchical partitioning. This was in turn used to perform list ranking of the processors in $O(\log n)$ rounds and a total of $O(n)$ operations.

Two surprising applications of the DCT appeared (almost simultaneously) in 1994, which generalized DCT to strings with character repetitions [13, 15, 16]. LCP is this generalization of DCT to string partitioning. In [13], an efficient data structure for maintaining a dynamic collection of strings that allow equality tests, concatenation and split operations is described. In [15, 16], a novel algorithm for building the suffix tree of a given string in polylogarithmic parallel time while performing a total of $O(n)$ operations is given. Both applications are based iterative application of LCP followed by a consistent labeling of these blocks or their extensions.

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 22–35, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Later in [17], a single-pass data compression algorithm based on LCP is introduced. The number of codewords output by this algorithm is at least as many as that output by the ever popular Lempel-Ziv'77 algorithm and at most $O(\log n)$ factor higher. However, the size of the dictionary and, thus, the size of each codeword can be substantially smaller for low entropy texts, potentially resulting in a better compression. A recent paper [7] presents (among other results) an extension of this algorithm that achieves the following: after preprocessing a given string in time $O(n \log^2 n)$ compute the Lempel-Ziv'77 compressibility of any of its substrings in $O(1)$ time within an approximation factor of $O(\log n \log^* n)$.

In [18], LCP is applied to pattern matching under edit distance; here, the goal is to find all substrings of a given text string T whose edit distance to a pattern string P is no more than k. The algorithm presented here is the first one achieving $o(|P| \cdot |T|)$ time for small values of k. This result was later improved in [4] by the use of other techniques. Also in [18], an efficient data structure for dynamic text indexing based on LCP is described. Here the goal is to maintain a dynamic string, subject to single character insertions, deletions and replacements, so that substring membership queries can be answered efficiently. An improvement of this data structure, which is again based on LCP, was later described in [1]. Finally, [18] shows how to maintain a dictionary data structure under insertion and deletion of dictionary entries so that given a text string, all occurrences of each dictionary entry can be efficiently computed.

In this paper we review applications of LCP to approximately computing several variants of the edit distance between a pair of strings in almost linear time. The standard (character) edit distance between two strings can be computed exactly in quadratic time for general alphabets and slightly under quadratic time for a constant-size alphabet [11]. On the other hand, the block edit distance and edit distance with (block) moves are known to be NP-hard to compute. The task of efficiently computing these edit distance variants, even approximately, is of significance, especially in computational biology, where the data is very large and thus fast algorithms are highly desired. LCP has been successfully used to achieve this task for all three measures of string similarity.

One fast method for quickly approximating certain variants of edit distance is based on embedding strings to metric spaces with simpler-to-compute distance measures. The first such embedding is described in [14] for strings under block edit distance (the minimum number of character edits and block moves, copies and "uncopies" to transform one string into the other) into the Hamming space. The embedding, which results in a distortion of $O(\log n(\log^* n)^2)$, is computed in almost linear time, implying a fast approximation algorithm for the block edit distance. A followup result [6] gives a similar embedding of strings under edit distance with (block) moves into L_1 space with distortion $O(\log n(\log^* n)^2)$.

All the results described so far are obtained by a specific version of LCP which partitions an input string into blocks of size 2 or 3. A generalization of LCP so that the input strings are partitioned into blocks of size at least c and at most $2c-1$ for any user defined c is recently described in [3]. This particular version of LCP is applied to obtain a dimensionality reduction technique on

strings. More specifically [3] shows how to embed any string S of length n into a string of length at most n/r for any value of the *contraction parameter* r. The embedding preserves the edit distance between two strings within a factor of $\tilde{O}(r^{1+\mu})$ for some small μ. This embedding together with some annotations are later used to compute the edit distance $\mathcal{D}(S, R)$ between two strings S and R within an approximation factor of $\min\{n^{\frac{1}{3}+o(1)}, \mathcal{D}(S, R)^{\frac{1}{2}+o(1)}\}$ in almost linear time.

Overview of the paper. We start in Section 2, by giving some definitions and notation. In Section 3, we describe the (generalized) Locally Consistent Parsing method for any value of (minimum block length) c. We summarize applications of LCP to approximately computing edit distances in Section 4.1.

2 Preliminaries

Let S, R be two strings over some alphabet σ. We use $S[i]$ to denote the i^{th} character of the string S and $S[i, j]$ to denote the substring of S between positions i and j (inclusive). $|S|$ denotes the length of S and $S^r[i, j]$ denotes the reverse of $S[i, j]$. For two strings S and R, $S \circ R$ denotes the string obtained by concatenating S and R. We denote by $\mathcal{D}(S, R)$ the edit distance between S and R, i.e., the minimum number of character insertions, deletions, and substitutions needed to obtain R from S.

An alignment between S and R associates each character of S (and R) with at most one character from R (correspondingly S) such that given $i < j$, if $S[i]$ is aligned with $R[i']$ and $S[j]$ with $R[j']$ then $i' < j'$. An optimal alignment between S and R minimizes the sum of the number of unaligned characters and misalignments (alignments between non-identical characters) in S (and R). The sum of the number of unaligned characters in S and R and misalignments between S and R in an optimal alignment is equal to $\mathcal{D}(S, R)$.

Another measure of similarity between the strings S and R is the block edit distance, denoted $\mathcal{BED}(S, R)$, which is defined to be the minimum number of character edits and *block edits* to transform one string into the other. Character edits are insertion, deletion and replacement of a single character. Block (substring) edits are relocating an entire substring, copying a substring from one location to another and "uncopying" a block; i.e. deleting one of the two copies of a substring[1].

One final measure of similarity between strings S and R is edit distance with moves, denoted by $\mathcal{MV}(S, R)$. Here, all single character edit operations as well as substring relocation operation is allowed.

Metric embeddings. Many of the applications of LCP to "approximately" compare strings are based on embedding strings under various edit distances to other

[1] Other versions of block edit distance that allow substring reversals and linear transformations on substrings have also been described. Here we only focus on the "vanilla" block edit distance

metric spaces. Given two metric spaces $M_1 = (X_1, D_1)$ and $M_2 = (X_2, D_2)$, where X_i is the universe and D_i is the distance measure for metric M_i, $\phi : X_1 \rightarrow X_2$ is an embedding with distortion $d = d_1 \cdot d_2$ for $d_1, d_2 \geq 0$, if, for any $y, z \in X_1$,

$$D_1(y, z)/d_1 \leq D_2(\phi(y), \phi(z)) \leq d_2 \cdot D_1(y, z).$$

3 A Generalized Description of Locally Consistent Parsing

Locally Consistent Parsing is a *consistent* way of partitioning any string S into (non-overlapping) blocks such that the minimum block length is c and the maximum block length is $2c - 1$. For full generality, the maximum block length cannot be less than $2c - 1$; for instance, if $|S| = 2c - 1$, then S cannot be partitioned into blocks with length in the range $[c, 2c - 2]$. The blocks obtained will be consistent in the following sense: if two identical substrings $S[i, i + b]$ and $S[j, j + b]$ appear in long enough identical "contexts" $S[i - \gamma(c), i + b + \gamma'(c)]$ and $S[j - \gamma(c), j + b + \gamma'(c)]$ for increasing functions $\gamma(c), \gamma'(c)$, and if $S[i, i + b]$ is identified as a block then $S[j, j + b]$ must be identified as a block. Note that $c \leq b + 1 \leq 2c - 1$.

Observe that a single edit operation on S will only have a local effect on such a partitioning of S. A single edit can change (1) the block in which it lies, and (2) the blocks whose neighborhoods, as defined by $\gamma()$ and $\gamma'()$, contain the edit operation. Hence the number of blocks that can change as a result of a single edit operation is $O((\gamma(c) + \gamma'(c))/c)$.

LCP was originally described for $c = 2$. This basic case, which we denote by LCP(2), relies on the Deterministic Coin Tossing technique [5]; it is quite simple but is sufficiently powerful to achieve the desired performance in many applications, some of which are described here. The more general LCP(c)was introduced in [3] to solve a number of problems where the choice of c turns out to be of crucial importance. Here we describe the general version while illustrating the main ideas through examples based on LCP(2).

3.1 Description of LCP(c) for Small Alphabets

Given input string S, LCP(c) treats repetitive and nonrepetitive substrings of S differently. Repetitive substrings are partitioned in a straightforward way; for partitioning non-repetitive substrings, a generalized version of the Deterministic Coin Tossing technique is used to guarantee block sizes of c to $2c - 1$.

Here we describe LCP(c) for small alphabets; in Section 3.2, we show how to generalize it to integer alphabets. We start off by describing how to identify the repetitive substrings of the input string.

Definition 1. *A string R is called* r-repetitive *if it is of the form Q^ℓ where $\ell \geq 2$ and Q, the* repetition*, is a string of length r. Given a string S, a substring R of S is called* maximally r-repetitive *if (1) it is r-repetitive with repetition T,*

where T is the substring that is the lexicographically greatest substring among all length-r substrings of R, and (2) the length-r substring following or preceding R (in S) is not T.

For example, for $T = ababa$, as well as $T' = babab$, the only substring that is maximally 2-repetitive is $baba$. This information is helpful since it implies that T and T' have a long common substring. Note that every maximally repetitive string is periodic but not all periodic strings are maximally repetitive, e.g., $abab$ and $ababa$ are both periodic with period 2 but are not maximally repetitive since ab, a substring of both, is lexicographically smaller than ba.

LCP(c) performs the partitioning of S in two phases. Phase 1 partitions S into substrings that are maximally ℓ-repetitive for $\ell < c$ and maximally non-repetitive as follows. For $r = c-1, \ldots, 1$, LCP(c) extracts all maximally r-repetitive substrings of S of length at least c that so far remain unextracted. All the remaining substrings (of maximal length) are identified as maximally non-repetitive substrings.

For example, if $S = aabaaaababa$ and $c = 2$, then LCP(c) will first identify $S[1,2] = aa$ and $S[4,7] = aaaa$ as maximally 1-repetitive substrings.

Phase 2 further partitions the substrings extracted in Phase 1 to obtain blocks of length c to $2c-1$.

For partitioning repetitive substrings, each maximally r-repetitive substring is partitioned into blocks of length t where t is the smallest multiple of r greater than c. If the substring length is not divisible by t, the two leftmost blocks can be arranged so that the leftmost one is of size c. (This choice is arbitrary.)

For partitioning maximally non-repetitive substrings, first, any non-repetitive substring Q of length less than c is merged with the (necessarily repetitive) block to its left. If Q is a prefix of S, it is merged with the (again necessarily repetitive) block to its right. If the resulting block is of length greater than $2c$, it is partitioned (arbitrarily) into two blocks such that the left one is of length c.

In the above example, $S[1,2] = aa$ will be identified as a single block of size 2 and $S[4,7] = aaaa$ will be partitioned into two blocks $S[4,5]$ and $S[6,7]$. The non-repetitive block $S[3,3] = b$ is then merged to the block to its right to form the block $S[3,5] = baa$.

For non-repetitive substrings of length at least c LCP(c) performs a more sophisticated partitioning scheme that ensures partition consistency as stated earlier. To achieve this, whether a character is selected to be a block boundary depends on the character's immediate neighborhood. The operations defined below facilitate the comparison of a character to other characters in its neighborhood.

Definition 2. *Given two distinct binary words w and y of length k each, let $f_y(w)$ be a binary word of length $k' = \lceil \log k \rceil + 1$, defined as the concatenation of (i) the position of the rightmost bit b of w where w differs from y, represented as a binary number (counted starting from 1 at the right end), and (ii) the value of w at bit b. We define $f_w(w) = 0^{k'}$.*

For example, $f_{1111}(1101) = 0100$ as the position of the rightmost bit of 1101 that differs from 1111 is 2 (010 in binary) and its value is 0.

Definition 3. *For a character $S[i]$ and positive integers c and ℓ, we define*

$$g_{c,\ell}(S[i]) \stackrel{\text{def}}{=} f_{S[i-c-\ell+2,i+c-\ell-2]}(S[i-c+2, i+c-2]).$$

If $S[i]$ is represented by a k-bit word, then $g_{c,\ell}(S[i])$ is a k'-bit word where $k' = \lceil \log((2c-3)k) \rceil + 2$.

Intuitively, $g_{c,\ell}(S[i])$ relates the substring of size $2c-3$ around $S[i]$ to that of size $2c-3$ around $S[i+\ell]$.

Given a maximally non-repetitive substring Q, LCP(c) generates an auxiliary substring Q' to help identify some of the block boundaries. Let $\lceil \log_2 |\sigma| \rceil = k$, where σ is the alphabet. For each $Q[i]$ (represented as a k-bit word), let

$$Q'[i] \stackrel{\text{def}}{=} g_{c,c-1}(Q[i]) \circ g_{c,c-2}(Q[i]) \circ \ldots \circ g_{c,1}(Q[i]).$$

The characters of Q' are represented as words of length $k' = (c-1) \cdot (\lceil \log((2c-3)k) \rceil + 2) = O(c \log(ck))$ bits. Thus the construction of Q' from Q constitutes an *alphabet reduction* as long as $k' < k$. Since Q' will only be used to determine block boundaries, the information loss resulting from this alphabet reduction is not problematic.

Lemma 1. *Let Q be a non-repetitive substring and let $Q'[3c-5, |Q|-c+2]$ be the string obtained from $Q[3c-5, |Q|-c+2]$ after the alphabet reduction. Then $Q'[3c-5, |Q|-c+2]$ is non-repetitive.*

Proof. Observe that given binary words x, y, z, such that $x \neq y$ and $y \neq z$, if the position of the rightmost bit b of x that differs from y is identical to the position of the rightmost bit b' of y that differs from z, then the bit values of b and b' must be different; i.e., $f_x(y) \neq f_y(z)$.

Fix $i \in [4c-6, |Q|-c+2]$ and $\ell \in [1, c-1]$. Consider

$$g_{c,\ell}(Q[i]) = f_{Q[i-c-\ell+2,i+c-\ell-2]}(Q[i-c+2, i+c-2])$$

and

$$g_{c,\ell}(Q[i-\ell]) = f_{Q[i-c-2\ell+2,i+c-2\ell-2]}(Q[i-c-\ell+2, i+c-\ell-2]).$$

Now, let $x = Q[i-c-2\ell+2, i+c-2\ell-2]$, $y = Q[i-c-\ell+2, i+c-\ell-2]$, and $z = Q[i-c+2, i+c-2]$.

Note that $x \neq y$; otherwise $Q[i-c-2\ell+2, i+c-\ell-2]$ includes an ℓ-repetitive substring of length more than c (which is impossible for a non-repetitive substring extracted by the algorithm). Similarly, $y \neq z$. Using the opening observation of the proof, we have $g_{c,\ell}(Q[i-\ell]) = f_x(y) \neq f_y(z) = g_{c,\ell}(Q[i])$. Hence, $Q'[i-\ell] \neq Q'[i]$. □

Now we are ready to identify the block boundaries on the nonrepetitive substring Q, using information from Q'. A character $Q[i]$ is set to be a *primary marker* if $Q'[i]$ is lexicographically greater than each character in its immediate neighborhood of length $2c-1$, namely, in $Q'[i-c+1, i+c-1]$. Note that primary markers are set in Q; Q' is solely used in helping determine their locations.

Lemma 2. *Let $Q[i]$ and $Q[j]$ be two consecutive primary markers in Q such that $i < j$. Then, $c \leq j - i \leq 4(2^{k'}) = O((kc)^c)$.*

Proof. We first give the proof for the lower bound. Assume for contradiction that both $Q[i]$ and $Q[i+\ell]$ are primary markers for $\ell < c$. Let Q' be the string obtained from Q by the alphabet reduction. Then, by definition of a primary marker, $Q'[i]$ must be lexicographically greater than $Q'[i+\ell]$ and $Q'[i+\ell]$ must be lexicographically greater than $Q'[i]$, a contradiction. Thus, both $Q[i]$ and $Q[i+\ell]$ cannot be primary markers for $\ell < c$.

We now give the proof for the upper bound. It is by induction on the size t of the alphabet for Q'. The bound holds for $t = 3$. By Lemma 1, we know that $Q'[i] \notin \{Q'[i-c+1], Q'[i-1], Q'[i+1], Q'[i+c-1]\}$. Also, recall that the characters of Q' are represented by binary strings of length k' (i.e., $t = 2^{k'}$). Without loss of generality, assume that both $Q'[i]$ and $Q'[j]$ are the lexicographically maximal alphabet character in σ' (this is the worst case). Then, since there are no primary markers between i and j, no character in $Q'[i+1, j-1]$ is the lexicographically maximal character. Moreover, in $Q'[i, j]$, the character just below the lexicographically maximal character can only be in positions $i+1, i+2, j-2, j-1$; otherwise, another primary marker would have been set. Without loss of generality, assume $Q'[i+2]$ and $Q'[j-2]$ are at most this lexicographically second largest character. Then, by the induction hypothesis, $j - 2 - (i+2) \leq 4(t-1)$. Thus, we get $j - i \leq 4t$. □

Having established the primary markers, LCP(c) now partitions Q into blocks as follows. Q is partitioned into the substrings that are between two consecutive primary markers (inclusive of the left primary marker), the one to the left of the leftmost primary marker, and the one to the right of the rightmost primary marker. Each of these substrings is further partitioned into blocks of length c; if the substring length is not divisible by c, the leftmost block will be of length between $c+1$ and $2c-1$. The next claim then follows.

Claim. If $S[i, j]$ is a block obtained by LCP(c) then $c \leq j - i + 1 \leq 2c - 1$.

The consistency of the above partitioning is established in the following manner. If $S[i, j]$ and $S[i', j']$ are two identical non-repetitive substrings of S of sufficient length, then the blocks within $S[i, j]$ and $S[i', j']$, except at the left and right ends, are identical, regardless of the locations of $S[i, j]$ and $S[i', j']$ in S.

Lemma 3. *Suppose that for some $b \in [c-1, 2c-2]$, $S[i - 2^{k'+2} - 4c + 7, i + b + 4c - 3] = S[i' - 2^{k'+2} - 4c + 7, i' + b + 4c - 3]$, and furthermore, both substrings are parts of substrings identified as being maximally nonrepetitive in S. Then, if $S[i, i+b]$ is set as a block by LCP(c), then so is $S[i', i'+b]$.*

Proof. By definition of primary markers and LCP(c), whether a character $S[\ell]$ (within a non-repetitive substring) is set as a marker depends only on $S[\ell - 4c + 7, \ell + 2c - 3]$. Since the decision to set $S[i, i + b]$ as a block depends only on the primary marker immediately to the left of $S[i]$ and whether there is a primary maker before $S[i + 2c]$, we can conclude that this decision depends only on $S[i - 2^{k'+2} - 4c + 7, i + b + 4c - 3]$ by Lemma 2. As a result, $S[i, i + b]$ is set as a block only if $S[i', i' + b]$ is set as a block as well. □

In the preceding discussion, we described how to partition a nonrepetitive string Q, where Q is over alphabet σ such that $\lceil \log |\sigma| \rceil = k$, into blocks of size between c and $2c - 1$ while maintaining a consistency property formalized in Lemma 3. This lemma guarantees that identical substrings are partitioned identically except in the margins. This implies thus that a single edit operation cannot change the partitioning of a string by "too much." More specifically, the number of blocks that can change as a result of an edit operation is $O((ck)^c)$ in a non-repetitive substring (by Lemma 3) and is only a constant in a repetitive substring.

3.2 Iterative Reduction of the Alphabet Size

If $c \cdot k = O(1)$, by the above discussion, each edit operation results in only $O(1)$ changes in the partitioning of the input string and thus one application of the alphabet reduction suffices to obtain the desired partition. For $ck = \omega(1)$, there is a need to reduce the alphabet size further before setting the primary markers in order to guarantee that an edit operation will have limited effect on the partitioning. In this case, LCP(c) performs the alphabet reduction on each non-repetitive substring $S[i, j]$ of S, for $\log^* kc + O(1)$ iterations before setting the primary markers. Let $S^*[i, j]$ be the output of this process. Due to Lemma 1, since $S^*[i, j]$ is non-repetitive, so is $S^*[i, j]$. In the first iteration the alphabet size will be reduced to $O((ck)^c)$; in the second it will be $O((c^2 \log ck)^c)$ and so on. After $\log^* kc + O(1)$ iterations, the alphabet size will be $O((3c^2 \log c)^c)$, which is independent of k. The primary markers of $S[i, j]$ are then chosen as the local maxima in $S^*[i, j]$; this will assure that the maximum distance between two primary markers will be $O((3c^2 \log c)^c)$ as well. (Recall that the alphabet reduction is only for the purpose of obtaining primary markers. Afterwards, the partitioning is performed on the original string.)

Theorem 1. *A sequence of h single character edit operations to a string S can change at most $O(h \cdot [(3c^2 \log c)^c/c + \log^* kc])$ and at least $h/(2c - 1)$ blocks in the sequence of blocks obtained by LCP(c).*

Proof. The lower bound follows from the fact that the maximum block size is $2c - 1$ and thus the minimum possible number of blocks that can contain all h edit operations is $h/(2c - 1)$.

The upper bound follows from repeated application of the next lemma.

Claim. An edit operation on S can change only $O((3c^2 \log c)^c/c + \log^* kc)$ blocks obtained by LCP(c).

Proof. LCP(c) initially partitions S into non-repetitive and r-repetitive substrings for $1 \leq r < c$.

Suppose the edit operation is performed on a non-repetitive substring, which remains non-repetitive after the operation. The first alphabet reduction on any $S[i]$ depends only on $S[i-3c+6, i+c-2]$. In general, the j^{th} application of the alphabet reduction on $S[i]$ depends on the substring $S[i-(3c-6)j, i+(c-2)j]$. Thus, for $j = \log^* kc + O(1)$, the output of the j^{th} alphabet reduction on $S[i]$ will be of size $O((3c^2 \log c)^c)$ and depend only on a substring of size $4c(\log^* kc + O(1))$ that contains $S[i]$. This further implies that the decision of whether to choose $S[i]$ as a primary marker also depends only on a size $4c(\log^* kc + O(1)) + O((3c^2 \log c)^c)$ substring that contains $S[i]$. All blocks within this substring can change as a result of an edit operation on $S[i]$, implying a change of $4(\log^* kc + O(1)) + O((3c^2 \log c)^c/c)$ blocks. As the distance between the first changed primary marker and its preceding primary marker is $O((3c^2 \log c)^c)$, a further $O((3c^2 \log c)^c/c)$ blocks can change as a result.

If the edit operation is performed on a non-repetitive substring that becomes repetitive then the same argument applies: The new repetitive substring splits the non-repetitive substring into two. This can change $4(\log^* kc + O(1)) + O((3c^2 \log c)^c/c)$ blocks on the two sides of the new repetitive substring.

If the edit operation is performed on a repetitive substring then the exact locations of the blocks may change; however only $O(1)$ of these blocks will change content. That is, one has to edit only $O(1)$ blocks in the original string in order to obtain the partitioning of the modified string. □

This completes the proof of Theorem 1. □

Lemma 4. *LCP(c) runs in time* $O(n[c \log c + (k+c) \log^* kc])$.

Proof. Clearly the partitioning of a repetitive substring into blocks can be done in linear time in the size of the substring. We now show that the partitioning of all non-repetitive substrings of S takes $O(n[c \log c + (k+c) \log^* kc])$ time.

We first focus on the time for the first application of the alphabet reduction on a given $S[i]$ to obtain $S'[i]$. Consider the compact trie T_S that comprises the bitwise representations of $S^r[j-c+2, j+c-2]$ for all j. T_S can be obtained in $O(nk)$ time using any linear-time suffix-tree construction algorithm (e.g., [12]). After preprocessing T_S in $O(n)$ time, the lowest common ancestor (LCA) of two leaves representing $S^r[i-c+2, i+c-2]$ and $S^r[i'-c+2, i'+c-2]$ for any $i-c+1 \leq i' < i$ can be found in $O(1)$ time (c.f., [8, 19]). The LCA of these leaves gives $g_{c,i-i'}(S[i])$. To obtain $S'[i]$ one only needs to compute $g_{c,i-i'}(S[i])$ for all i' such that $i-c+1 \leq i' < i$; this can be done in time $O(c)$. Thus the overall running time for performing the alphabet reduction for all characters of S is $O(nk+nc)$.

Subsequent $O(\log^* kc)$ applications of the alphabet reduction work on smaller size alphabets; thus the overall running time is $O(n(k+c) \log^* kc)$.

After the alphabet reduction, determining whether each $S[i]$ is a primary marker can be done as follows. The number of bits needed to represent $S^*[i]$ is $O(c \log c)$; because $c \leq n$, this requires $O(c)$ machine words. One can use a

priority queue that includes each one of the $O(c)$ characters in the substring $S^*[i-c+2, i+c-2]$ to determine whether $S^*[i]$ is the local maxima. This can be done, for all i, in time $O(nc \log c)$.

Once the primary markers are obtained, the final partitioning can be obtained in $O(n)$ time. □

4 Applications of LCP to Approximate String Comparisons

Among the applications of LCP our focus will be embeddings of strings under character and block edit distances to other metric spaces such as the Hamming Space or L_1, as well as to "shorter" strings (such an embedding is called a dimensionality reduction). These embeddings (sometimes together with additional information) give fast algorithms to approximately compute the edit distance variants of interest. Here we give a brief description of the techniques underlying each application as well as theorem statements without proofs.

4.1 Dimensionality Reduction in Strings Under Edit Distance

An embedding ϕ from $M_1 = (X_1, D_1)$ to $M_2 = (X_2, D_2)$, is a dimensionality reduction if $D_1 = D_2$ and each element in X_1 is mapped to an element with shorter representation in X_2. A string embedding with *contraction* $r > 1$ is an embedding from strings of length at most n over an alphabet σ_1 under edit distance, to strings of length at most n/r over another alphabet σ_2, again under edit distance, which contracts the length of the string to be embedded by a factor of at least r. Thus, such an embedding is a dimensionality reduction. A proof for the following basic lemma can be found in [3].

Lemma 5. *A string embedding with contraction $r > 1$ cannot have a distortion d less than r.*

Here we present a dimensionality reduction technique for strings which follows from an iterative application of LCP(c) followed by consistent labeling of blocks [3][2]. The number of iterations, ℓ, is determined by the contraction parameter r as follows.

Given input strings S and R and $|\sigma| = 2^k$, denote by S_1 and R_1 the strings that are obtained after a single application of LCP(c), respectively. Now, denote by S_ℓ the string obtained by applying LCP(c) on $S_{\ell-1}$ followed by consistent block labeling. Each label in S_ℓ corresponds to a substring of S with size in the range $[c^\ell, (2c-1)^\ell]$.

Theorem 1 implies the following lemma.

[2] The label of a block could either be set to the block's contents, implying a large alphabet, or be computed via a hash function, introducing a small probability of error

Lemma 6. *A single edit operation on S results in $O((3c^2 \log c)^c/c + \log^* kc)$ labels to change in S_ℓ. Thus,*

$$\mathcal{D}(S,R)/(2c-1)^\ell \leq \mathcal{D}(S_\ell, R_\ell) \leq \mathcal{D}(S,R) \cdot O((3c^2 \log c)^c/c + \log^* kc).$$

Now let $c = O((\log\log n)/\log\log\log n)$; then, $\phi(S) = S_{\lceil \log_c r \rceil}$ provides a string embedding with distortion (roughly) r, which is almost optimal.

Theorem 2. *Given string S and $r > 1$, the embedding $\phi(S) = S_{\lceil \log_c r \rceil}$ has contraction r and distortion $\tilde{O}(r^{1+\mu})$, where $\mu = \Omega(2/\log\log\log n)$. The embedding can be computed in time $\tilde{O}(2^{1/\mu} \cdot n)$.*

4.2 Embedding Strings into L_1 and Hamming Space

In this section we describe two very similar results: (i) embedding strings under block edit distance to the Hamming space [14] and (ii) embedding strings under edit distance with moves to L_1 [6].

The embedding of a string S of length n under block edit distance to a Hamming vector is based on iterative application of LCP(2). The consistent labeling following LCP(2), however, is not performed on blocks obtained by LCP(2). Rather, it is applied to "extended blocks" that are called *core* blocks, which are defined as follows. For each block divider between $S[i-1]$ and $S[i]$, there is a corresponding core block, which is, for some $b = O(\log^* n)$, the string $S[i-b, i+b]$ [3]. Accordingly, let S'_1 be the sequence of labels of the core blocks implied by LCP(2) applied on S. One can define, for $\ell > 1$, the string S'_ℓ as the sequence of labels of the core blocks implied by LCP(2) applied on $S'_{\ell-1}$. Notice that each core block at level ℓ corresponds to a substring of S; this substring is called a *core substring* of S. Appropriate choice of b ensures the following property of the core substrings [14].

Lemma 7. *If Q is a non-repetitive core substring of S and Q' is another substring of S identical to Q, then Q' must be a core substring of S as well.*

We now describe the embedding of S into a binary vector; without loss of generality, we assume that $\sigma = \{0, 1\}$.

Definition 4. *Given string S, its* level-i binary histogram, *denoted $T_i(S)$, is the binary vector of size c^i where $c = O(\log^* n)$; the jth entry of $T_i(S)$, denoted $T_i(S)[j]$, is 1 if the binary representation of j is a core substring of S whose corresponding core block is in S'_i. If j is not a core of S then $T_i(S)[j] = 0$. The embedding $\phi'(S)$ is then the concatenation of all $T_i(S)$ for $i = 0, \ldots, \log n$.*

The embedding $\phi'(S)$ is an $O(2^{|S|})$ dimensional binary vector whose j^{th} entry is set to 1 if the corresponding core is present in S. Although the number of dimensions in $\phi'(S)$ is very large, it is possible to represent it in $O(n)$ space by

[3] The LCP(2) description in [14] is a slightly different from the one provided here for the purpose of tolerating substring reversal as a block edit operation

simply listing the $O(n)$ core strings of S by the use of pointers. This alternative representation can be computed in time $O(n \cdot \log^* n)$.

The following lemma is implied by the proof of Theorem 1.

Lemma 8. *Let $C(S)$ denote the set of core blocks obtained by LCP(2) on a string S. A sequence of h single character edits, block copy, block move and block uncopy operations to S can add or remove most $O(h \log^* n)$ and at least $\Omega(h)$ core blocks from $C(S)$.*

As a result, given two strings S and R s.t. $|S|, |R| \leq n$, their embeddings $\phi'(S)$ and $\phi'(R)$ satisfy the following.

Theorem 3. *$\Omega(\mathcal{BED}(S,R))/\log^*(n) = h(\phi'(S), \phi'(R)) = O(\mathcal{BED}(S,R) \log n \log^* n)$; here $h(,)$ denotes the Hamming distance.*

The embedding of strings under block edit distance to Hamming vectors, together with efficient data structures for $(1+\epsilon)$-factor approximate nearest neighbor search in the Hamming space [9, 10], can be used to obtain an $O(\log n \log^* n)$ factor approximate nearest neighbor search data structure for strings under block edit distance. The preprocessing time and space for the data structure is $O((nm)^{O(1)})$ where m is the number of strings in the data structure; the query time is $O(n \cdot \text{polylog}(nm))$. This is the first and yet the only known approximate nearest neighbor search data structure for strings under non-trivial edit operations that achieves almost optimal preprocessing time, space and query time, while guaranteeing an approximation factor polylogarithmic in the query size.

A very similar embedding $\phi''(S)$ from strings under edit distance with moves to L_1 space is described in [6]. Without loss of generality, we describe this embedding for $\sigma = \{0, 1\}$.

Definition 5. *Given string S, its* level-i histogram*, denoted $H_i(S)$, is the integer vector of size c^i ($c = O(\log^* n)$) where the jth entry of $H_i(S)$, denoted $H_i(S)[j]$, is the number of occurrences of the binary representation of j as a core substring in S, whose corresponding core block is in S''_i. The embedding $\phi''(S)$ is the concatenation of all $H_i(S)$ for $i = 0, \ldots, \log n$.*

The embedding ϕ'' can be shown to satisfy the following property based on a variation of Lemma 8.

Theorem 4. *$\Omega(\mathcal{MV}(S,R)) = ||\phi'(S) - \phi'(R)||_1 = O(\mathcal{MV}(S,R) \log n \log^* n)$.*

4.3 Approximately Computing the Edit Distance in (Near) Linear Time

Given $\gamma > 1$, a γ-factor approximation algorithm for $\mathcal{D}(S,R)$ outputs a value d such that $\mathcal{D}(S,R) \leq d \leq \gamma \cdot \mathcal{D}(S,R)$. Recently Bar-Yossef et al. developed an algorithm that computes $\mathcal{D}(S,R)$ within an approximation factor of $n^{3/7}$ in $\tilde{O}(n)$ time [2]. It was recently shown in [3] a method to apply LCP to compute

the edit distance between two strings within an approximation factor of $n^{\frac{1}{3}+o(1)}$ again in $\tilde{O}(n)$ time. Here we summarize this result.

Let S_ℓ and R_ℓ be as described in Section 4.1 for $c = (\log n)/\log\log n$. We have already demonstrated that $\mathcal{D}(S_\ell, R_\ell)$ approximates $\mathcal{D}(S, R)$ within a factor of $\tilde{O}((2c-1)^\ell)$. By the use of standard dynamic programming, $\mathcal{D}(S_\ell, R_\ell)$ can be computed in time $O((n/c^\ell)^2)$. However, if an upper bound t on $\mathcal{D}(S, R)$ is given, it is possible to approximate $\mathcal{D}(S, R)$ by computing $\mathcal{D}(S_\ell, R_\ell)$ only along a band of width $2t/c^\ell$ around the main diagonal of the dynamic programming table.

By applying this strategy for potential upper bounds on $\mathcal{D}(S, R)$, $t = 2^i$, for $i = 1, \ldots, O(\log n)$, followed by checking t indeed provides an upper bound, it is possible to obtain an approximation to $\mathcal{D}(S, R)$ as per [3].

Theorem 5. *One can compute $\mathcal{D}(S, R)$ within an approximation factor of*

$$\min\{n^{\frac{1}{3}+o(1)}, \mathcal{D}(S, R)^{\frac{1}{2}+o(1)}\}$$

in time $\tilde{O}(n)$.

References

1. Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. Pattern matching in dynamic texts. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 819–828, 2000.
2. Ziv Bar-Yossef, T.S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.
3. Tugkan Batu, Funda Ergun, and S. Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. Technical Report TR2005-11, School of Computing Science, Simon Fraser University, 2005.
4. Richard Cole and Ramesh Hariharan. Approximate string matching: A simpler faster algorithm. In *SODA*, pages 463–472, 1998.
5. Richard Cole and Uzi Vishkin. Deterministic coin tossing and accelerating cascades: Micro and macro techniques for designing parallel algorithms. In *ACM Symposium on Theory of Computing (STOC)*, 1986.
6. Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *SODA*, pages 667–676, 2002.
7. Graham Cormode and S. Muthukrishnan. Substring compression problems. In *SODA*, 2005.
8. Dov Harel and Robert E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, May 1984.
9. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.
10. Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *STOC*, pages 614–623, 1998.
11. William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, February 1980.

12. Edward M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, 1976.
13. Kurt Mehlhorn, R. Sundar, and Christian Uhrig. Maintaining dynamic sequences under equality-tests in polylogarithmic time. In *SODA*, pages 213–222, 1994.
14. S. Muthukrishnan and S. Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. In *STOC*, pages 416–424, 2000.
15. S. Cenk Sahinalp and Uzi Vishkin. On a parallel-algorithms method for string matching problems. In *CIAC*, pages 22–32, 1994.
16. S. Cenk Sahinalp and Uzi Vishkin. Symmetry breaking for suffix tree construction. In *STOC*, pages 300–309, 1994.
17. S. Cenk Sahinalp and Uzi Vishkin. Data compression using locally consistent parsing. *UMIACS Technical Report*, 1995.
18. S. Cenk Sahinalp and Uzi Vishkin. Efficient approximate and dynamic matching of patterns using a labeling paradigm. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1996.
19. Baruch Schieber and Uzi Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM Journal on Computing*, 17(6):1253–1262, December 1988.

Central Sturmian Words: Recent Developments

Arturo Carpi[1] and Aldo de Luca[2]

[1] Dipartimento di Matematica e Informatica dell'Università di Perugia
Via Vanvitelli 1, 06123 Perugia, Italy
carpi@dipmat.unipg.it
[2] Dipartimento di Matematica e Applicazioni dell'Università di Napoli "Federico II"
via Cintia, Monte S. Angelo, 80126 Napoli, Italy
aldo.deluca@unina.it

Abstract. A word w is central if it has two periods p and q which are coprime and such that $|w| = p+q-2$. Central words play an essential role in the combinatorics of Sturmian words. The aim of this paper is to give an overview on central words focusing some recent developments in the study of their structure, combinatorics, and arithmetics. Moreover, some results are concerned with remarkable languages of central words such as central codes and Farey's codes. Another interesting class of central languages is given by the so-called Farey languages which give faithful representations of Farey's series.

1 Introduction

A Sturmian word is an infinite word such that for any integer $n \geq 0$ the number of its distinct factors is equal to $n + 1$. Sturmian words are of great interest in various fields such as Algebra, Theory of numbers, Physics, and Computer science. They have been studied for at least two centuries with a great number of research papers on the subject mainly in recent years. Two valuable overviews on Sturmian words are in [4, Chap. 2] and [1, Chap.s 10–11].

A geometrical definition of a Sturmian word is the following: consider the sequence of the cuts (*cutting sequence*) in a squared lattice made by a ray having a slope which is an irrational number. A horizontal cut is denoted by the letter b, a vertical cut by a and a cut with a corner by ab or ba. Any such cutting sequence is a Sturmian word. Sturmian words represented by cutting sequences produced by rays starting from the origin are called *standard* or *characteristic*.

The most famous Sturmian word is the *Fibonacci word*

$$f = abaababaabaababaababaabaababaabab\cdots .$$

which is the limit of the sequence of words $(f_n)_{n\geq 0}$, inductively defined by:

$$f_0 = b,\ f_1 = a,\ \text{and } f_{n+1} = f_n f_{n-1} \text{ for } n \geq 1\,.$$

Standard Sturmian words can be equivalently defined in the following way which is a natural generalization of the definition of the Fibonacci word. Let

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 36–56, 2005.
© Springer-Verlag Berlin Heidelberg 2005

$c_0, c_1, \ldots, c_n, \ldots$ be any sequence of integers such that $c_0 \geq 0$ and $c_i > 0$ for $i > 0$. We define, inductively, the sequence of words $(s_n)_{n\geq 0}$, where

$$s_0 = b,\ s_1 = a,\ \text{and}\ s_{n+1} = s_n^{c_n - 1} s_{n-1}\ \text{for}\ n \geq 1\,.$$

The sequence $(s_n)_{n\geq 0}$ converges to a limit s which is an infinite standard Sturmian word. Any standard Sturmian word is obtained in this way. We shall denote by *Stand* the set of all the words s_n, $n \geq 0$ of any *standard sequence* $(s_n)_{n\geq 0}$. Any word of *Stand* is called *finite standard Sturmian word*.

In the study of combinatorial properties of Sturmian words a crucial role is played by the set *PER* of all finite words w having two periods p and q such that $\gcd(p, q) = 1$ and $|w| = p + q - 2$.

The set *PER* was introduced in [14] where its main properties were studied. In particular, it has been proved that *PER* is equal to the set of the palindromic prefixes of all standard Sturmian words. The words of *PER*, which are in a two-letter alphabet, have been called *central Sturmian words*, or simply *central words* in [4].

The aim of this paper is to give an overview on central words focusing some recent developments in the study of their structure, combinatorics, and arithmetics. Moreover, some results are concerned with remarkable languages of central words (*central languages*) such as the class of *harmonic* and *gold* words [7]. A further interesting case is when the language is a code (*central code*) [8]. A suitable class of central codes are the so-called *Farey codes* which are maximal prefix central codes. Another remarkable class of central languages are the so-called *Farey languages of order* n which give faithful representations of Farey's series of order n [9].

2 Central Words

The periodicity theorem of Fine and Wilf (cf. [18]) states that if a word w has two periods p and q and length $|w| \geq p + q - \gcd(p, q)$, then w has also the period $\gcd(p, q)$. Therefore a central word having the coprime periods p and q is either a power of a single letter or it has the maximal length $p + q - 2$ to which the theorem of Fine and Wilf does not apply. The empty word ε is assumed to be a central word (this is formally coherent with the definition if one takes $p = q = 1$).

The set *PER* of all central words on a fixed binary alphabet $\mathcal{A} = \{a, b\}$ has remarkable structural properties (cf. [2, 4, 10–12, 14]). For instance, *PER* is equal to the set of palindromic prefixes of all standard Sturmian words. Moreover, the set *St* of all finite factors of all Sturmian words equals the set of factors of *PER*. The set *Stand* of all finite standard Sturmian words is given by

$$\mathit{Stand} = \mathcal{A} \cup \mathit{PER}\{ab, ba\}\,. \tag{1}$$

Thus, any finite standard Sturmian word which is not a single letter is obtained by appending ab or ba to a central word. We recall [14] that for any $n \geq 0$ the

number of central words of length n is given by $\phi(n+2)$, where ϕ is the *Euler totient function*.

The following useful characterization of central words is a slight generalization of a statement proved in [12] (see also [8, 17]).

Proposition 1. *A word w is central if and only if w is a power of a single letter or it satisfies the equation*

$$w = w_1abw_2 = w_2baw_1$$

with $w_1, w_2 \in \mathcal{A}^*$. Moreover, in this latter case, w_1 and w_2 are central words, $p = |w_1| + 2$ and $q = |w_2| + 2$ are coprime periods of w, and $\min\{p, q\}$ is the minimal period of w.

From the previous proposition and (1) one derives that any standard word is the product of two palindromes.

There exist palindromes in St which are not central. For instance, there are 20 palindromes of length 9, whereas the number of central words of length 9 is 10. One can prove that any palindrome in St is a 'median' factor of a central word (cf. [5, 13]). An exact formula which gives for any integer $n \geq 0$ the number of Sturmian palindromes of length n has been recently found in [13], where an interesting characterization of Sturmian palindromes has been also given.

A central word w is called *gold* if its minimal period π_w and the period $q = |w| + 2 - \pi_w$ are prime numbers. For instance, the central word $w = abaaba$ is a gold word. Gold words were introduced in [7], where some remarkable combinatorial properties have been studied. We limit ourselves to recall here that any element of St is a factor of a gold word.

An interesting recent result on standard words is related to the *Burrows-Wheeler transform*, which is a useful tool for data compression [6]. In [19] it has been proved that the Burrows-Wheeler transform of a word $w \in \mathcal{A}^*$ has the form $b^r a^s$ with $\gcd(r, s) = 1$ if and only if w is a conjugate of a standard word.

2.1 Generation

For any word w we denote by $w^{(-)}$ the shortest palindrome having the suffix w. The word $w^{(-)}$ is called the *palindromic left-closure of* w. For any language X, we set $X^{(-)} = \{w^{(-)} \mid w \in X\}$. The following lemma was proved in [12].

Lemma 1. *For any $w \in PER$, one has $(aw)^{(-)}, (bw)^{(-)} \in PER$. More precisely, if $w = w_1abw_2 = w_2baw_1$, then*

$$(aw)^{(-)} = w_2baw_1abw_2\,, \quad (bw)^{(-)} = w_1abw_2baw_1\,.$$

If $w = x^n$ with $\{x, y\} = \mathcal{A}$, then $(xw)^{(-)} = x^{n+1}$ and $(yw)^{(-)} = x^n y x^n$.

There exist two different, and in some respect dual, methods of constructing central words. The first method, based on the operation of palindromic left-closure, was introduced in [12]. By Lemma 1 we can define the map

$$\psi : \mathcal{A}^* \to PER,$$

as follows: $\psi(\varepsilon) = \varepsilon$ and for all $v \in \mathcal{A}^*$, $x \in \mathcal{A}$,

$$\psi(vx) = (x\psi(v))^{(-)} .$$

The map $\psi : \mathcal{A}^* \to PER$ is a bijection. Thus, for any $w \in PER$ there exists a unique word $v \in \mathcal{A}^*$ such that $w = \psi(v)$. The word v will be called the *ψ-generating word* of w. One has that for all $v, u \in \mathcal{A}^*$

$$\psi(vu) \in \mathcal{A}^*\psi(v) \cap \psi(v)\mathcal{A}^* .$$

Example 1. Let $w = abba$. One has

$$\begin{aligned} \psi(a) &= a , \\ \psi(ab) &= aba , \\ \psi(abb) &= ababa , \\ \psi(abba) &= ababaababa . \end{aligned}$$

In the sequel we shall denote by E the automorphism of $\mathcal{A}^*$ defined by $E(a) = b$, $E(b) = a$. From the definition one has that for any $x \in \mathcal{A}^*$, $\psi(E(x)) = E(\psi(x))$.

As usually, one can extend ψ to the subsets of $\mathcal{A}^*$ by setting, for all $X \subseteq \mathcal{A}^*$, $\psi(X) = \{\psi(x) \mid x \in X\}$. In particular, one has $\psi(a\mathcal{A}^*) = PER_a$ and $\psi(b\mathcal{A}^*) = PER_b$, where

$$PER_a = PER \cap a\mathcal{A}^* \quad \text{and} \quad PER_b = PER \cap b\mathcal{A}^* .$$

A second method to generate PER is obtained by introducing a further bijection $\varphi : \mathcal{A}^* \to PER$ based on some endomorphisms of $\mathcal{A}^*$. More precisely, let μ and λ be the morphisms defined as

$$\mu(a) = ab,\ \mu(b) = a, \quad \text{and} \quad \lambda(a) = a,\ \lambda(b) = ab .$$

The morphism μ is the so-called *Fibonacci morphism* and $\lambda = \mu E$.

Let φ be the map defined on $\mathcal{A}^*$ as follows: $\varphi(\varepsilon) = \varepsilon$, $\varphi(a) = a$, and for $w \in a\mathcal{A}^*$,

$$\varphi(wa) = \lambda(\varphi(w))a, \quad \varphi(wb) = \mu(\varphi(w))a .$$

Moreover, for $w \in b\mathcal{A}^*$ one sets $\varphi(w) = E(\varphi(E(w)))$. From the definition one has that for any $w \in \mathcal{A}^*$, $\varphi(E(w)) = E(\varphi(w))$. It has been proved [10, 11] that φ is a bijection of $\mathcal{A}^*$ onto PER. Thus, for any $w \in PER$ there exists a unique word $v \in \mathcal{A}^*$ such that $w = \varphi(v)$. The word v will be called the *φ-generating word* of w.

Example 2. Let $w = abba$. One has

$$\begin{aligned} \varphi(a) &= a , \\ \varphi(ab) &= aba , \\ \varphi(abb) &= abaaba , \\ \varphi(abba) &= aabaaabaa . \end{aligned}$$

The inner bijection $T = \psi^{-1}\varphi$ of $\mathcal{A}^*$ which maps the φ-generating word of a central word into the corresponding ψ-generating word, called *standard correspondence*, has been studied in [10].

2.2 Arithmetization

Let $\mathcal{I}$ be the set of all irreducible positive fractions. We consider the map $\theta : PER \to \mathcal{I}$, called the *ratio of periods*, defined as follows. By Proposition 1 any $w \in PER \setminus \{\varepsilon\}$ has the periods $p = \pi_w$ and $q = |w| + 2 - \pi_w$, where π_w is the minimal period of w. We set

$$\theta(w) = \frac{p}{q} \quad \text{if } w \in PER_a\,, \quad \theta(w) = \frac{q}{p} \quad \text{if } w \in PER_b\,.$$

Moreover,

$$\theta(\varepsilon) = \frac{1}{1}\,.$$

As proved in [12] the map θ is a bijection.

As is well known, for any $w \in PER$ the numbers $|w|_a + 1$ and $|w|_b + 1$ are coprime so that one can introduce the map $\eta : PER \to \mathcal{I}$ defined by

$$\eta(w) = \frac{|w|_b + 1}{|w|_a + 1}\,.$$

The map η is a bijection [2] that we call *rate*. We observe that the rate $\eta(w)$ is the 'slope' of the standard words wab and wba. As one easily checks, for any $w \in PER$ one has

$$\eta(E(w)) = \frac{1}{\eta(w)}\,.$$

Any word $w \in a\mathcal{A}^*$ can be uniquely represented as:

$$w = a^{\alpha_1} b^{\alpha_2} a^{\alpha_3} b^{\alpha_4} \cdots x^{\alpha_n}\,,$$

with $\alpha_i > 0$, $i = 1, \ldots, n$ and $x = a$ or $x = b$ according to the parity of n. As is well known (cf. [2]), $\eta(\psi(w))$ and $\theta(\psi(w))$ have a development in continued fractions given, respectively, by

$$\begin{aligned} \eta(\psi(w)) &= [0; \alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n + 1]\,, \\ \theta(\psi(w)) &= [0; \alpha_n, \alpha_{n-1}, \ldots, \alpha_2, \alpha_1 + 1]\,. \end{aligned} \tag{2}$$

Example 3. Let $w = ab^2a^2$. One has $\psi(w) = ababaababaababa$. Thus, $\eta(\psi(w)) = [0; 1, 2, 3] = 7/10$ and $\theta(\psi(w)) = [0; 2, 2, 2] = 5/12$.

Since θ and η are two bijections between PER and $\mathcal{I}$, one has that $\theta^{-1}\eta$ is an inner bijection of PER. One can easily derive from (2) that $\theta^{-1}\eta$ is involutory, i.e., $\theta^{-1}\eta = \eta^{-1}\theta$. In the sequel $\theta^{-1}\eta$ will be called the *natural involution* of PER. We notice that the natural involution is a length preserving map. Indeed,

if $\theta(w) = \eta(w') = h/k$, $w, w' \in PER$, then from the definition it follows easily that $|w| = |w'| = h + k - 2$.

The fixpoints of the natural involution of PER have been called *harmonic central words* or, simply, *harmonic words* [7]. The following theorem, which synthesizes some results proved in [7], gives some characterizations of harmonic words. We recall that a word $v \in \mathcal{A}^*$ is called *sesquipalindrome* if $v = E(v^\sim) = (E(v))^\sim$, where $^\sim$ denotes the *reversal* operation. We say that a rational number $\alpha > 1$ has a *symmetric development in continued fractions* if $\alpha = [q_0; q_1, \ldots, q_n]$ with $q_i = q_{n-i}$ for $i = 0, \ldots, n$.

Theorem 1. *Let w be a central word having minimal period π_w. The following conditions are equivalent:*

- *w is harmonic,*
- *the generating word $\psi^{-1}(w)$ is a palindrome or a sesquipalindrome,*
- *$1 + 1/\theta(w)$ (or $1 + 1/\eta(w)$) has a symmetric development in continued fractions,*
- *$\pi_w^2 \equiv \pm 1 \pmod{|w| + 2}$.*

We recall that the Fibonacci word f and $E(f)$ are the only infinite standard Sturmian words having all palindromic prefixes harmonic. The class $Harm$ of harmonic words is of great interest since one can prove that any element of St is a factor of a harmonic word, i.e.,

$$St = \mathrm{Fact}(Harm)\,.$$

Example 4. Let w be the central word $aabaabaa$. One has $\theta(w) = \eta(w) = [0; 2, 3] = 3/7$. The generating word $\psi^{-1}(w)$ of w is the sesquipalindrome a^2b^2. The rational number $1 + 1/\theta(w)$ has the symmetric development in continued fractions $[3; 3]$. Finally, $\pi_w^2 = 9 \equiv -1 \pmod{10}$.

2.3 Some Inner Bijections of $\mathcal{A}^*$

We introduce three maps T_1, T_2, and T_3 from $\mathcal{A}^*$ to $\mathcal{A}^*$ which can be defined in terms of suitable operations on the words of $\mathcal{A}^*$. These maps are bijections of $\mathcal{A}^*$ and length preserving. Moreover, they are related to the natural inner bijection of PER and to the generation maps ψ and φ.

The map T_1 is defined as follows: $T_1(\varepsilon) = \varepsilon$ and if $w = a_1 a_2 \ldots a_k$, $a_i \in \mathcal{A}$, $1 \le i \le k$, then $T_1(w) = b_1 b_2 \ldots b_k$ where $b_1 = a_1$ and for $1 < i \le k$,

$$b_i = \begin{cases} b_{i-1} & \text{if } a_i = a_1\,, \\ E(b_{i-1}) & \text{if } a_i = E(a_1)\,. \end{cases}$$

For instance, one has $T_1(aabbba) = aababb$. The map T_2 is defined for any $w \in \mathcal{A}^*$ by

$$T_2(w) = \begin{cases} E(w^\sim) & \text{if } w \in a\mathcal{A}^*b \cup b\mathcal{A}^*a\,, \\ w^\sim & \text{otherwise.} \end{cases}$$

Therefore, $T_2(aabbba) = abbbaa$ and $T_2(aabbb) = aaabb$. Finally, the map T_3 is defined as $T_3(\varepsilon) = \varepsilon$ and for any $x \in \mathcal{A}$ and $w \in \mathcal{A}^*$ as

$$T_3(xw) = xw^{\sim} .$$

Thus, for instance, $T_3(abbbaa) = aaabbb$.

From the definition one has that the maps T_1, T_2, and T_3 are length preserving. Moreover, T_2 and T_3 are involutory whereas T_1 is not. For instance, $T_1^2(aabbba) = aabbab$. The following theorem summarizes some results obtained in [9].

Theorem 2. *The following holds:*

- $T_1 = \psi^{-1}\theta^{-1}\eta\varphi$,
- $T_2 = \psi^{-1}\theta^{-1}\eta\psi$,
- $T_3 = \varphi^{-1}\theta^{-1}\eta\varphi$.

Moreover, $T = T_2T_1 = \psi^{-1}\varphi$ is the standard correspondence.

The preceding theorem shows, in particular, that the maps T_1, T_2, T_3, and T are inner bijections of $\mathcal{A}^*$.

One can naturally introduce the bijections Fa, Ga, Ra, and Sa of $\mathcal{A}^*$ in $\mathcal{I}$ defined as:

$$\mathrm{Fa} = \theta\psi,\ \mathrm{Ga} = \eta\varphi,\ \mathrm{Ra} = \theta\varphi,\ \text{and}\ \mathrm{Sa} = \eta\psi\,.$$

The maps Fa and Sa are called, respectively, the *Farey map* and the *Stern-Brocot map* [2, 10]. From the preceding theorem one derives:

$$T_1 = \mathrm{Fa}^{-1}\,\mathrm{Ga} = \mathrm{Sa}^{-1}\,\mathrm{Ra}\,,\ T_2 = \mathrm{Sa}^{-1}\,\mathrm{Fa}\,,\ T_3 = \mathrm{Ra}^{-1}\,\mathrm{Ga}\,,$$

and

$$T = \mathrm{Sa}^{-1}\,\mathrm{Ga} = \mathrm{Fa}^{-1}\,\mathrm{Ra}\ .$$

3 The Tree of Irreducible Fractions

We introduce in $\mathcal{I}$ the binary relation $\Rightarrow$ defined as follows: for $p/q, r/s \in \mathcal{I}$, one sets

$$\frac{p}{q} \Rightarrow \frac{r}{s} \quad \text{if} \quad p \le q\,,\ r \in \{p, q\}\,,\ s = p + q \quad \text{or} \quad p \ge q\,,\ s \in \{p, q\}\,,\ r = p + q\,.$$

One easily verifies that the graph of this relation is a complete binary tree with root $1/1$.

We denote by $\overset{*}{\Rightarrow}$ the reflexive and transitive closure of $\Rightarrow$. For instance, one has $1/2 \Rightarrow 2/3 \Rightarrow 2/5 \Rightarrow 5/7$, so that $1/2 \overset{*}{\Rightarrow} 5/7$. Moreover, for any $n \ge 0$ we denote by $\overset{n}{\Rightarrow}$ the nth power of the relation $\Rightarrow$. Thus, $1/2 \overset{3}{\Rightarrow} 5/7$.

The two following lemmas [9] give some insight in the structure of the derivation relation $\Rightarrow$.

Lemma 2. *Let $p/q \in \mathcal{I}$ with $p < q$. There exists $r/s \in \mathcal{I}$ such that $p/q \Rightarrow r/s$ and $r/s > p/q$ if and only if $p/q < g - 1$, where g is the golden ratio $g = (1 + \sqrt{5})/2$.*

Lemma 3. *Let $p/q \in \mathcal{I}$. If*

$$\frac{1}{1} \overset{m}{\Rightarrow} \frac{p}{q}, \quad m \geq 0,$$

then $m + 1 \leq \max\{p, q\} \leq F_{m+2}$, where $(F_n)_{n\geq 1}$ is the Fibonacci series. These bounds are tight.

The following proposition [9] relates the generation of central words by the maps ψ and φ with the derivation relation $\Rightarrow$.

Proposition 2. *Let w and w' be central words and u and v be the ψ and φ generating words of w. Then*

- $\theta(w) \Rightarrow \theta(w')$ *if and only if* $w' \in \psi(u\mathcal{A}) = (\mathcal{A}w)^{(-)}$,
- $\eta(w) \Rightarrow \eta(w')$ *if and only if* $w' \in \varphi(v\mathcal{A})$.

Some results of [8, 9, 12] are summarized in the following

Proposition 3. *Let $x, x' \in \mathcal{A}^*$. The following conditions are equivalent:*

1. *x is a prefix of x',*
2. $\mathrm{Fa}(x) \overset{*}{\Rightarrow} \mathrm{Fa}(x')$,
3. $\mathrm{Ga}(x) \overset{*}{\Rightarrow} \mathrm{Ga}(x')$,
4. *$\psi(x)$ is a prefix of $\psi(x')$.*

We say that a subset $\mathcal{H}$ of $\mathcal{I}$ is *independent* if for any pair of fractions $p/q, r/s \in \mathcal{H}$ such that $p/q \overset{*}{\Rightarrow} r/s$ one has $p/q = r/s$. A subset $\mathcal{H}$ of $\mathcal{I}$ is *full* if for any fraction $p/q \in \mathcal{I}$ there exists a fraction $r/s \in \mathcal{H}$ such that $p/q \overset{*}{\Rightarrow} r/s$ or $r/s \overset{*}{\Rightarrow} p/q$.

A set $X \subseteq \mathcal{A}^*$ is called *prefix* if no word of X is a proper prefix of another word of X. As is well known a prefix set $X \neq \{\varepsilon\}$ is a code [3]. From Proposition 3 one can easily prove the following:

Theorem 3. *Let $X \subseteq \mathcal{A}^*$. The following conditions are equivalent:*

1. *X is a prefix set,*
2. $\mathrm{Fa}(X)$ *is an independent set,*
3. $\mathrm{Ga}(X)$ *is an independent set.*

Theorem 4. *Let $X \subseteq \mathcal{A}^*$ be a prefix code. The following conditions are equivalent:*

1. *X is a maximal prefix code,*
2. $\mathrm{Fa}(X)$ *is a full set,*
3. $\mathrm{Ga}(X)$ *is a full set.*

4 Central Codes

Our concern in this section will be with central languages which are codes, i.e., bases of free submonoids of $\mathcal{A}^*$ [3]. These codes, which are in a two-letter alphabet, are called *Sturmian central codes* or, simply, *central codes*.

For instance, the sets $X_1 = \{a, b\}$, $X_2 = \{b, aa, aba\}$, $X_3 = \{aa, aabaa, babbab\}$, and $X_4 = \{b^2\} \cup (ab)^*a$ are central codes.

Central codes have been studied in [8]. We report here only some main results.

A central code is *maximal* if it is not properly included in another central code. By using a classical argument based on the Zorn property, which is satisfied by the family of central codes, one easily derives that any central code is included in a maximal central code. For instance, one can prove that the set $X = PER \setminus D$, where

$$D = \bigcup_{i \geq 0} ((ab)^i a)^* \cup ((ba)^i b)^* \,, \tag{3}$$

is a maximal central code.

We say that a set X is *PER-complete* if $PER \subseteq \operatorname{Fact} X^*$. This is equivalent to the statement that any word of St is a factor of a word of X^*.

Theorem 5. *A maximal central code X is PER-complete.*

Indeed, suppose by contradiction that there exists a word $u \in PER \setminus \operatorname{Fact} X^*$. We can assume, without loss of generality, that $u \in PER_a$ so that $\theta(u) = p/q$ with $p < q$. Since $p/q \stackrel{*}{\Rightarrow} (p+q)/(2p+q)$, by Proposition 3, u is a prefix of the word $w \in PER$ such that $\theta(w) = (p+q)/(2p+q)$. As $Y = X \cup \{w\}$ is not a code, there exist $h, k > 0$ and words $y_1, \ldots, y_h, y'_1, \ldots, y'_k \in Y$ such that $y_1 \neq y'_1$ and

$$y_1 \cdots y_h = y'_1 \cdots y'_k \,.$$

Since u is a prefix of w, one has that $w \notin \operatorname{Fact} X^*$. Moreover, as X is a code, one derives that w has to occur in both sides of the previous equation, i.e., there exist minimal positive integers i and j such that $w = y_i = y'_j$. With no loss of generality, we can assume $|y_1 \cdots y_{i-1}| < |y'_1 \cdots y'_{j-1}|$. Then one has

$$y'_1 \cdots y'_{j-1} = y_1 \cdots y_{i-1} s, \quad sw = wt, \quad y_{i+1} \cdots y_h = t y'_{j+1} \cdots y'_k \,, \tag{4}$$

for suitable non-empty words $s, t \in \mathcal{A}^*$. This equation shows that $|s|$ is a period of w, so that $|s| \geq \pi_w = p + q = |u| + 2$. Since u is a prefix of w, one derives from (4) that u is a prefix of s and, consequently, a factor of $y'_1 \cdots y'_{j-1} \in X^*$, which is a contradiction.

The following proposition whose proof is quite cumbersome since it requires several technical lemmas, shows that it does not exist a non-trivial finite *PER*-complete central code.

Proposition 4. *Let X be a finite PER-complete central code. Then $X = \mathcal{A}$.*

By the previous proposition and Theorem 5 it follows that any maximal central code $X \neq \mathcal{A}$ is infinite.

The following proposition shows that *PER*-completeness is a necessary but not sufficient condition in order that a central code is maximal.

Proposition 5. *There exists a PER-complete central code which is not a maximal central code.*

Indeed, one can prove that the set $Y = PER \setminus (D \cup \{aabaa\})$, where D is given by (3), is a *PER*-complete central code.

Proposition 6. *A central code $X \neq \mathcal{A}$ is not complete, i.e., $\mathcal{A}^* \not\subseteq \operatorname{Fact} X^*$.*

As any maximal code is complete [3], by the previous proposition, one derives that a central code $X \neq \mathcal{A}$ is not maximal as code.

4.1 Prefix Central Codes

A particular subclass of central codes is formed by *prefix central codes*, i.e., central codes which are prefix codes. Since the words of such codes are palindromes, one has that a prefix central code is also a suffix code and then a biprefix code.

For instance, the set $X = \{a, bab, bb\}$ is a prefix central code. The following theorem characterizes prefix central codes.

Theorem 6. *Let $Y \subseteq PER$. The following conditions are equivalent:*

1. *Y is prefix,*
2. *$Y = \psi(X)$, with X a prefix set,*
3. *$\theta(Y)$ is an independent set.*

Indeed, as a trivial consequence of Proposition 3 one has that $Y = \psi(X)$ is a prefix set if and only if X is a prefix set. By Theorem 3, this occurs if and only if $\operatorname{Fa}(X) = \theta(Y)$ is an independent set.

We call *pre-code* of a prefix central code Y the prefix code X such that $Y = \psi(X)$. For instance, the pre-code of $\{a, bab, bb\}$ is the prefix code $\{a, ba, bb\}$ and the pre-code of the prefix central code $\{aba, bb, babab, babbab\}$ is the prefix code $\{ab, bb, baa, bab\}$. The pre-code of the prefix central code $\{a^n ba^n \mid n \geq 0\}$ is the prefix code a^*b.

Theorem 6 shows that the property of being a prefix code is preserved by ψ and ψ^{-1}. On the contrary, the property of being a code is not, in general, preserved by ψ or ψ^{-1}, as shown by the following example.

Example 5. The set $X = \{ab, ba, abbb\}$ is a code whereas the set $\psi(X) = \{aba, bab, abababa\}$ is not a code. Conversely, the set $X = \{a, ab, bab\}$ is not a code whereas $\psi(X) = \{a, aba, babbab\}$ is a code.

A prefix central code is a *maximal prefix central code* if it is not properly included in another prefix central code. The following propositions have been proved in [8].

Proposition 7. *A prefix central code is a maximal prefix central code if and only if its pre-code is a maximal prefix code.*

Proposition 8. *A set $Y \subseteq PER$ is a maximal prefix central code if and only if $Y \neq \{\varepsilon\}$ and $\theta(Y)$ is an independent and full set.*

Indeed, by Theorem 6 and Proposition 7, Y is a maximal prefix central code if and only if $Y = \psi(X)$, with X a maximal prefix code. By Theorems 3 and 4, this occurs if and only if $Y \neq \{\varepsilon\}$ and $\mathrm{Fa}(X) = \theta(Y)$ is an independent and full set.

4.2 Farey Codes

For any positive integer n, we consider the set

$$\mathcal{F}_n = \left\{ \frac{p}{q} \in \mathcal{I} \,\middle|\, 1 \leq p \leq q \leq n \right\} .$$

As is well known, by ordering the elements of $\mathcal{F}_n$ in an increasing way one obtains the *Farey series of order n* (cf. [16]). Now, set for $n \geq 0$,

$$\mathcal{G}_n = \left\{ \frac{p}{q} \in \mathcal{F}_{n+1} \,\middle|\, p + q - 2 \geq n \right\} \quad \text{and} \quad \mathcal{G}'_n = \left\{ \frac{q}{p} \in \mathcal{I} \,\middle|\, \frac{p}{q} \in \mathcal{G}_n \right\} .$$

We introduce the sets of central words $\Delta_{n,a}$, $\Delta_{n,b}$, and Δ_n defined as

$$\Delta_{n,a} = \{ s \in PER \mid \theta(s) \in \mathcal{G}_n \} , \quad \Delta_{n,b} = \{ s \in PER \mid \theta(s) \in \mathcal{G}'_n \} ,$$

$$\Delta_n = \Delta_{n,a} \cup \Delta_{n,b} .$$

Since θ is a bijection, one has $\theta(\Delta_{n,a}) = \mathcal{G}_n$ and $\theta(\Delta_{n,b}) = \mathcal{G}'_n$.

One easily checks that $\Delta_0 = \Delta_{0,a} = \Delta_{0,b} = \{\varepsilon\}$ and for $n > 0$, $\Delta_{n,a} \subseteq PER_a$ and $\Delta_{n,b} \subseteq PER_b$. Moreover, the words of $\Delta_{n,b}$ are obtained from those of $\Delta_{n,a}$ by interchanging the letter a with b. The following noteworthy lemma holds [9]:

Lemma 4. *For all $n \geq 0$, the set $\mathcal{G}_n \cup \mathcal{G}'_n$ is independent and full.*

Since $\theta(\Delta_n) = \mathcal{G}_n \cup \mathcal{G}'_n$, from the preceding lemma and Proposition 8, one derives:

Proposition 9. *For all $n > 0$, Δ_n is a maximal prefix central code.*

For $n > 0$, the code Δ_n has been called the *Farey code of order n* [12].

Example 6. In the following table, we report the elements of $\mathcal{G}_6$ with the corresponding words of the prefix code $\Delta_{6,a}$ and their lengths.

1/7	$aaaaaa$	6
2/7	$abababa$	7
3/7	$aabaabaa$	8
4/7	$aabaaabaa$	9
3/5	$abaaba$	6
5/7	$ababaababa$	10
4/5	$aaabaaa$	7
5/6	$aaaabaaaa$	9
6/7	$aaaaabaaaaa$	11

Some interesting properties of Farey codes have been proved in [12] and [8]. We recall that for all $n > 0$,

$$\operatorname{Card} \Delta_n = \sum_{i=1}^{n+1} \phi(i) \, .$$

The length of the words of Δ_n is between n and $2n-1$. More precisely, for $0 \leq h \leq n-1$ one has

$$\operatorname{Card}(\Delta_n \cap \mathcal{A}^{n+h}) = \phi_{[h+1,n+1]}(n+h+2) \, ,$$

where $\phi_{[\alpha,\beta]}(n)$ denotes the number of integers in the interval $[\alpha, \beta]$ which are coprime with n.

The following proposition gives an equivalent definition for Farey codes.

Proposition 10. *For any $n \geq 0$ one has*

$$\Delta_n = \{w \in PER \mid n \leq |w| \leq n + \pi_w - 1\} \, .$$

Indeed, if $n = 0$ the result is trivial. Now suppose that $w \in PER_a$ and set $\theta(w) = p/q$, so that $p = \pi_w$ and $q = |w| - \pi_w + 2$. One has $w \in \Delta_{n,a}$ if and only if $p/q \in \mathcal{F}_{n+1}$ and $p + q - 2 = |w| \geq n$. Since $p/q \in \mathcal{F}_{n+1}$ if and only if $q = |w| - \pi_w + 2 \leq n + 1$, one derives that $w \in \Delta_{n,a}$ if and only if $n \leq |w| \leq n + \pi_w - 1$. If $w \in PER_b$, by a similar argument one obtains that $w \in \Delta_{n,b}$ if and only if $n \leq |w| \leq n + \pi_w - 1$.

For any $n \geq 0$ we set

$$U_n = PER \cap \mathcal{A}^n \, .$$

For $n > 0$, U_n is a *maximal uniform central code* [8]. For instance, one has

$$\begin{aligned} U_5 &= \{aaaaa, aabaa, ababa, babab, bbabb, bbbbb\} \, , \\ U_7 &= \{aaaaaaa, aaabaaa, abababa, bababab, bbbabbb, bbbbbbb\} \, . \end{aligned}$$

The following proposition which summarizes some results proved in [8] shows some important relations between Farey codes of consecutive orders and maximal uniform central codes.

Proposition 11. *For any $n \geq 0$ the following holds:*

- $\Delta_n \setminus \Delta_{n+1} = U_n$,
- $\Delta_{n+1} \setminus \Delta_n = (\mathcal{A}U_n)^{(-)} = \{w \in PER \mid |w| = n + \pi_w\}$,
- $\Delta_{n+1} = (\Delta_n \setminus U_n) \cup (\mathcal{A}U_n)^{(-)}$.

Since by the preceding proposition, $U_n \subseteq \Delta_n$, $n \geq 0$, one derives $PER = \bigcup_{n\geq 0} U_n \subseteq \bigcup_{n\geq 0} \Delta_n \subseteq PER$. Therefore, one has

$$PER = \bigcup_{n\geq 0} \Delta_n \, .$$

Proposition 12. *Let i and n be integers such that $0 \leq i \leq n$. Any element of Δ_i is a palindromic prefix of an element of Δ_n.*

Indeed, it is sufficient to prove that any $w \in \Delta_i$ is a palindromic prefix of an element of Δ_{i+1}. If $w \in \Delta_i \setminus U_i$, then by Proposition 11 one has $w \in \Delta_{i+1}$. If, on the contrary, $w \in U_i$, then w is a palindromic prefix of the word $(aw)^{(-)} \in \Delta_{i+1} \setminus \Delta_i$ by Proposition 11.

5 Farey Languages

In this section we introduce for each $n > 0$ two languages of central words L_n and M_n. The language L_n (resp., M_n) is called the *Farey* (resp., *dual Farey*) *language of order n*. The name is motivated by the fact that for any $n > 0$, L_n and M_n give faithful representations of the set of Farey's fractions of order n.

For any $n > 0$ the language L_n is defined as

$$L_n = \bigcup_{k=0}^{n-1} \Delta_{k,a} \, .$$

By Proposition 10 one has

$$L_n = \{w \in PER_a \cup \{\varepsilon\} \mid |w| \leq n + \pi_w - 2\} \, .$$

The set L_n is called the *Farey language* of order n. Indeed, the following proposition [9] shows that L_n gives a faithful representation of the set $\mathcal{F}_n$.

Proposition 13. *For any $n > 0$ one has $\theta(L_n) = \mathcal{F}_n$.*

It is remarkable that for any $n > 0$ the language L_n coincides with the set of palindromic prefixes of the Farey code $\Delta_{n-1,a}$. More precisely, denoted by PAL the set of palindromes over $\mathcal{A}$ and, for any $X \subseteq \mathcal{A}^*$, by $\operatorname{Pref} X$ the set of prefixes of the words of X, one has:

Proposition 14. *For all $n > 0$, $L_n = PAL \cap \operatorname{Pref} \Delta_{n-1,a}$.*

Indeed, by Proposition 12 one has $\Delta_{i,a} \subseteq PAL \cap \operatorname{Pref} \Delta_{n-1,a}$ for $i = 1, \ldots, n-1$ so that $L_n \subseteq PAL \cap \operatorname{Pref} \Delta_{n-1,a}$. Conversely, let $w \in PAL \cap \operatorname{Pref} \Delta_{n-1,a}$. Thus, there exists $v \in \Delta_{n-1,a}$ such that $v = w\lambda$, $\lambda \in \mathcal{A}^*$. Since any palindromic prefix

of a central word is central, one has $w \in PER$. We set $\theta(w) = r/s$ and $\theta(v) = p/q$. By Proposition 3 one has $r/s \stackrel{*}{\Rightarrow} p/q$. Thus, $s \leq q \leq n$. Therefore, $r/s \in \mathcal{F}_n$ so that by Proposition 13, $w \in L_n$.

The following proposition collects some results proved in [9]:

Proposition 15. *For all $n > 0$ one has*

- $L_{n+1} \setminus L_n = \{w \in PER_a \mid |w| = n + \pi_w - 1\} = \Delta_{n,a} \setminus \Delta_{n-1,a}$,
- $\mathrm{Card}(\Delta_{n,a} \setminus \Delta_{n-1,a}) = \mathrm{Card}(\mathcal{F}_{n+1} \setminus \mathcal{F}_n) = \phi(n+1)$,
- $L_n = \Delta_{n-1,a} \cup \bigcup_{i=0}^{n-2} U_{i,a}$,

where for any $i \geq 0$, $U_{i,a} = U_i \setminus b\mathcal{A}^*$.

5.1 Dual Farey's Languages

For $n \geq 0$ we set

$$H_{n,a} = \{s \in PER \mid \eta(s) \in \mathcal{G}_n\}\,, \quad H_{n,b} = \{s \in PER \mid \eta(s) \in \mathcal{G}'_n\}\,,$$

and

$$H_n = H_{n,a} \cup H_{n,b}.$$

One easily checks that $H_0 = H_{0,a} = H_{0,b} = \{\varepsilon\}$ and for $n > 0$, $H_{n,a} \subseteq PER_a$ and $H_{n,b} \subseteq PER_b$. Moreover, the words of $H_{n,b}$ are obtained from those of $H_{n,a}$ by interchanging the letter a with b. Since η is a bijection one has $\eta(H_{n,a}) = \mathcal{G}_n$ and $\eta(H_{n,b}) = \mathcal{G}'_n$.

Proposition 16. *For any $n > 0$ and $x \in \mathcal{A}$ one has*

$$H_{n,x} = \{w \in PER_x \mid |w|_x \leq n \leq |w|\}\,.$$

Indeed, let $w \in PER_a$. Since $\eta(w) = (|w|_b + 1)/(|w|_a + 1)$, one has $\eta(w) \in \mathcal{G}_n$ if and only if $|w|_a \leq n$ and $|w| \geq n$. The proof in the case $w \in PER_b$ is similar.

Example 7. In the following table, we report the elements of $\mathcal{G}_6$ with the corresponding words of the set $H_{6,a}$ and their lengths.

1/7	*aaaaaa*	6
2/7	*aaabaaa*	7
3/7	*aabaabaa*	8
4/7	*abaabaaba*	9
3/5	*abaaba*	6
5/7	*ababaababa*	10
4/5	*abababa*	7
5/6	*ababababa*	9
6/7	*abababababa*	11

We remark that differently from Δ_n, the set H_n in general is not a code. For instance, $H_{6,a}$ is not a code.

The following proposition, which is for some aspects dual of Proposition 11, shows some remarkable relations existing between H sets and uniform central codes [9].

Proposition 17. *For any $n \geq 0$ and $x \in \mathcal{A}$ one has*

- $H_{n,x} \setminus H_{n+1,x} = U_{n,x}$,
- $H_{n+1,x} \setminus H_{n,x} = \{w \in PER_x \mid |w|_x = n+1\}$,
- $H_{n+1,a} \setminus H_{n,a} = (\lambda(U_{n,a}) \cup \mu(U_{n,a}))a = \mu(U_n)a$,
- $H_{n+1,a} = (H_{n,a} \setminus U_{n,a}) \cup (\lambda(U_{n,a}) \cup \mu(U_{n,a}))a$.

By the preceding proposition one derives $U_n \subseteq H_n$, $n \geq 0$, so that $PER = \bigcup_{n\geq 0} U_n \subseteq \bigcup_{n\geq 0} H_n \subseteq PER$. Hence,

$$PER = \bigcup_{n\geq 0} H_n \, .$$

For any $n > 0$ we introduce the language

$$M_n = \bigcup_{k=0}^{n-1} H_{k,a} \, .$$

By Proposition 16 one easily derives that

$$M_n = \{w \in PER_a \cup \{\varepsilon\} \mid |w|_a \leq n-1\} \, . \tag{5}$$

The set M_n is called the *dual Farey language* of order n. In fact, also M_n gives a faithful representation of the set $\mathcal{F}_n$, as shown by the following [9]:

Proposition 18. *For any $n > 0$ one has $\eta(M_n) = \mathcal{F}_n$.*

One can easily verify that $L_n = M_n$ for $n = 1, 2, 3$. For $n \geq 4$ one has $L_n \neq M_n$. Indeed, one can prove that for any $n > 0$ and $(n+1)/2 < k \leq n-1$ the word $w = a^{k-1}ba^{k-1}$ belongs to $L_n \setminus M_n$.

Proposition 19. *For all $n > 0$, $PAL \cap \operatorname{Pref} H_{n-1,a} \subseteq M_n$.*

Indeed, if $w \in PAL \cap \operatorname{Pref} H_{n-1,a}$, there exists $v \in H_{n-1,a}$ such that $v = w\lambda$, $\lambda \in \mathcal{A}^*$. Since any palindromic prefix of a central word is central, one has $w \in PER$. By (5) one has $|w|_a \leq |v|_a \leq n-1$, so that $w \in M_n$.

We observe that the inclusion in the previous proposition is in general strict. For instance, one can verify that $w = aaabaaa \in H_{6,a} \subseteq M_9$. However, w is not a palindromic prefix of $H_{8,a}$.

The following proposition, which is the dual of Proposition 15, synthesizes some results proved in [9]:

Proposition 20. *For all $n > 0$ one has*

- $M_{n+1} \setminus M_n = \{w \in PER_a \mid |w|_a = n+1\} = H_{n,a} \setminus H_{n-1,a}$,
- $\operatorname{Card}(H_{n,a} \setminus H_{n-1,a}) = \operatorname{Card}(\mathcal{F}_{n+1} \setminus \mathcal{F}_n) = \phi(n+1)$,
- $M_n = H_{n-1,a} \cup \bigcup_{i=0}^{n-2} U_{i,a}$.

6 Farey Pre-codes

For any $n > 0$, the pre-codes of $\Delta_{n,a}$, $\Delta_{n,b}$, and Δ_n will be respectively denoted by $P_{n,a}$, $P_{n,b}$, and P_n. The prefix code $P_n = P_{n,a} \cup P_{n,b}$ will be called the *Farey pre-code of order* n.

Example 8. In the following table we report the elements of $\Delta_{6,a}$, the corresponding elements of the pre-code $P_{6,a}$, and their lengths.

$aaaaaa$	6	$aaaaaa$	6
$abababa$	7	$abbb$	4
$aabaabaa$	8	$aabb$	4
$aabaaabaa$	9	$aaba$	4
$abaaba$	6	aba	3
$ababaababa$	10	$abba$	4
$aaabaaa$	7	$aaab$	4
$aaaabaaaa$	9	$aaaab$	5
$aaaaabaaaaa$	11	$aaaaab$	6

As a consequence of Propositions 9 and 7 one has

Proposition 21. *For all $n > 0$, the Farey pre-code of order n is a maximal prefix code.*

In view of Proposition 11 one easily derives:

Proposition 22. *For all $n > 0$ one has*

$$P_{n+1} = (P_n \setminus \Gamma_n) \cup \Gamma_n \mathcal{A}\,,$$

where $\Gamma_n = \psi^{-1}(U_n)$.

The following proposition [9], whose proof is based on Lemma 3, gives the maximal and minimal value of the lengths of the words of the Farey pre-code of order n.

Proposition 23. *Let $n > 0$. For all $x \in P_n$ one has*

$$\min\{k \mid F_{k+3} \geq n+2\} \leq |x| \leq n\,.$$

These bounds are tight.

We consider also the sets $Q_n = \varphi^{-1}(H_n)$, $R_n = \psi^{-1}(H_n)$, and $S_n = \varphi^{-1}(\Delta_n)$. For all $n > 0$ the following relations hold [9]:

$$Q_n = T_1^{-1}(P_n)\,, \quad R_n = T_2(P_n)\,, \quad \text{and} \quad S_n = T_3 T_1^{-1}(P_n)\,, \tag{6}$$

where T_1, T_2, and T_3 are the inner bijections of $\mathcal{A}^*$ introduced in Sect. 2. Therefore, the sets Q_n, R_n, and S_n can be determined from P_n by the maps T_1, T_2, and T_3. Since these maps are length preserving, these sets and P_n have the same word-length distribution. Moreover, for any $n > 0$, Q_n is a maximal prefix code, R_n is a maximal suffix code, and $S_{n,x} = S_n \cap x\mathcal{A}^*$, $x \in \mathcal{A}$, is a suffix code.

In the following table we report the elements of $P_{6,a}$ and the corresponding elements of the codes $Q_{6,a}$, $R_{6,a}$, and $S_{6,a}$ computed by using (6)

$P_{6,a}$	$Q_{6,a}$	$R_{6,a}$	$S_{6,a}$
$aaaaaa$	$aaaaaa$	$aaaaaa$	$aaaaaa$
$abbb$	$abaa$	$aaab$	$aaab$
$aabb$	$aaba$	$aabb$	$aaba$
$aaba$	$aabb$	$abaa$	$abba$
aba	abb	aba	abb
$abba$	$abab$	$abba$	$abab$
$aaab$	$aaab$	$abbb$	$abaa$
$aaaab$	$aaaab$	$abbbb$	$abaaa$
$aaaaab$	$aaaaab$	$abbbbb$	$abaaaa$

7 Farey Languages and Riemann's Hypothesis

As we have seen in Sect. 2 one can introduce two bijections of the set of central words onto the set of positive irreducible fractions, namely the ratio of periods θ and the rate η.

These bijections allow one to define in PER two natural total order relations by setting for any $w_1, w_2 \in PER$,

$$w_1 \leq w_2 \quad \text{if} \quad \theta(w_1) \leq \theta(w_2)$$

and

$$w_1 \preceq w_2 \quad \text{if} \quad \eta(w_1) \leq \eta(w_2)\,.$$

For any $L \subseteq PER$ and $x \in PER$ one can consider the sets

$$L * x = \{w \in L \mid w \leq x\} \quad \text{and} \quad L \circ x = \{w \in L \mid w \preceq x\}\,.$$

We define the θ-*order* of x relative to L the quantity

$$\mathrm{ord}^{\theta}_{L}\, x = \mathrm{Card}(L * x)\,.$$

In a similar way, the η-*order* of x relative to L is defined as

$$\mathrm{ord}^{\eta}_{L}\, x = \mathrm{Card}(L \circ x)\,.$$

In the sequel we shall mainly refer to the θ-order and denote $\mathrm{ord}^{\theta}_{L}\, x$ simply by $\mathrm{ord}_L\, x$.

If L is finite, then $\mathrm{ord}_L\, x < \infty$. If L is infinite, the order of an element $x \in PER$ may be infinite or finite. For instance, the order of any central word with respect to PER is infinite. On the contrary, if $L = \{w \in PER \mid \theta(w) = (n-1)/n,\ n > 1\}$, then the order of any central word with respect to L is finite. If $L \subseteq M \subseteq PER$, then for all $x \in PER$ one has $\mathrm{ord}_L\, x \leq \mathrm{ord}_M\, x$.

From the definition one derives that if $L = \bigcup_{i=1}^{n} K_i$, where K_i, $i = 1, \ldots, n$, are pairwise disjoint sets, then for all $x \in PER$,

$$\mathrm{ord}_L\, x = \sum_{i=1}^{n} \mathrm{ord}_{K_i}\, x\,. \tag{7}$$

If $L \subseteq PER$ has finite cardinality k, then

$$\sum_{x \in L} \mathrm{ord}_L\, x = \frac{k(k+1)}{2}\,.$$

Given a finite subset L of PER and a word $x \in PER$ one can consider the quantity

$$\delta_L(x) = \theta(x) - \frac{\mathrm{ord}_L\, x}{\mathrm{Card}\, L}\,.$$

If $x \in PER_a$ the *companion* of x is the word $x' \in PER_a$ such that $\theta(x') = 1 - \theta(x)$. A set $L \subseteq PER_a \cup \{\varepsilon\}$ is *closed by companion* if $\varepsilon \in L$ and for any $x \in L \setminus \{\varepsilon\}$ the companion of x is in L. The following lemma holds [9]:

Lemma 5. *Let $L \subseteq PER_a \cup \{\varepsilon\}$ be a central language of finite cardinality and closed by companion. For any $x \in L \setminus \{\varepsilon\}$ one has*

$$\delta_L(x) = -\delta_L(x')\,,$$

where x' is the companion of x. Moreover,

$$\sum_{w \in L} \theta(w) = \sum_{w \in L} \frac{\mathrm{ord}_L\, w}{k} = \frac{k+1}{2}\,,$$

where $k = \mathrm{Card}\, L$.

Given a finite set $L \subseteq PER_a \cup \{\varepsilon\}$ one can introduce the quantity

$$\sum_{x \in L} |\delta_L(x)|$$

which gives an evaluation of the regularity of the distribution of θ in L.

In the previous section we have considered for any $n \geq 0$ the Farey language L_n which gives a faithful representation of $\mathcal{F}_n$. The order of a central word x with respect to L_n will be briefly denoted by $\mathrm{ord}_n\, x$.

We notice that for all $x \in PER_b \cup \{\varepsilon\}$, $\mathrm{ord}_n\, x = \mathrm{Card}\, L_n = \mathrm{Card}\, \mathcal{F}_n = \Phi(n)$, where $\Phi(n) = \sum_{i=1}^{n} \phi(i)$.

In the sequel we set $K_0 = \{\varepsilon\}$ and for $n > 0$, $K_n = \Delta_{n,a} \setminus \Delta_{n-1,a}$. The following proposition holds [9]:

Proposition 24. *For any $x \in PER_a \cup \{\varepsilon\}$,*

$$\mathrm{ord}_{K_n}\, x = \phi_{[1,(n+1)\theta(x)]}(n+1)\,.$$

By Proposition 11 one derives that the sets K_n, $n \geq 0$, are pairwise disjoint. For any $n \geq 0$ we can decompose L_n as

$$L_n = \bigcup_{i=0}^{n-1} \Delta_{i,a} = \bigcup_{i=0}^{n-1} K_i\,.$$

By (7) and Proposition 24 one derives that for any $x \in PER_a \cup \{\varepsilon\}$,

$$\operatorname{ord}_n x = \sum_{i=0}^{n-1} \operatorname{ord}_{K_i} x = \sum_{i=1}^{n} \phi_{[1,i\theta(x)]}(i)\,.$$

Example 9. For $n = 5$ one has $K_0 = \{\varepsilon\}$, $K_1 = \{a\}$, $K_2 = \{a^2, aba\}$, $K_3 = \{a^3, a^2ba^2\}$, $K_4 = \{a^4, ababa, aba^2ba, a^3ba^3\}$, and

$$\theta(K_0) = \left\{\frac{1}{1}\right\}\,,\ \theta(K_1) = \left\{\frac{1}{2}\right\}\,,\ \theta(K_2) = \left\{\frac{1}{3}, \frac{2}{3}\right\}\,,$$

$$\theta(K_3) = \left\{\frac{1}{4}, \frac{3}{4}\right\}\,,\ \theta(K_4) = \left\{\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}\right\}\,.$$

For instance, let $w = a^2ba^2$. One has $\theta(w) = 3/4$, $\operatorname{ord}_{K_0} w = 0$, $\operatorname{ord}_{K_1} w = 1$, $\operatorname{ord}_{K_2} w = 2$, $\operatorname{ord}_{K_3} w = 2$, and $\operatorname{ord}_{K_4} w = 3$, so that $\operatorname{ord}_5 w = 8$.

Since for any $n \geq 0$ the Farey language L_n is closed by companion and $\operatorname{Card} L_n = \Phi(n)$, by Lemma 5 one has

$$\sum_{w \in L_n} \theta(w) = \sum_{w \in L_n} \frac{\operatorname{ord}_n w}{\Phi(n)} = \frac{\Phi(n)+1}{2}\,.$$

Since θ is a bijection, for any $x \in PER$ one has $\operatorname{ord}_n x = \operatorname{Card}(L_n * x) = \operatorname{Card} \theta(L_n * x)$. Since $\theta(L_n) = \mathcal{F}_n$ one derives

$$\operatorname{ord}_n x = \operatorname{Card}\left\{\frac{p}{q} \in \mathcal{F}_n \;\middle|\; \frac{p}{q} \leq \theta(x)\right\}\,. \tag{8}$$

If $x \in L_n$, then $\theta(x) \in \mathcal{F}_n$ so that $\operatorname{ord}_n x$ gives the 'place' occupied by $\theta(x)$ in the Farey series of order n.

Let f_ν, $\nu = 1, \ldots, \Phi(n)$, be the Farey series of order n. A theorem of Franel and Landau (cf. [15]) shows that the famous Riemann hypothesis on Zeta function is equivalent to the statement that

$$\sum_{\nu=1}^{\Phi(n)} \left| f_\nu - \frac{\nu}{\Phi(n)} \right| = o(n^{(1/2)+\varepsilon})$$

for all $\varepsilon > 0$ as $n \to \infty$. Since $\operatorname{Card} L_n = \Phi(n)$, by (8) one derives

$$\sum_{x \in L_n} |\delta_{L_n}(x)| = \sum_{\nu=1}^{\Phi(n)} \left| f_\nu - \frac{\nu}{\Phi(n)} \right|\,,$$

so that Riemann's hypothesis is equivalent to the statement that

$$\sum_{x \in L_n} |\delta_{L_n}(x)| = o(n^{(1/2)+\varepsilon})$$

for all $\varepsilon > 0$ as $n \to \infty$.

If one considers the η-order instead of the θ-order, one can obtain results analogous to the previous ones. In particular, for any finite subset L of PER and any $x \in PER$ one defines the quantity

$$\delta_L^\eta(x) = \eta(x) - \frac{\operatorname{ord}_L^\eta x}{\operatorname{Card} L}\,.$$

One easily verifies that Riemann's hypothesis is equivalent to the statement

$$\sum_{x \in M_n} |\delta_{M_n}^\eta(x)| = o(n^{(1/2)+\varepsilon})$$

for all $\varepsilon > 0$ as $n \to \infty$.

It is noteworthy that Riemann's hypothesis can be restated in terms of a combinatorial property of the Farey languages L_n and M_n. We recall that a similar result was obtained by Mignosi [20] by considering suitable languages of finite Sturmian words.

References

1. Allouche, J.-P., Shallit, J.: Automatic Sequences. Cambridge University Press (Cambridge, U.K., 2003)
2. Berstel, J., de Luca, A.: Sturmian words, Lyndon words and trees. Theoretical Computer Science **178** (1997) 171–203
3. Berstel, J., Perrin, D.: Theory of Codes. Academic Press (New York, 1985)
4. Berstel, J., Séébold, P.: Sturmian words. In M. Lothaire: Algebraic Combinatorics on Words. Cambridge University Press (Cambridge, U.K., 2002), pp. 45–110
5. Borel, J. P., Reutenauer, C.: Palindromic factors of billiard words. Theoretical Computer Science (to appear)
6. Burrows, M., Wheeler, D. J.: A block sorting data compression algorithm. Technical report, DIGITAL System Research Center, 1994
7. Carpi, A., de Luca, A.: Harmonic and gold Sturmian words. European Journal of Combinatorics **25** (2004) 685–705
8. Carpi, A., de Luca, A.: Codes of central Sturmian words. Theoretical Computer Science (to appear)
9. Carpi, A., de Luca, A.: Farey codes and languages. Manuscript, 2005
10. de Luca, A.: Combinatorics of standard Sturmian words. In Mycielski, J., Rozenberg, G., Salomaa, A. (eds.): Structures in Logic and Computer Science. Lecture Notes in Computer Science, vol. 1261, Springer (Berlin, 1997) pp. 249–267
11. de Luca, A.: Standard Sturmian morphisms. Theoretical Computer Science **178** (1997) 205–224
12. de Luca, A.: Sturmian words: structure, combinatorics, and their arithmetics. Theoretical Computer Science **183** (1997) 45–82
13. de Luca, A., De Luca, A.: Palindromes in Sturmian words. In De Felice, C., Restivo, A. (eds.): Proc.s DLT '05. Lecture Notes in Computer Science, Springer (Berlin, 2005)
14. de Luca, A., Mignosi, F.: On some combinatorial properties of Sturmian words. Theoretical Computer Science **136** (1994) 361–385
15. Edwards, H. M.: Riemann's Zeta Function. Academic Press (New York, 1974)

16. Hardy, G. H., Wright, E. M.: An Introduction to the Theory of Numbers. Clarendon, Oxford University Press (Oxford, U.K., 1968)
17. Ilie, L., Plandowski, W.: Two-variable word equations. Theoretical Informatics and Applications **34** (2000) 467–501
18. Lothaire, M.: Combinatorics on Words. Addison-Wesley (Reading, MA, 1983); 2nd edition: Cambridge University Press (Cambridge, U.K., 1997)
19. Mantaci, S., Restivo, A., Sciortino, M.: Burrows-Wheeler transform and Sturmian words. Information Processing Letters **86** (2003) 241–246
20. Mignosi, F.: On the number of factors of Sturmian words. Theoretical Computer Science **82** (1991) 71–84

Reversible Cellular Automata

Jarkko Kari[1,2,⋆]

[1] Department of Mathematics, FIN-20014 University of Turku, Finland
[2] Department of Computer Science, University of Iowa, Iowa CIty, IA 52242, USA
jjkari@cs.uiowa.edu

Abstract. Reversible cellular automata (RCA) are models of massively parallel computation that preserve information. This paper is a short survey of research on reversible cellular automata over the past fourty plus years. We discuss the classic results by Hedlund, Moore and Myhill that relate injectivity, surjectivity and reversibility with each other. Then we review algorithmic questions and some results on computational universality. Finally we talk about local reversibility vs. global reversibility.

1 Introduction

Cellular automata (CA) are discrete dynamical systems and models of massively parallel computation that share many properties of the physical world. They consist of very large numbers of simple elements that operate in parallel and interact only locally. The update rules are invariant with respect to location and time. Most interestingly, by choosing proper update rules one can program into the system fundamental properties of microscopic physics such as reversibility and conservation laws without sacrificing computational universality.

Reversible cellular automata (RCA), also known as invertible cellular automata, are cellular automata that fully preserve information. They are capable to mimic reversible physical phenomena more closely than any other computational model proposed so far. Lattice gases are a perfect example of this. But more importantly, because RCA obey fundamental laws of physics, their hardware implementation may – at least in principle – be more energy efficient than the implementations of irreversible systems used today. According to the Landauer's principle erasure of one bit of information at absolute temperature T always dissipates at least $kT \ln 2$ Joule of energy, where k is the Bolzmann's constant. This will eventually become prohibitive for further miniaturization and increasingly dense packing of irreversible gates such as AND and OR. Reversible computation has been proposed as an alternative where no bits need to be erased, hence avoiding the $kT \ln 2$ lower bound on the energy dissipation.

RCA are among the most closely studied types of cellular automata: There has been a steady stream of results since the early 60's until today. This survey article presents these results in a compact form. We give references to the original sources where complete proofs can be found.

⋆ Research supported by the Academy of Finland grant 54102

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 57–68, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Definitions and Classic Results

A cellular automaton is an infinite lattice of finite state machines, called *cells*. The cells are located at the integer lattice points of the d-dimensional Euclidean space. We refer to the cells by their coordinates, i.e. cells are addressed by the elements of $\mathbb{Z}^d$. Let S be the finite state set. A *configuration* of the CA is a function

$$c : \mathbb{Z}^d \longrightarrow S$$

where $c(\boldsymbol{x})$ is the current state of the cell $\boldsymbol{x}$. The set $S^{\mathbb{Z}^d}$ of all configurations is denoted by $\mathcal{C}(d, S)$, or briefly $\mathcal{C}$ when d and S are known from the context. Constant functions are called *homogeneous* configurations.

The cells change their states synchronously at discrete time steps. The next state of each cell depends on the current states of the neighboring cells according to an update rule. All cells use the same rule, and the rule is applied to all cells at the same time. The neighboring cells may be the nearest cells surrounding the cell, but more general neighborhoods can be specified by giving the relative offsets of the neighbors. Let $N = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ be a vector of n distinct elements of $\mathbb{Z}^d$. Then the *neighbors* of a cell at location $\boldsymbol{x} \in \mathbb{Z}^d$ are the n cells at locations

$$\boldsymbol{x} + \boldsymbol{x}_i, \text{ for } i = 1, 2, \ldots, n.$$

The *local rule* is a function $f : S^n \longrightarrow S$ where n is the size of the neighborhood. State $f(a_1, a_2, \ldots, a_n)$ is the new state of a cell whose n neighbors were at states $a_1, a_2, \ldots, a_n$ one time step before. This update rule then determines the global dynamics of the CA: Configuration c becomes in one time step the configuration e where, for all $\boldsymbol{x} \in \mathbb{Z}^d$,

$$e(\boldsymbol{x}) = f(c(\boldsymbol{x} + \boldsymbol{x}_1), c(\boldsymbol{x} + \boldsymbol{x}_2), \ldots, c(\boldsymbol{x} + \boldsymbol{x}_n)).$$

We say that $e = G(c)$, and call $G : \mathcal{C} \longrightarrow \mathcal{C}$ the *global transition function* of the CA.

We have seen that cellular automata are dynamical systems that are homogeneous and discrete in both time and space, and they are updated locally. A d-dimensional CA is specified by a triple (S, N, f) where S is the state set, $N \in (S^{\mathbb{Z}^d})^n$ is the neighborhood vector, and $f : S^n \longrightarrow S$ is the local update rule. We usually identify a cellular automaton with its global transition function G, and talk about cellular automaton function G, or simply cellular automaton G. In algorithmic questions G is however always specified using the three finite items S, N and f.

Let us call a CA *injective* (*surjective*) if and only if its global function G is one-to-one (onto, respectively). The CA is *bijective* if G is both onto and one-to-one. A cellular automaton with global function G is called *reversible* or *invertible* if there is cellular automaton function F such that $G \circ F = F \circ G = id$ where id is the identity function. Then F and G are the *inverse automata* of each other. One can effectively decide whether two given cellular automata are inverses of each other: One can namely effectively form the compositions of the two automata and test whether the compositions are equal to the identity function.

Example 1. Let $d = 1$, $N = (0, 1)$ and $S = \{0, 1\}$. Cells are laid on a line and indexed by $\mathbb{Z}$. Each cell i has two neighbors: i itself and $i+1$. Let the local rule f be the modulo two addition. This CA is not injective as the two homomogeneous configurations c_0 and c_1 consisting entirely of 0's and 1's, respectively, are both mapped into the same configuration c_0. The CA is, however, surjective. We call this CA the XOR -automaton.

2.1 The Curtis-Hedlund-Lyndon Theorem

Let $\boldsymbol{y} \in \mathbb{Z}^d$ be arbitrary. The *translation* $\tau_{\boldsymbol{y}}$ by $\boldsymbol{y}$ is the CA with the single element neighborhood $(-\boldsymbol{y})$ and the identity local rule. Every translation is a composition of *shifts*, i.e. translations by a single cell in one of the coordinate directions. Homogeneousity in space means that every CA function G commutes with every translation τ, that is, $G \circ \tau = \tau \circ G$.

Topology turns out to be a useful tool in cellular automata theory. Let us endow the set $\mathcal{C}$ with the *Cantor topology*, i.e. the topology generated by the basis of cylinder sets

$$\mathrm{Cyl}(c, M) = \{e \in \mathcal{C} \mid e(\boldsymbol{x}) = c(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in M\}$$

for $c \in \mathcal{C}$ and finite $M \subseteq \mathbb{Z}^d$. Cylinder sets are both open and closed in the Cantor topology. This topology is induced by a metric, and most importantly, the topology is compact. Relevance to cellular automata stems from the fact that all cellular automata functions are continuous in this topology. In fact, a classic result from 1969 known as the Curtis-Hedlund-Lyndon theorem states also the converse:

Theorem 1 ([6]). *A function $G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ is the global transition function of a cellular automaton if and only if*

(i) G is continuous, and
(ii) G commutes with shifts.

If G is a reversible CA function then $G : \mathcal{C} \longrightarrow \mathcal{C}$ is by definition bijective. Conversely, suppose that G is a bijective CA function. Then G has an inverse function that clearly commutes with the shifts. The inverse function is also continuous because the space $\mathcal{C}$ is compact. So by the Curtis-Hedlund-Lyndon theorem the inverse is a CA function. We have proved:

Corollary 1 ([6]). *A cellular automaton G is reversible if and only if it is bijective.*

The point of the corollary is that in bijective CA each cell can determine its previous state by looking at the current states in some bounded neighborhood around them. In symbolic dynamics literature it is customary to call reversible (one-dimensional) cellular automata *automorphisms* of the shift dynamical system.

2.2 The Garden of Eden-Theorem

Sometimes one state $q \in S$ is specified as a *quiescent state*. It should be *stable*, which means that $f(q, q, \ldots, q) = q$. The quiescent configuration Q is the configuration where all cells are quiescent: $Q(\boldsymbol{x}) = q$ for all $\boldsymbol{x} \in \mathbb{Z}^d$. A configuration $c \in S^{\mathbb{Z}^d}$ is called *finite* if only a finite number of cells are non-quiescent, i.e. the support

$$\{\boldsymbol{x} \in S^{\mathbb{Z}} \mid c(\boldsymbol{x}) \neq q\}$$

is finite. Let us denote by $\mathcal{C}_F(d, S)$, or briefly $\mathcal{C}_F$, the subset of $S^{\mathbb{Z}^d}$ that contains only the finite configurations. Because of the stability of q, finite configurations remain finite in the evolution of the CA, so the restriction G_F of G on the finite configurations is a function $\mathcal{C}_F \longrightarrow \mathcal{C}_F$.

A *periodic configuration*, or more precisely, a spatially periodic configuration is a configuration that is invariant under d linearly independent translations. This is equivalent to the existence of d positive integers $t_1, t_2, \ldots, t_d$ such that for every $\boldsymbol{x} \in \mathbb{Z}^d$ and every $i = 1, 2, \ldots, d$ we have

$$c(\boldsymbol{x}) = c(\boldsymbol{x} + t_i \boldsymbol{e}_i),$$

where $\boldsymbol{e}_i$ is the i'th unit coordinate vector. Let us denote by $\mathcal{C}_P(d, S)$, or briefly $\mathcal{C}_P$, the set of periodic configurations. Cellular automata are homogeneous in space and consequently they preserve periodicity of configurations. The restriction G_P of G on the periodic configurations is hence a function $\mathcal{C}_P \longrightarrow \mathcal{C}_P$.

Finite configurations and periodic configurations are used in simulations of cellular automata on computers. Periodic configurations are often referred to as the periodic boundary conditions on a finite cellular array. For example, in the case $d = 2$ this is equivalent to running the CA on a torus that is obtained by "gluing" together the opposite sides of a rectangle. One should, however, keep in mind that the behavior of a CA can be quite different on finite, periodic and general configurations, so experiments done with periodic boundary conditions may be misleading. One should also keep in mind that G_F may be bijective even if the CA is not reversible. This has led to some confusion on the terminology, especially in some online resources where only finite configurations are considered.

Example 2. Let $d = 1$, $N = (0, 1)$ and $S = \{00, 01, 10, 11\}$. The first bit of each state is a control symbol that does not change. If the control symbol of a cell is 0 then the cell is inactive and does not change its state. If the control symbol is 1 then the cell is active and applies the XOR rule of Example 1 on the second bit. In other words,

$$f(ab, cd) = \begin{cases} ab, & \text{if } a = 0, \\ ax, & \text{if } a = 1, \text{ where } x = b + d \pmod 2. \end{cases}$$

State 00 is the quiescent state. This CONTROLLED-XOR automaton is easily seen bijective on finite configurations. It is not injective on unrestricted configurations as two configurations, all of whose cells are active, have the same image if their second bits are complements of each other.

If G is not surjective then there exist *Garden-of-Eden* configurations, that is, configurations without a pre-image. A trivial property of any finite set is that a function from the set into itself is injective if and only if it is surjective. In cellular automata the same is true only in one-direction: an injective CA is always surjective, but the converse is not true. However, finite configurations behave more like finite sets: G_F is injective if and only if G is surjective. This result, known as the Garden-of-Eden theorem, is one of the oldest results in the theory of cellular automata. The two directions of the theorem are due to E.F. Moore in 1962 [16] and J. Myhill in 1963 [19]:

Theorem 2 ([16, 19]). *G_F is injective if and only if G is surjective.*

It is trivial that the injectivity of the full function G implies the injectivity of its restrictions G_F and G_P, so we immediately get the following corollary:

Corollary 2. *Injective CA are also surjective. Hence injectivity, bijectivity and reversibility are equivalent.*

Notice that sets $\mathcal{C}_F$ and $\mathcal{C}_P$ are dense in $\mathcal{C}$. Then it follows from the continuity of G and the compactness of $\mathcal{C}$ that the surjectivity of G_F or G_P implies the surjectivity of G. The next theorem summarizes these and other known relations. The proofs can be found, for example, in [4]. The results are summarized in Figures 1 and 2.

Theorem 3. *The following implications are true in every dimension d:*

- *If G is injective then G_P and G_F are injective,*
- *If G_P or G_F is surjective then G is surjective,*
- *If G_P is injective then G_P is surjective,*
- *If G is injective then G_F is surjective.*

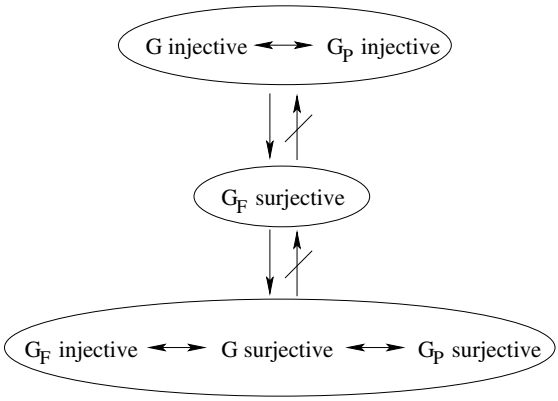


Fig. 1. Implications between injectivity and surjectivity properties in one-dimensional CA

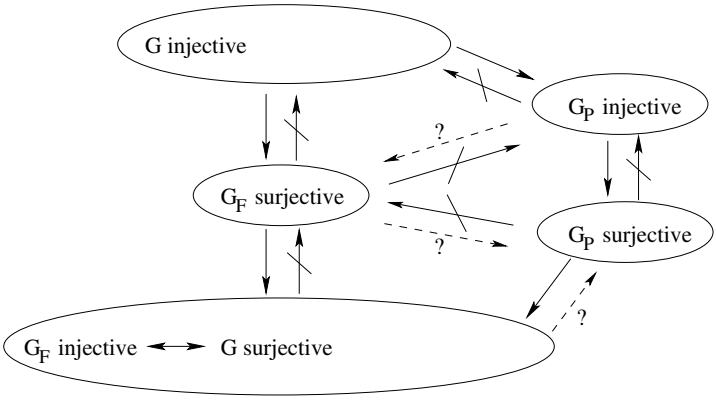


Fig. 2. Implications between injectivity and surjectivity properties in two- and higher dimensional CA

In addition, the following implications are true for one-dimensional CA:

- *If G_P is injective then G is injective,*
- *If G is surjective then G_P is surjective.*

The two non-implications in Figure 1 are proved by the automata XOR and CONTROLLED-XOR of Examples 1 and 2, respectively. Notice that in two- and higher dimensional cases there are three implications whose status is unknown.

3 The Inverse Automaton

There are natural algorithmic questions that arise. How does one determine if a given CA is reversible ? If the CA is reversible how does one find its inverse ? How large can the neighborhood of the inverse automaton be ? In the one-dimensional case S.Amoroso and Y.Patt gave already in 1972 algorithms to test whether a given CA is reversible or surjective [1]:

Theorem 4 ([1]). *There exist algorithms to determine if a given one-dimensional cellular automaton is injective or surjective.*

Elegant decision algorithms based on de Bruijn graphs were later designed by K. Sutner [22]. In higher dimensional spaces the questions are however much harder. It was shown in [10] that the questions are then undecidable (see also [8]):

Theorem 5 ([10]). *There are no algorithms to determine if a given two-dimensional cellular automaton is injective or surjective.*

Even though Corollary 1 guarantees that in bijective CA every cell can determine its previous state based on the current states in some fixed neighborhood of the cell, Theorem 5 implies that this neighborhood may be very large. There can be no computable upper bound on the radius of the neighborhood, because

otherwise we could test all candidate inverses one-by-one. Let us say that a CA has radius r if all elements of its neighborhood vector N belong to the set $\{-r, \ldots, r\}^d$.

Corollary 3. *For every computable function $h : \mathbb{N} \longrightarrow \mathbb{N}$ there exists a two-dimensional CA of radius one whose inverse function is not computed by any CA of radius $h(s)$, where s is the number of states.*

In contrast, in the one-dimensional space the inverse automaton can only have a relatively small neighborhood. Consider a one-dimensional CA with s states and the two element neighborhood vector $(0, 1)$. It is fairly easy to establish a qudratic $O(s^2)$ upper bound on the neighborhood radius of the inverse CA. Recently we improved this bound and showed that the inverse neighborhood consists of at most $s-1$ consecutive cells [3]. This bound is tight since examples are known that require precisely $s-1$ elements in the inverse neighborhood [9].

Theorem 6 ([3, 9]). *A one-dimensional RCA with s states and neighborhood $(0, 1)$ has an inverse whose neighborhood consists of at most $s-1$ consecutive positions. This bound is tight.*

If the neighborhood in the forward direction is contained in m rather than two consecutive positions then a similar argument shows that the inverse neighborhood is contained in $s^{m-1} - 1$ consecutive positions [3]

4 Computational Universality

Another natural line of investigation is to study the computational power of RCA. It is clear that any Turing machine can be simulated by a one-dimensional CA. In 1977 T.Toffoli demonstrated how any d-dimensional CA can be simulated by a $d+1$-dimensional RCA [23]. As a corollary we obtain a two-dimensional universal RCA.

The result was improved in [17] where it was shown that reversible Turing machines can be simulated by one-dimensional reversible CA. This establishes the existence of universal one-dimensional reversible CA, since reversible Turing machines can be computationally universal [2]. A direct reversible one-dimensional simulation of an arbitrary one-dimensional CA (with finite configurations) was presented in [18].

Theorem 7 ([17]). *One-dimensional reversible cellular automata exist that are computationally universal.*

An elegant universal two-dimensional RCA was presented by N.Margolus in 1984 [15]. This *Billiard Ball RCA* introduced the concept of space partitioning as a tool to enforce reversibility. In this technique the update is done in two steps (see Figure 3): In the first step the plane is partitioned into 2×2 blocks along, say, odd coordinates. A permutation π_1 of S^4 is applied inside each block, where S is the state set. In the second step the partitioning is shifted so that

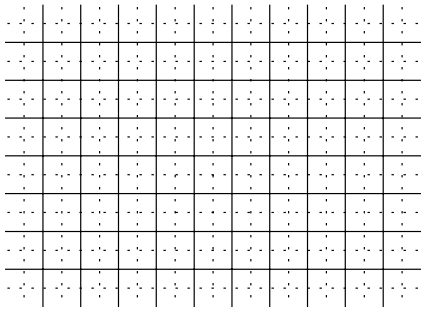
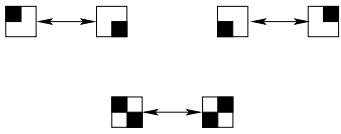


Fig. 3. The Margolus neighborhood. Odd updates use the solid partitioning, even updates the dashed partitioning

the plane is partitioned along even coordinates, and another permutation π_2 of S^4 is applied. This technique became known as the Margolus neighborhood.

In the billiard ball computer we have $\pi_1 = \pi_2$. The state set is binary, and we denote the states as white and black. The permutations π_1 and π_2 only make the following exchanges:



All other 2×2 blocks are unchanged. Because of the alternating partitioning, a single black state propagates on the plane in one of the four diagonal directions that depends on the parity of its location. With such a simple update rule it is possible to simulate the motion and collisions of billiard balls of positive diameter. Walls from which the balls bounce can also be created, and all these constructs can be combined to perform arbitrary computation [15].

A CA that uses the Margolus neighborhood is trivially reversible – the inverse automaton applies the inverse permutations. Also other physical constraints can be easily programmed into a CA that uses such a neighborhood. For example, the number of black cells is automatically preserved if the permutations are such that they conserve black states.

Strictly speaking Margolus neighborhood is not a CA neighborhood in the sense of our definitions. The local update rule is different for even and odd cells. But if we consider "supercells" that consist of 2×2 blocks then all cells are identical. Also time steps become identical if we combine the even and odd clock cycles into one update step. In this strict sense the billiard ball computer by Margolus has $2^4 = 16$ states and neighborhood radius one.

5 Local Invertibility

In view of our motivation in physics and computation – namely implementing reversible massively parallel computers using some reversible physical systems

– reversibility of the global transition function G does not seem like the most relevant property to study. Even though $G : \mathcal{C} \longrightarrow \mathcal{C}$ is one-to-one on infinite configurations, the local rule f that produces G is not one-to-one. Function f is then useless if we want to implement G using finite, reversible logic gates. Rather, we prefer local rules that, unlike f, are themselves reversible, such as the permutations in the Margolus neighborhood. In [25] the natural question was asked whether all reversible cellular automata can be implemented using reversible local update rules. This question was answered affirmatively in [11] for one- and two-dimensional cellular automata. In higher dimensional spaces the question remains open, although in [5] it was shown how to simulate any RCA of any dimension by a locally reversible RCA.

For simplicity we only consider the one-dimensional case here. Let G be a one-dimensional RCA whose neighborhood is within $\{-l, \ldots, r\}$ and suppose the inverse G^{-1} has neighborhood within $\{-l', \ldots, r'\}$. Take an arbitrary configuration c, write $G(c)$ under c, and extract the state contents of the cells within the stair-like shape shown in Figure 4. We obtain some element of S^m where S is the state set and m is the number of cells within the shape. We repeat this for all configurations $c \in S^{\mathbb{Z}}$. Let $L \subseteq S^m$ be the set of "left stairs" we can extract this way. Note that typically $L \neq S^m$ because the states we extract from c restrict $G(c)$ and vice versa. For example, if $G = id$ is the identity function then the extractions from c and $G(c)$ in the middle part of the stair must be identical.

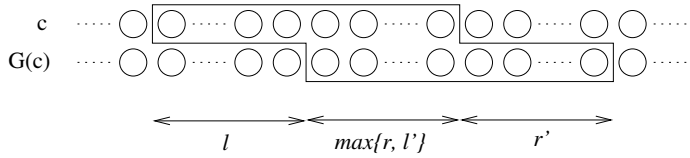


Fig. 4. Extraction of left stairs

The dimensions of the stair shape are chosen large enough to prevent the influence of the states on either side of the stair to the other side. We repeat the analogous process to obtain the set R of "right stairs", as illustrated in Figure 5. Let us denote by λ and ρ the total widths of the left and the right stairs, respectively, that is

$$\lambda = l + \max\{r, l'\} + r', \text{ and}$$
$$\rho = l' + \max\{r', l\} + r.$$

We are interested in the cardinalities of sets L and R. For example, if $G = id$ then $|L| = s^\lambda$ and $|R| = s^\rho$ where $s = |S|$ is the number of states. If G is the shift to the left then $|L| = s^{\lambda+1}$ and $|R| = s^{\rho-1}$. It was shown in [11] that always $|L| \cdot |R| = s^{\lambda+\rho}$. Let us associate with G the number

$$\varphi(G) = \frac{|L|}{s^\lambda}.$$

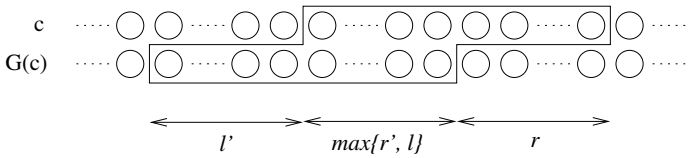


Fig. 5. Extraction of right stairs

Function φ has the interesting property that it is a group homomorphism from the group of RCA functions with state set S into the multiplicative group of rational numbers [11], i.e. we have that

$$\varphi(G \circ F) = \varphi(G)\varphi(F)$$

for any RCA G and F over the same state set.

Consider a one-dimensional RCA G that uses a generalized Margolus neighborhood: the update consists of t steps: $G = G_1 \circ G_2 \circ \ldots \circ G_t$. Each step G_i is a *block permutation rule*: it partitions the configuration space into blocks of equal length k_i and applies some permutation π_i of S^{k_i} on each block. One can easily see that $\varphi(G_i) = 1$, that is, each G_i belongs to the kernel $ker(\varphi)$ of the homomorphism φ. It follows then that also $G \in ker(\varphi)$. This means that functions that are outside the kernel – for example the shifts – cannot be expressed as a composition of block permutation rules.

Conversely, if G is a one-dimensional RCA function that satisfies $\varphi(G) = 1$ then it is a composition $G = G_1 \circ G_2$ where both G_1 and G_2 are block permutations that use blocks of length $k_1 = k_2 = \rho + \lambda$. The first block permutation G_1 computes in each block of configuration c two adjacent left/right stairs, and stores these stairs at the left/right ends of the block, see Figure 6. The second rule G_2 takes right/left stairs and decodes them into blocks of $G(c)$. We have the following result:

Theorem 8 ([11]). *A one-dimensional RCA G is a composition of block permutation rules if and only if $\varphi(G) = 1$. In this case two block permutations with block sizes $\lambda + \rho$ suffice.*

As block permutations are locally reversible, we conclude that all one-dimensional RCA can be implemented using finite reversible logic gates. (Those RCA

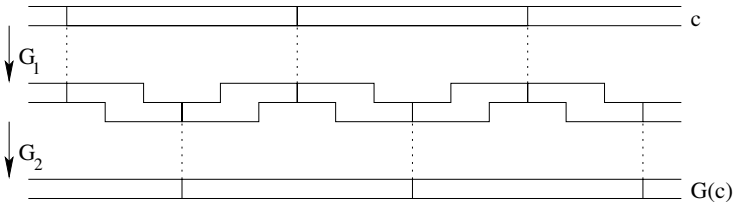


Fig. 6. Factoring RCA G into a sequence of two block permutations G_1 and G_2

that are not in the kernel of φ can be combined with a shift or a shift-like function before applying Theorem 8.) A more complex analysis shows that an analogous result applies to two-dimensional RCA [11]. In this case two group homomorphisms are found, corresponding to the two dimensions of the space, and those RCA that are in the kernels of both homomorphisms can be factored into a sequence of two-dimensional block permutation rules. Three block permutations are sufficient in this case [12]. Factoring in higher dimensional spaces is still open, but what is known is that in the d-dimensional space any RCA that is a composition of some block permutations is a composition of at most $d+1$ block permutations:

Theorem 9 ([12]). *If $G_1, G_2, \ldots, G_t$ are d-dimensional block permutation rules then there are $d+1$ block permutation rules $B_1, B_2, \ldots, B_{d+1}$ such that*

$$G_1 \circ G_2 \circ \ldots \circ G_t = B_1 \circ B_2 \circ \ldots \circ B_{d+1}.$$

6 Conclusion

We have briefly surveyed some developments in the theory of reversible cellular automata. RCA are interesting mathematical objects to study. In the future, they potentially play a role in the design of energy efficient massively parallel computers. Applications in data coding (e.g. cryptography and compression) are also possible. Interesting developments not covered in this survey include many results on reversible and surjective CA whose local update rule is additive or has some other algebraic structure [7, 13, 14, 21].

References

1. S. Amoroso and Y. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, Journal of Computer and System Sciences 6 (1972) 448–464.
2. C. Bennett, Logical reversibility of computation, IBM Journal of Research and Development 6 (1973) 525–532.
3. E. Czeizler and J. Kari, A tight linear bound on the neighborhood of inverse cellular automata, To appear in the Proceedings of ICALP 2005, Lecture Notes in Computer Science, Springer, 2005.
4. B. Durand, Global properties of 2D cellular automata, in: Cellular Automata and Complex Systems, E. Goles and S. Martinez (Eds.), Kluwer, 1998.
5. J. Durand-Lose, Representing reversible cellular automata with reversible block cellular automata, in: Discrete Models, Combinatorics, Computation and Geometry, R. Cori, J. Mazoyer, M. Morvan and R. Mosery (Eds.), 145–154, Springer, 2001.
6. G. Hedlund, Endomorphisms and automorphisms of shift dynamical systems, Math. Systems Theory 3 (1969) 320–375.
7. M. Ito, N. Osato and M. Nasu, Linear Cellular Automata over $\mathbb{Z}_m$, Journal of Computer and System Sciences 27 (1983) 125–140.

8. J. Kari, Reversibility of 2D cellular automata is undecidable, Physica D 45 (1990) 379–385.
9. J. Kari, On the Inverse Neighborhoods of Reversible Cellular Automata, in: Lindenmayer Systems, Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology, G. Rozenberg, A. Salomaa (Eds.), 477–495, Springer-Verlag, 1992.
10. J. Kari, Reversibility and surjectivity problems of cellular automata, Journal of Computer and System Sciences 48 (1994) 149–182.
11. J. Kari, Representation of reversible cellular automata with block permutations,Mathematical Systems Theory 29 (1996) 47–61.
12. J. Kari, On the circuit depth of structurally reversible cellular automata, Fundamenta Informatica 38 (1999) 93–107.
13. J. Kari, Linear Cellular Automata with Multiple State Variables, in: Proceedings of STACS'2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 1770, 110–121, Springer-Verlag, 2000.
14. G. Manzini and L. Margara, Invertible Linear Cellular Automata over $\mathbb{Z}_m$: Algorithmic and Dynamical Aspects, Journal of Computer and System Sciences 56 (1998) 60–67.
15. N. Margolus, Physics-like models of computation, Physica D 10 (1984) 81–95.
16. E.F. Moore, Machine Models of Self-reproduction, Proceedings of the Symposium in Applied Mathematics 14 (1962) 17–33.
17. K. Morita and M. Harao, Computation Universality of one.dimensional reversible (injective) cellular automata, IEICE Transactions E72 (1989) 758–762.
18. K. Morita, Reversible simulation of one-dimensional irreversible cellular automata, Theoretical Computer Science 148 (1995) 157–163.
19. J. Myhill, The Converse to Moore's Garden-of-Eden Theorem, Proceedings of the American Mathematical Society 14 (1963) 685–686.
20. D. Richardson, Tessellations with Local Transformations, Journal of Computer and System Sciences 6 (1972) 373–388.
21. T. Sato, Decidability of some problems of linear cellular automata over finite commutative rings, Information Processing Letters 46 (1993) 151–155.
22. K. Sutner, De Bruijn graphs and linear cellular automata, Complex Systems 5 (1991) 19–31.
23. T. Toffoli, Computation and construction universality of reversible cellular automata, Journal of Computer and System Sciences 15 (1977) 213–231.
24. T. Toffoli and N. Margolus, Cellular Automata Machines, MIT Press, 1987.
25. T. Toffoli and N. Margolus, Invertible cellular automata: a review, Physica D 45 (1990) 229–253.

Inexpressibility Results for Regular Languages in Nonregular Settings

Howard Straubing

Computer Science Department
Boston College
Chestnut Hill, Massachusetts USA 02476
straubin@cs.bc.edu

My ostensible purpose in this talk is to describe some new results (found in collaboration with Amitabha Roy) on expressibility of regular languages in certain generalizations of first-order logic. [10]. This provides me with a good excuse for describing some the work on the algebraic theory of regular languages in what one might call "nonregular settings".

The syntactic monoid and syntactic morphism of a regular language provide a highly effective tool for proving that a given regular language is not expressible or recognizable in certain compuational models, as long as the model is guaranteed to produce only regular languages. This includes finite automata, of course. but also formulas of propositional temporal logic, and first-order logic, provided one is careful to restrict the expressive power of such logics. (For example, by only allowing the order relation in first-order formulas.)

Things become much harder, and quite a bit more interesting, when we drop this kind of restriction on the model. The questions that arise are important (particularly in computational complexity), and most of them are unsolved. They all point to a rich theory that extends the reach of algebraic methods beyond the domain of finite automata

1 Uniformizing Nonuniform Automata with Ramsey's Theorem

Let's start with an especially trivial application of the syntactic monoid: Let $\Sigma = \{0, 1\}$, and consider the two languages

$$L_1 = \{w \in \Sigma^* : |w| \equiv 0 \pmod 2\}$$

and

$$L_2 = \{w \in \Sigma^* : |w|_1 \equiv 0 \pmod 2\}.$$

(We denote by $|w|_1$ the number of 1's that in the string w.)

These two languages have the same syntactic monoid, namely the group of two elements. It follows immediately that neither can be recognized by a finite automaton whose transition monoid is aperiodic (i.e., contains no nontrivial groups). Put another way, if $\phi : \Sigma^* \to M$ is a homomorphism onto a finite

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 69–77, 2005.
© Springer-Verlag Berlin Heidelberg 2005

aperiodic monoid, then there is no $X \subseteq M$ such that $\phi^{-1}(X) = L_1$, or $\phi^{-1}(X) = L_2$.

Suppose that instead of a homomorphism, we consider a map

$$\psi : \Sigma \times \mathbf{Z}^+ \to M,$$

and extend it to $\Psi : \Sigma^* \to M$ by mapping

$$w = \sigma_1 \cdots \sigma_n$$

to

$$\Psi(w) = \psi(\sigma_1, 1)\psi(\sigma_2, 2) \cdots \psi(\sigma_n, n).$$

You can think of this as a kind of "nonuniform automaton", in which the state transition induced by a letter depends on the position of the letter within the input string. This obviously increases the computational power of the model; the languages recognized need not even be recursively enumerable! Can we recognize L_1 with such a setup? Can we recognize L_2?

The answer to the first question is "yes": Let M be the aperiodic monoid $\{1, a, b\}$ with multiplication defined by $Ma = a$, $Mb = b$. Set $\psi(\sigma, i) = b$ whenever i is odd, and $\psi(\sigma, i) = a$ when i is even. Then $L_1 = \Psi^{-1}(a)$.

But L_2 cannot be recognized. Let us suppose that we have some aperiodic monoid M and map Ψ that does recognize L_2. We will show that in spite of the unruliness of Ψ, we can tame it so that it behaves like a homomorphism on a large set of inputs: Since M is aperiodic, it satisfies some identity of the form $x^n = x^{n+1}$ for some n. Let $0 \leq i < j$, and let us color the segment (i, j) by the pair

$$(\psi(0, i)\psi(0, i+1) \cdots \psi(0, j-1), \psi(1, i)\psi(0, i+1) \cdots \psi(0, j-1)).$$

Ramsey's Theorem guarantees the existence of a sequence

$$i_1 < i_2 < \cdots < i_{2n+2}$$

such that each (i_j, i_{j+1}) is has the same color (m_0, m_1). Consider a string $w_1 \in \Sigma^*$ of length $i_{2n+2} - 1$ that has 1's in positions $i_1, \ldots, i_{n+1}$, and 0's elsewhere. Then

$$\Psi(w_1) = \Psi(0^{i_1 - 1})m_1^{n+1}m_0^n.$$

Let us change the last 1 in w_1 to 0, giving a new string w_2. We now have

$$\Psi(w_2) = \Psi(0^{i_1 - 1})m_1^n m_0^{n+1} = \Psi(0^{i_1 - 1})m_1^{n+1}m_0^n = \Psi(w_1),$$

but the numbers of 1's in the two strings differ by 1. So Ψ cannot recognize L_2.

We get the same conclusion if we allow M to contain nontrivial groups of odd order. We get a similar conclusion if we replace L_2 by the set of strings in which the number of 1's is divisible by $q > 1$: This language cannot be recognized by a map $\psi \times \mathbf{Z}^+ \to M$ if every group in M has cardinality relatively prime to q.

2 Programs over Finite Monoids

The results of the last section are due to Barrington and Straubing [5]. The "nonuniform automata" we considered are special cases of *programs over finite monoids*. These are defined as follows: With each integer $n > 0$ we associate a sequence of *instructions*

$$(\psi_{1,n}, i_1), \ldots, (\psi_{r_n,n}, i_{r_n}),$$

where each $i_j \in \{1, \ldots, n\}$ and each $\psi_{j,n}$ is a map from Σ into M. The value of the program Ψ on

$$w = \sigma_1 \cdots \sigma_n \in \Sigma^n$$

is

$$\Psi(w) = \prod_{j=1}^{r_n} \psi_{j,n}(\sigma_{i_j}) \in M$$

In other words, the program scans the input word in some haphazard order, possibly revisiting the same input letter many times. At each input letter, the program emits an element of M, which depends on both the letter itself and the instruction. The product of these elements determines whether w is accepted or not. We call the function $n \mapsto r_n$ the *length* of the program.

The nonuniform automata of the preceding section are programs that make a single scan over their input strings. In fact, [5] establishes a stronger result:

Theorem 1. *Let $q > 0$, and let $L \subset \{0,1\}^*$ be the set of strings in which the number of 1's is divisible by q. Let M be a finite monoid in which every group has cardinality relatively prime to q. Then any program over M recognizing L has length $\Omega(n \log\log n)$.*

If we allow the program length to be polynomial in the input length, however, we get the following remarkable result, due to Barrington [1]

Theorem 2. *If M is a finite monoid that contains a nonsolvable group,then every regular language is recognized by some polynomial-length program over M.*

This implies, for example, that Theorem 1 cannot be extended to polynomial-length programs, since the set of strings in which the number of 1's is divisible by 7 is recognized by the alternating group A_5, whose order is 60. In fact, Barrington showed that every language in the circuit complexity class NC^1 is recognized by a polynomial-length program over A_5. (Any other nonsolvable group will do.) It is this connection with circuit complexity that motivates the interest in programs over monoids.

3 Programs, Logic and Circuits

Let's take a look again at Theorem 2 and ask what happens when M contains only solvable groups. Such monoids are called *solvable monoids.* If the program

into M is an ordinary homomorphism, then the language L recognized by the program is recognized by M in the ordinary sense; in particular, the syntactic monoid $M(L)$ of L is solvable. Straubing, Thérien and Thomas [13] give a characterization in generalized first-order logic of the regular languages recognized by finite solvable monoids: Consider the logic in which variables represent positions in a string over Σ, and which there are the following relation symbols: $x < y$, which is interpreted to mean that position x is to the left of positon y, and $Q_\sigma x$, where $\sigma \in \Sigma$, which is interpreted to mean that the letter in position x is σ. In addition to the usual boolean operations and ordinary quantification, we allow *modular* quantifiers $\exists^{r \bmod q}$, where $\exists^{r \bmod q} x\phi$ is interpreted to mean "the number of positions x satisfying ϕ is congruent to r modulo q.

A sentence in this logic accordingly defines a language over Σ. For instance, the formula $\phi(x)$ given by

$$\exists y(x < y \wedge \neg \exists z(x < z \wedge z < y) \wedge Q_\sigma x \wedge Q\sigma y)$$

says "position x and its successor both contain the letter σ", and thus the sentence

$$\exists^{0 \bmod 2} x\phi(x)$$

defines the set of strings in which $\sigma\sigma$ occurs an even number of times as a factor. In [13] it is proved that the family of languages defined in this way is precisely the family of regular languages over Σ recognized by finite monoids in which every group is solvable.

Suppose now that instead of a homomorphism, we have a polynomial-length program over M. On an input of length n, the program emits a sequence

$$m_1, m_2, \ldots, m_{n^k}$$

of elements of M. We can view this sequence as an element w' of M^*, where we treat M as a finite alphabet. We then have $w \in L$ if and only if $w' \in L'$, where L' is the set of strings in M^* that multiply to a value in X.

Of course, L' is a regular language recognized by M, so by the theorem just cited, L' is defined by a sentence ϕ' using both modular and ordinary quantifiers, and in which the only numerical predicate is $<$.

In [2] it is shown how we can rewrite this sentence to obtain a defining sentence ϕ for L The essential idea is that each position in the w' can be encoded by a k-tuple of positions in w, and thus each variable in ϕ' is replaced by a k-tuple of variables in ϕ. However, in constructing the sentence for L, we are obliged to introduce new numerical predicates that encode the nonuniform behavior of the k-program Ψ. This is no surprise: since k-programs over M can recognize uncomputable languages, so we would necessarily introduce uncomputable numerical predicates in the defining sentences.

Thus every language recognized by a polynomial-length program over a solvable monoid is definable by sentence with modular quantifiers (with no restriction on the kinds of numerical predicates introduced into the sentence).

Conversely, every language defined by such a sentence is recognized by a polynomial-length program over a finite solvable monoid. Thus we have:

Theorem 3. *$L \subseteq \Sigma^*$ is defined by a sentence using modular and ordinary quantifiers if and only if L is recognized by a polynomial-length program over a finite solvable monoid.*

Moreover, these are exactly the languages recognized by a special kind of boolean circuit: polynomial-size, constant-depth families of circuits with unbounded fan-in gates to compute AND, OR, and MOD_q, for a modulus $q > 1$ fixed throughout the family. In computational complexity theory, this class of languages is called ACC. The connection between ACC and programs over solvable monoids was discovered by Barrington and Thérien [6]. We will not discuss circuits further here, but instead concentrate on the representations of ACC in terms of programs and logic.

4 The Main Question

As we've seen, when we do not restrict the numerical predicates that occur in our quantified formulas, we obtain languages that have arbitrarily high computational complexity, in the sense that they may be uncomputable, but have tightly bounded computational complexity in quite a different sense, since they are recognized by small circuits. However, the true computational power of such circuits is an open question, since we do not even know how to show that ACC does not contain NP-complete languages.

But what if a language L defined by such a sentence is known to be regular? Let us first consider the case of a regular language defined by a first-order sentence, without the use of modular quantifiers. The example given in the first section shows that the syntactic monoid of such a language might contain a group. Indeed, the sentence

$$\forall x(\forall y(y \leq x) \rightarrow (x \bmod 2 = 0))$$

defines the set of strings of even length. Let us suppose, however, that the input alphabet Σ contains a *neutral letter* for L – that is, we suppose there exists $\sigma \in \Sigma$ such that σ is mapped to the identity element of the syntactic monoid of L. This rules out such examples as the set of strings of even length. We can then show

Theorem 4. *Let $L \subseteq \Sigma^*$ be a regular language such that Σ contains a neutral letter for L. If L is defined by a first-order sentence, then $M(L)$ is aperiodic.*

This theorem is due to Barrington, *et. al.* [2]. The proof, however, requires the solution of a difficult problem in circuit complexity. (The separation of AC^0 from NC^1, see Furst, Saxe and Sipser [7].)

We do not know how to show that there is any regular language not definable by a sentence with modular quantifiers. But it has long been conjectures that nonsolvability is necessary for the behavior observed in Theorem 2.

Conjecture 1. Let $L \subseteq \Sigma^*$ be a regular language defined by a sentence that inlcudes both modular and ordinary quantifiers. Suppose that Σ contains a neutral letter for L. Then every group in $M(L)$ is solvable.

This conjecture is equivalent to the assertion that the circuit complexity class ACC is properly contained in the class NC^1, a long-unsolved problem in computational complexity. (The neutral letter hypothesis is not, strictly speaking, necessary, since if the conjecture is true in the above form it remains true without the hypothesis.)

There is, as well, a purely modular form of the conjecture:

Conjecture 2. Let $L \subseteq \Sigma^*$ be a regular language defined by a sentence that inlcudes only modular quantifiers. Suppose that Σ contains a neutral letter for L. Then $M(L)$ is a solvable group..

How might we approach such a question? The arguments in Section 1 show that for languages defined by single-scan programs, we can use Ramsey's Theorem to smooth out the non-uniformity in the program and make it look like a homomorphism. Can we apply the same ideas to logical formulas? Perhaps we can prove the conjectures for special classes of formulas.

If the only numerical predicate occurring in our formulas is the order relation, then Conjectures 1 and 2 hold because of the results, already cited, in [13]. But this is what we have been calling a *regular setting:* Formulas such as these cannot define nonregular languages. The simplest example of a nonregular setting is provided by formulas in which, in addition to the ordering relation, there are *monadic* numerical predicates (predicates with a single argument). The following is from Straubing [11]:

Theorem 5. *Conjectures 1 and 2 hold for formulas in which every numerical predicate is the order relation or a monadic relation.*

This holds because such formulas give rise to single-scan programs with equivalent behavior. More general formulas give rise to programs whose length is n^k for $k > 1$, and uniformizing these is considerably more difficult.

5 Presburger Arithmetic and Active-Domain Sentences

Presburger Arithmetic is the first-order theory of the natural numbers with addition. In other words, a formula in this theory is just a first-order formula (typically with free variables) with the single ternary predicate $x = y + z$. A formula such as

$$\exists y(x = y + y)$$

expresses the property "x is even". Of course, we need quantification to express such a property. Suppose though, that we add some new symbols to our logic: Specifically, we allow constants 0 and 1, and atomic formulas $t \equiv_m t'$, where t and t' are terms, interpreted to mean t and t' are congruent modulo m. We can express "x is even" by the quantifier free formula

$$x \equiv_2 0.$$

Presburger [9] showed that every formula in the original logic is equivalent to a quantifier-free formula, provided we add the constants 0 and 1, extend the set

of relations to include ordering and all the formuals $t \equiv_m t'$. (Since these can all be expressed in the original logic, we do not change the properties definable in Presburger arithmetic by adjoining them.)

We can use the apparatus of Presburger arithmetic to define languages: Let us add a single unary predicate symbol π. We interpret $\pi(x)$ to mean "the bit in position x is 1". (In other words, $\pi(x)$ has the same meaning as $Q_1 x$, but it is important in this context that there is no $Q_0 x$.) A sentence in this logic can be interpreted in a string of bits and accordingly defines a language in $\{0,1\}^*$. We also allow interpretation in infinite strings in $\{0,1\}^*0^\omega$, that is, infinite bit strings in which there are finitely many 1's.

We say that a sentence in this logic is an *active-domain* sentence if every quantifier occurs in the form

$$\exists x(\pi(x) \wedge \psi).$$

In other words, we only allow quantification over positions that contain a 1.

The same techniques used to prove quantifier elimination in Presburger arithmetic can be used to show that every sentence in this logic is equivalent to an active-domain sentence, provided we extend the signature to include 0,1, ordering, and congruence modulo k. A proof is outlined in Libkin [8].

When we have modular quantifiers available as well, active-domain quantification means that every modular quantifer occurs in the form

$$\exists^{r \bmod q} x(\pi(x) \wedge \psi).$$

Very recently, A. Roy and I [10] showed that the elimination of non-acitive-domain quantifiers can be extended to formulas that contain modular as well as ordinary quantifers:

Theorem 6. *Let $L \subseteq \{0,1\}^*$ or $L \subseteq \{0,1\}^*0^\omega$. Suppose L is defined by a sentence ϕ, with both modular and ordinary quantifiers and with relation symbols $+$ and π. Then L is defined by an active-domain sentence ϕ' with modular and ordinary quantifers, constant symbols 0 and 1, and relation symbols $+$, $<$ and $\equiv_m$. Moreover, the moduli of the modular quantifiers in ϕ' all occur in ϕ.*

6 Ramsey's Theorem Again

We can talk about active-domain sentences with reference to arbitrary alphabets Σ of input letters, not just the binary alphabet $\{0,1\}$. We simply designate some $\tau \in \Sigma$ to be the "inactive letter", and require all quantifiers to occur in the context

$$\mathcal{Q}x(\bigvee_{\sigma \neq \tau} Q_\sigma x \wedge \phi),$$

where $\mathcal{Q}$ is either an ordinary existential quantifier, or a modular quantifier.

Theorem 7. *Conjectures 1 and 2 are true for languages defined by active-domain sentences in which the neutral letter is inactive. Moreover, every such language is itself regular and its syntactic monoid is a solvable monoid (in Conjecture 1) or a solvable group (in Conjecture 2).*

There is no restriction in the foregoing theorem on the numerical predicates that can occur in the sentence. Combining this with Theorem 6, we obtain the main result of [10]:

Theorem 8. *Conjecture 1 is true for languages defined by sentences in which the only numerical predicate symbol is +.*

Let us sketch how Theorem 7 is proved. Once again, we can use Ramsey's theorem to uniformize a non-uniform computation. This time the uniformization functions at the level of the formula, instead of the program. Following Libkin [8], we were able to prove:

Theorem 9. *Let $L \subseteq \Sigma^*$. Suppose $L0^\omega$ is defined by an active-domain sentence ϕ with both modular and ordinary quantifiers, and arbitrary numerical predicates. Then there is an infinite subset Y of $\mathbf{N}$, and a sentence ψ, with the following properties: (a) The only numerical predicate in ψ is $<$. (b) ψ has both ordinary and modular quantifiers, and the modulus of every modular quantifier in ψ also occurs in ϕ. (c) ψ is an active-domain sentence. (d) If $w \in \Sigma^*$ and all the active letters in w are in positions belonging to Y, then $w \in L$ if and only if w satisfies ψ.*

In other words, nonuniform sentences (those with unrestricted numerical predicates) behave exactly like highly uniform sentences (those in which the only numerical predicate is $<$) when restricted to some large set of positions. This is precisely analogous to the behavior we observed in Section 1.

Now let $L \subseteq \Sigma^*$ be a language with a neutral letter defined by an active-domain sentence ϕ in which the neutral letter τ is inactive. Then $L0^\omega$ is also defined by ϕ, and thus there is an infinite subset Y of $\mathbf{N}$ and a sentence ψ as in the theorem above. Let

$$Y = \{y_1 < y_2 < \cdots\}.$$

Let $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$, and consider the word w' of length y_n such that the y_i^{th} letter of w' is σ_i, and all the other letters are equal to the neutral letter τ. Then $w \in L$ if and only if $w' \in L$ if and only if w' satisfies ψ. But since ψ uses only active-domain quantification and has only $<$ for a numerical predicate, this occurs if and only if w satisfies ψ. This means that L is regular and its syntactic monoid contains only solvable groups.

7 Conclusions

Much of what we have described can be viewed as a generalization of the work in Barrington, *et. al.* [4] on the "Crane Beach Conjecture" from the domain of first-order formulas with + to formulas with modular quantifiers and +. The outstanding challenge, of course, is to extend our non-expressibility results to formulas with arbitrary numerical predicates.

There are, however, considerable technical obstacles to doing so. We might try to approach the problem by first showing that Conjecture 1 holds when we

adjoin the numerical predicate $\times$ (integer multiplication). (The resulting logic is important from a computational standpoint, since it defines precisely the languages in a natural uniform version of ACC [3].) But in [4] it is shown that when this is done, one can define languages with neutral letters that are not regular. Together with Theorem 7 this implies, in particular, that one cannot eliminate non-active-domain quantification from such a formula. Quantifier-elimination techniques may still be useful, however – it would be enough to show the reduction to active-domain quantification under the assumption that the language defined is regular. It may be necessary to begin with very simple formulas: In Straubing [12] something of the kind is carried out for the modular analogue of boolean combinations of Σ_1 sentences. Once again, Ramsey theory is an essential ingredient.

Results like Theorem 2 show that nonsolvable groups have special computational properties. What we have been trying to do is show that the nonsolvability is essential – that monoids with only solvable groups have radically different computation capabilities. Completing this program would extend the application of finite semigroups in computation well beyond the domain of finite automata.

References

1. D. Mix Barrington, "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1", *J. Comp. Syst. Sci.* **38** (1989) 150–164.
2. D. Mix Barrington, K. Compton, H. Straubing, and D. Thérien, "Regular Languages in NC^1", *J. Comp. Syst. Sci.* **44** (1992) 478–499.
3. D. Mix Barrington, N. Immerman, and H. Straubing, "On Uniformity in NC^1", *J. Comp. Syst. Sci.* **41** (1990) 274–306.
4. D.M. Barrington, N. Immerman, Clemens Lautemann, Nicole Schweikardt, and Denis Thérien The Crane Beach Conjecture, LICS '01, 187-196.
5. D. Mix Barrington and H. Straubing, "Superlinear Lower Bounds for Bounded-Width Branching Programs", *J. Comp. Syst. Sci.* **50** (1995) 374–381.
6. D. Mix Barrington and D. Thérien, "Finite Monoids and the Fine Structure of NC^1", *JACM* **35** (1988) 941–952 .
7. M. Furst, J. Saxe, and M. Sipser, "Parity, Circuits and the Polynomial-time Hierarchy", *Math. Systems Theory,* **17** (1984) 13-27.
8. L. Libkin, "Embedded finite models and constraint databases", in E. Grädel, et. al., eds., *Finite Model Theory and its Applications,* Springer, New York (2005).
9. M. Presburger, "Ueber die Vollstaendigkeit eines gewissen Systems der Arithmetik ganzer Zahlen", in welchem die Addition als einzige Operation hervortritt." In Comptes Rendus du I congrés de Mathématiciens des Pays Slaves. Warsaw, Poland: pp. 92-101, 1929.
10. A. Roy and H. Straubing, 'Definability of Languages by Generalized First-ordr Formulas over (**N**,+)". Preprint.
11. "Circuit complexity and the expressive power of generalized first-order formulas". In Proceedings 1992 ICALP *LNCS* **623** (1992) 16–27.
12. H. Straubing, "Languages defined with modular counting quantifiers". Inform. and Comput., **166** (2001) 112-132..
13. H. Straubing, D. Thérien, and W. Thomas, "Regular Languages Defined with Generalized Quantifiers", *Information and Computation* **118** (1995) 289-301.

Complexity of Quantum Uniform and Nonuniform Automata*

Farid Ablayev and Aida Gainutdinova

Dept. of Theoretical Cybernetics,
Kazan State University
420008 Kazan, Russia
{ablayev,aida}@ksu.ru

Abstract. We present two different types of complexity lower bounds for quantum uniform automata (finite automata) and nonuniform automata (OBDDs). We call them "metric" and "entropic" lower bounds in according to proof technique used. We present explicit Boolean functions that show that these lower bounds are tight enough.
We show that when considering "almost all Boolean functions" on n variables our entropic lower bounds gives exponential ($2^{c(\delta)(n-\log n)}$) lower bound for the width of quantum OBDDs depending on the error δ allowed.
Next we consider "generalized measure-many" quantum automata. It is appeared that for uniform and nonuniform automata (for space restricted models) their measure-once and measure-many models have different computational power.

1 Introduction

We consider two types of quantum automata: uniform – *quantum finite automata* and nonuniform – *quantum oblivious read-once branching programs* (or Ordered Binary Decision Diagrams QOBDDs). Branching programs and their restriction model – ordered read-once branching programs have proven useful in a variety of domains, such as hardware verification, model checking, and other CAD applications (see for example the book by Wegener [19]). Moreover, branching programs are a very natural model for comparing the power of quantum computation with classical computation, both deterministic and randomized.

Oblivious branching programs (definitions see below) and in particular constant width OBDDs are often called *non-uniform deterministic finite automata* (NUDFAs). Constant width OBDDs are too restricted and are of minor interest (they can compute only a very small part of Boolean functions). We consider OBDDs without constant width restrictions. We call such OBDDs *non-uniform automata* (NUAs).

Our paper is organized as follows. In the section "Measure-Once Quantum Automata" we present formal definitions of automata models. Then we present

* The research supported in part by Russia Fund for Basic Research 03-01-00769

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 78–87, 2005.
© Springer-Verlag Berlin Heidelberg 2005

two different types of complexity lower bounds for quantum uniform automata (finite automata) and nonuniform automata (OBDDs). We call them "metric" and "entropic" lower bounds in according to proof technique used. We present explicit Boolean functions that show that these lower bounds are tight enough.

Next we show that when considering "almost all Boolean functions" on n variables our entropic lower bounds gives exponential $(2^{(1-(2+\theta)H(\delta))(n-\log 3n)})$ lower bound for the width of quantum OBDDs depending on the error δ allowed. Here $H(\delta)$ is Shannon entropy function and $\theta \in (0,1)$.

In the section "Generalized Measure-Many Automata" we consider "generalized measure-many" quantum automata. It is appeared that for uniform and nonuniform automata (for space restricted models) their measure-once and measure-many models have different computational power.

2 Measure-Once Quantum Automata

Uniform Automata. We consider 1-way quantum finite automata (QFA) model similar to [8]. 1-way measure-once quantum finite automaton (MO-QFA) is a tuple

$$\mathcal{Q} = \langle X,\, S,\, \{U(x) : x \in X\},\, s_0,\, F\rangle.$$

Here X is a finite input alphabet with an end-marker symbol \$, S is a finite set of states (let $d = |S|$), $\{U(x) : x \in X\}$ is a set of transition matrices ($U(x)$ is d-dimensional unitary matrix representing the transition of $\mathcal{Q}$ upon reading symbol x), $s_0 \in S$ is a starting state, and $F \subseteq S$ is the set of accepting states.

The *quantum state* of $\mathcal{Q}$ is a linear superposition of states and is represented by a d-dimensional complex unit vector

$$|\psi\rangle = z_1|s_1\rangle + z_2|s_2\rangle + \ldots + z_d|s_d\rangle$$

where $\{|s_i\rangle\}$ is the set orthonormal basis vectors corresponding to the states of $\mathcal{Q}$.

Computation of $\mathcal{Q}$ starts in superposition $|s_0\rangle$. If in the current step of computation a superposition of $\mathcal{Q}$ is $|\psi\rangle$ then after reading an input $x \in X$ the new superposition of $\mathcal{Q}$ will be $|\psi'\rangle = U(x)|\psi\rangle$.

After reading the end-marker \$ the computation of $\mathcal{Q}$ halts and $|\psi\rangle$ observed. After that, the superposition of states of $\mathcal{Q}$ collapses to a state $s_i \in S$. This observation gives $s_i \in S$ with the probability $|z_i|^2$. If we get $s_i \in F$, the input is accepted. If $s_i \in S \backslash F$, the input is rejected. The probability of accepting the input is $\sum_{s_i \in F} |z_i|^2$.

Acceptance Criteria and Complexity. Let $\delta \in (0, 1/2)$. We say that QFA $\mathcal{Q}$ bounded-error (δ-error) accepts language $L \subseteq X^*$ if words from L are accepted with probability at least $1-\delta$ and words from $X^* \backslash L$ are rejected with probability at least $1/2 + \delta$.

We denote $dim(\mathcal{Q})$ the number d of states of the automaton $\mathcal{Q}$.

Nonuniform Automata. A *branching program* (BP) is a finite directed acyclic graph which accepts some subset of $\{0,1\}^n$. Each node (except for the sink nodes) is labeled with an integer $1 \leq i \leq n$ and has two outgoing arrows labeled 0 and 1. This pair of edges corresponds to querying the i'th bit x_i of the input, and making a transition along one outgoing edge or the other depending on the value of x_i. There is a single source node corresponding to the start state, and a subset *Accept* of the sink nodes corresponding to accepting states. An input x is *accepted* if and only if it induces a chain of transitions leading from source to a node in *Accept*, and the set of such inputs is the language accepted by the program. A BP is *oblivious* if the nodes can be partitioned into levels $V_1, \ldots, V_\ell$ and a level $V_{\ell+1}$ such that the nodes in $V_{\ell+1}$ are the sink nodes, nodes in each level V_j with $j \leq \ell$ have outgoing edges only to nodes in the next level V_{j+1}, and all nodes in a given level V_j query the same bit x_{i_j} of the input. Such a program is said to have *length* ℓ, and *width* d if each level has at most d nodes. A BP is called *read-once* if, for each variable x_i, each of the paths in BP contains at most one node labeled by x_i. An OBDD is read-once BP where on each computation path the variables are tested according to the same order.

Computation on OBDD of width d (read-once restriction means that OBDD has length n) can be viewed as n step computation presented by d state finite automaton. A computation in a deterministic finite automaton is performed in according to order $x_1, \ldots, x_n$ of inputs testing and transformations in each level V_j determined by automaton transition function $\Delta : V_j \times \{0,1\} \to V_{j+1}$.

In a NUAs we allow variables to be queried in arbitrary order (fixed for a particular OBDD), and allow different transformations Δ_j in each level V_j.

Recently, several models of quantum branching programs and OBDDs have been proposed [3, 4, 7, 15, 18]. Here we consider leveled quantum OBDD (QOBDD) model without constant width restriction defined according to [3].

A QOBDD P of width d for Boolean function f on n variables that tests its n variables in order $\pi = (\pi(1) \ldots \pi(n))$ is a triple

$$P = \langle T, |\psi_0\rangle, F\rangle$$

where T is a sequence (of the length n) of d-dimensional unitary transformations of d-dimensional Hilbert space $\mathcal{H}^d$:

$$T = (\langle \pi(i), U_i(0), U_i(1)\rangle)_{i=1}^n,$$

$|\psi_0\rangle$ is the unitary vector of $\mathcal{H}^d$ (initial *state* of P). $F \subseteq \{1, \ldots, d\}$ is the set of accepting states.

Computation on P for an input $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^n$ is defined as follows:

1. A computation of P starts from the state $|\psi_0\rangle$. On the i-th step, $1 \leq i \leq n$, of computation P transforms state $|\psi\rangle$ to a state $|\psi'\rangle = U_i(\sigma_{\pi(i)})|\psi\rangle$.
2. After the n-th (last) step of quantum transformation P measures its resulting state $|\psi(\sigma)\rangle$. Measurement is presented by a diagonal zero-one projection matrix M where $M_{ii} = 1$ if $i \in F$ and $M_{ii} = 0$ if $i \notin F$. The probability $p_{accept}(\sigma)$ of P accepting input σ is defined by

$$p_{accept}(\sigma) = ||M|\psi(\sigma)\rangle||^2.$$

Acceptance Criteria and Complexity. Let $\delta \in (0, 1/2)$. We say that an QOBDD P computes f with *bounded error* (with δ-*error*) if for all $\sigma \in f^{-1}(1)$ the probability of P accepting σ is at least $1 - \delta$ and for all $\sigma \in f^{-1}(0)$ the probability of P accepting σ is at most δ. We call such QOBDD δ-error QOBDD and denote it P_δ.

We denote $width(P)$ the dimension d of the Hilbert space $\mathcal{H}^d$ of program P. For a Boolean function f we define its *quantum width* $Qwidth_\delta(f)$ to be the width of the best quantum OBDD that computes f with δ-error.

Notes

- According to our considerations we have the following evident relation on the complexity of uniform and nonuniform automata. If finite automaton A is presented in the OBDD setting then $dim(A) = width(A)$.
- Notice that uniform measure-once quantum finite automata can recognize only a proper subclass of regular languages (see for example [8] for more information). For a nonuniform model this property modifies into the statement that arbitrary Boolean function on n variables can be computed (exactly) by quantum OBDD of width 2^n [3].

2.1 Lower Bounds

In this section we consider two lower bounds for quantum automata complexity. We call them "metric" and "entropic" lower bounds in according to proof technique used. Both lower bounds are valid for uniform and nonuniform models. We present the "metric" lower bound for the uniform case and the "entropic" – for the nonuniform case.

Metric Lower Bound. In [2] we proved the following two lower bounds for quantum automata. We then applied this lower bound technique for QOBDDs [3].

1. Let $L \subseteq \Sigma^*$ be a language bounded-error accepted by MO-QFA $\mathcal{Q}$. Let here and below A_L is a minimal finite deterministic automaton accepting L. Then it holds that
   $dim(\mathcal{Q}) = \Omega\big((\log dim(A_L))/(\log\log dim(A_L))\big)$.
2. If $\delta \in (0, 1/8]$ then we have more precise lower bound
   $dim(\mathcal{Q}) \geq \big(\log dim(A_L)\big)/\big(\log(1 + 1/\gamma)\big)$ where $\gamma = \gamma(\delta)$.

In [4] we generalize our second lower bound which is true for QOBDDs for arbitrary $\delta \in (0, 1/2)$. Now we can state for quantum automata the similar lower bound.

Theorem 1. *Let $\varepsilon \in (0, 1/2)$. Let $L \subseteq \Sigma^*$ be a language δ-error accepted by MO-QFA $\mathcal{Q}$. Then it holds that*

$$dim(\mathcal{Q}) \geq \frac{\log dim(A_L)}{2\log(1 + 1/\varepsilon)}, \quad \textit{where} \quad \varepsilon = 1/2 - \delta.$$

Proof Sketch. The proof uses essentially the same techniques as were used for Theorem 3 of [17] and Proposition 6 of [13], which show that stochastic and quantum finite state automata that accept with bounded probability can be simulated by deterministic finite state automata and therefore accept regular languages.

The idea is that if two state vectors are within a distance θ of each other, where θ depends on δ, then if the same operators are applied to both they must either both accept or both reject, and so are equivalent. Since only a finite number of balls of radius $\theta/2$ can fit into the state space we end up with a finite number of states. □

Entropic Lower Bound. The next lower bound uses the communication complexity approach and its proof uses entropic technique.

We informally recall one-way communication protocols and communication complexity (see for example [11]). Let $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$ be a Boolean function. In a one-way communication protocol player Alice and Bob receive x and y and compute $f(x,y)$. Alice starts computation on her part of input and sends a message (binary string) to Bob. Bob on getting the message and his part of an input performs computation and produces a result. The communication complexity of a protocol is the worst case number of bits exchanged. The deterministic communication complexity $D(f)$ of f is the complexity of an optimal protocol for f.

In a quantum protocol both players have a private set of qubits. Some qubits are initialized to the input before the protocol starts, the other qubits are in state $|0\rangle$. Alice performs some unitary transformation on her qubits and then sends some of the qubits (channel qubits) to Bob. Bob performs some unitary transformation on channel qubits and his part of qubits. Then some qubits are measured and the result is taken as the output.

In a (bounded error) quantum protocol the correct answer must be given with a probability $1-\delta$ for some $\delta \in (0,1/2)$. The (bounded error) quantum complexity of a function is denoted $Q_\delta(f)$.

The communication matrix of f is a matrix with $CM_f[x,y] = f(x,y)$. Denote $cs(CM_f)$ the smallest number of columns of CM_f that are necessary to distinguish all different rows of CM_f (the cardinality of minimal control set for communication matrix).

With the function f, we associate a $2^{n_1} \times 2^{n_2}$ communication matrix CM_f whose (x,y)-th entry, $CM_f[x,y]$ is $f(x,y)$. For $V \subseteq \{0,1\}^{n_1}$ and $W \subseteq \{0,1\}^{n_2}$ denote $CM^{V,W}$ the $|V| \times |W|$ submatrix of CM_f formed by rows V and columns W of CM_f. Denote $\mathcal{M}_f$ a set of submatrices of CM_f with *pairwise different rows.* For matrix $CM^{V,W} \in \mathcal{M}_f$ call set W a *control set.*

Theorem 2. *For every function* $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, *every* $\delta \in (0,1/2)$, *and every* $CM^{V,W} \in \mathcal{M}_f$,

$$Q_\delta(f) \geq \log |V| - |W| H(\delta).$$

where $H(\delta) = -\delta \log \delta - (1-\delta)\log(1-\delta)$.

Proof. The proof uses entropic approach and omitted. Such approach used in several papers for classical randomized and quantum communication computations [1, 10, 11]. □

Corollary 21 *For every function* $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, *every* $\delta \in (0, 1/2)$,

$$Q_\delta(f) \geq D(f) - cs(CM_f)H(\delta).$$

Proof. Denote $nrow(CM_f)$ the number of pairwise distinct rows of matrix CM_f. The one-way deterministic communication complexity $D(f)$ of f is easily seen to be $\lceil \log(nrow(CM_f)) \rceil$ [20]. □

Klauck's lower bound (Theorem 3) is a corollary of Theorem 2. Recall that for a Boolean function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, a set $W \subseteq \{0,1\}^{n_2}$ is shattered, if for all $R \subseteq W$ there is an υ such that $f(\upsilon, \omega) = 1 \Leftrightarrow \omega \in R$ for all $\omega \in W$. The *VC*-dimension $VC(f)$ of f is the maximal size of a shattered set.

Theorem 3 ([11]). *For every Boolean function* $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, *every* $\delta \in (0, 1/2)$,

$$Q_\delta(f) = VC(f)(1 - H(\delta)).$$

Proof. According to definition of $VC(f)$ there exists matrix $M^{V,W} \in \mathcal{M}_f$, where W is shattered set with $|W| = VC(f)$ and $|V| = 2^{|W|}$. □

Communication complexity approach establishes a lower bounds for different computational models [9, 12] and in particular for OBDDs. This fact is explored in numerous papers. We refer to the book [19] for more information. The idea of such approach (which is folklore now) is that one can view on OBDD as a specific one-way communication protocol. We display this approach for quantum OBDDs in the following statement.

Property 1 *Let* $\delta \in (0, 1/2)$. *Let QOBDD* P *computes* f *with* δ*-error. Let* $\pi = (L, R)$ *be a partition of inputs of* f *between Alice and Bob with* L *and* R *defined according to an ordering* τ *of inputs of* P. *That is,* P *can read variables from* R *only after reading variables from* L *and cannot read variables from* L *after starting reading variables from* R. *Then*

$$width(P) \geq 2^{Q^\pi_\delta(f)},$$

here (and below in the paper) Q^π_δ *is communication complexity of function* f *in respect to partition* π *of inputs.*

Proof Sketch. Let $L = \{z_1, \ldots, z_{n_1}\}$ and $R = \{y_1, \ldots, y_{n_2}\}$. Let program P tests its variables in the order $z_1, \ldots, z_{n_1}, y_1 \ldots y_{n_2}$ We view quantum OBDD P as the following one-way quantum protocol. Alice having her part $\sigma = \sigma_1 \ldots \sigma_{n_1}$ of input, simulates computations on σ in according to sequence T of unitary transformations of P and gets state $|\psi(\sigma)\rangle \in \mathcal{C}$. She sends message $|\psi(\sigma)\rangle$ to Bob. Bob, having his part $\gamma = \gamma_1 \ldots \gamma_{n_2}$ of input and message $|\psi(\sigma)\rangle$, performs computations in according to sequence T of unitary transformations of P. Then Bob performs measurement in according to P and outputs result (*accept* or *reject*). □

Uniform Automata. The example of language L_p presented in [6] shows that the metric lower bound from Theorem 1 is tight for all $\delta \in (0, 1/2)$. For a prime p language L_p over a single letter alphabet defined as follows: $L_p = \{u : |u| \text{ is divisible by } p\}$.

Theorem 4 ([6]). *For any $\delta > 0$, there is a MO-QFA with $O(\log p)$ states recognizing L_p with probability $1 - \delta$.*

Clearly we have that any finite deterministic automaton for L_p needs at least p states. In [6] it was shown that constant bounded error finite probabilistic automata also need at least p number of states to recognize L_p.

Application of Entropic lower bound for quantum finite automata for explicit regular language was presented in [11].

Nonuniform Automata. Application of metric lower bound for QOBDDs for MOD_p function was presented in [3]. Our results [3] shows that metric lower bound for MOD_p is tight enough. In [18] for *indirect storage access* function ISA_n on $n + \log n$ variables proved that any QOBDD δ-error computed ISA_n has size $2^{\Omega(n/logn)}$. Theorem 2 and Property 1 give the following lower bound.

Property 2 $Qwidth_\delta(ISA_n) \geq 2^{(1-H(\delta))n/logn}$.

Below we consider lower bound for complexity for almost all Boolean functions. Denote by $\mathcal{F}(n)$ the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}$. Let E be some property of functions from $\mathcal{F}(n)$. Denote by $\mathcal{F}^E(n)$ the subset of functions from $\mathcal{F}(n)$ without property E. We say that almost all functions have the property E if

$$| \mathcal{F}^E(n) | / | \mathcal{F}(n) | \to 0, \quad \text{as} \quad n \to \infty.$$

Theorem 5. *For almost all functions $f : \{0,1\}^n \to \{0,1\}$, for $\delta \in (0, 1/2)$*

$$Qwidth_\delta(f) \ \geq \ 2^{(1-(2+\theta)H(\delta))(n-\log 3n)}.$$

Proof Sketch. For an ordering τ of n variables testing for QOBDD let $\pi = (L, R)$ be a corresponding partition of the set of variables such that $|R| = \lceil \log 3n \rceil$, be a partition of n variables of function f.

Consider communication computation for f for the partition π of inputs. Elementary counting proves that 1) $DC(f_n) = n$ for almost all functions f; 2) for an arbitrary $\theta \in (0, 1)$ it holds that $n \leq cs(CM) < (2 + \theta)n$ for almost all functions f. Now Theorem 2 gives that for almost all functions, for $\delta \in (0, 1/2)$

$$Q^\pi_\delta(f) \ \geq \ (1 - (2 + \theta)H(\delta))(n - \log 3n).$$

There are $C_n^{|R|}$ different partitions π of the set of n variables determined by different possible orderings τ of variables testings by QOBDDs. Further counting together with the previous inequality and Property 1 proves that for almost all functions f it holds that $Qwidth_\delta(f) \ \geq \ 2^{(1-(2+\theta)H(\delta))(n-\log 3n)}$. □

3 Generalized Measure-Many Automata

In this section we consider one-way generalized measure-many quantum uniform finite automata $gmQFA$ that allow measurements in each step of computation. For quantum circuits such model has already been considered by Aharonov, Kitaev and Nisan [5], who have proposed to describe the states and the computations of such quantum circuits by mixed states and superoperators, resp. In general such measure-many automaton is completely specified by representing the state of automaton as a density matrix and making the superoperators instead of purely unitary operators (we do not do this here).

In each step $gmQFA$ performs unitary transformation determined by the result of a previous measurement. Let us give more details on the model. d-dimensional $gmQFA$ acts on the d-dimensional Hilbert space $\mathcal{H}^d$. For $\mathcal{H}^d$ consider a set $\mathcal{O} = \{E_1, \ldots, E_t\}$ of mutually orthogonal subspaces such that $\mathcal{H} = E_1 \oplus \ldots \oplus E_t$.

Formally d-dimensional $gmQFA$ is a 5-tuple $Q = \langle X, S, \mathcal{U}, |\psi(0)\rangle, F \rangle$, where X is a finite input alphabet with an end-marker symbol \$, $S = \{1, \ldots, d\}$ is a finite set of basis states, $\mathcal{U} = \{U(x) : x \in X\}$ is a set of transitions of automaton Q (transition $U(x)$ acts unitary on each of subspaces $E_1, \ldots, E_t$), $F \subseteq S$ is the set of accepting states of Q. Automaton Q starts its work from the initial pure state $|\psi(0)\rangle$.

Every *computational step* of Q consists of two parts:

1. A *computational measurement* of current state $|\psi\rangle = \sum_{i=1}^{t} |\psi_i\rangle$ (here $|\psi_i\rangle$ is the projection of ψ into E_i) by $\mathcal{O}$ has the following consequences:
   – One of the subspaces $E_1, \ldots, E_t$, say E_i, is selected with the probability $p_i = |||\psi_i\rangle||^2$.
   – After the measurement, the configuration $|\psi\rangle$ "collapses" into the (renormalized) configuration $|\alpha_i\rangle = \frac{|\psi_i\rangle}{|||\psi_i\rangle||}$. All information not in $|\psi_i\rangle$ is irreversibly lost.
2. *Transformation:* Let x be an input of the automaton on the current step. If $x \in X$ then Q applies transformation $U(x) \in \mathcal{U}$ to $|\alpha_i\rangle$. If $x =$ \$ then automaton Q stops computation and applies a final measurement. The automaton accepts (rejects) an input word if $s \in F$ ($s \notin F$).

Theorem 6. *For arbitrary regular language L there exists a $gmQFA$ Q that recognizes L exactly.*

Proof Sketch. The definition of $gmQFA$ allows to simulate classical deterministic finite automaton by using measurement in each step in respect to basis states and then apply a transition determined by an input and a current state. □

Theorem 7. *Let L be a language bounded-error recognized by $gmQFA$ Q. Then L is a regular language.*

Proof Sketch. The proof is straightforward. That is, we construct a probabilistic automaton P that bounded error recognizes L. □

Discussion. It is known that measured once quantum automata can recognize only a proper subclass of regular languages [13]. Theorem 6 shows that generalized measured many quantum automata is more powerful (they can recognize arbitrary regular languages). Mention that the same effect is true for the nonuniform model [18]. That is in [18] presented Boolean function NO_n that can be computed exactly by linear size generalized measured many QOBDD ($gmQOBDD$), contrary NO_n is exponentially hard for ordinary QOBDD.

Remind that in [3] it is shown that arbitrary Boolean function can be computed by 2^n width QOBDD exactly.

References

1. Farid M. Ablayev. Lower Bounds for One-way Probabilistic Communication Complexity. In *Proc. of the 20th International Colloquium on Automata, Languages and Programming*, 1993, LNCS; Vol. 700, 241-252.
2. Farid Ablayev and Aida Gainutdinova. On the Lower Bound for One-Way Quantum Automata. *Proc. of MFCS'2000, LNCS 1893, Springer-Verlag*, 2000, 132-140.
3. F. Ablayev, A. Gainutdinova, and M. Karpinski. On the computational power of quantum branching programs. *Proc. FCT 2001*, LNCS 2138, 2001, 59–70.
4. F. Ablayev, C. Moore, and C. Pollett. Quantum and stochastic branching programs of bounded width. *Proc. 29th Intl. Colloquium on Automata, Languages and Programming*, 2002, 343-354.
5. D. Aharonov, A. Kitaev, and N. Nisan. Quantum circuits with mixed states. In *Proc. of 30th STOC*, 1998, 20-30.
6. A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalization. *Proc. of the 39th IEEE Conference on Foundation of Computer Science*, 1998, 332-342. See also quant-ph/9802062 v3.
7. A. Ambainis, L. Schulman, and U. Vazirani. Computing with highly mixed states. *Proc. 32nd ACM Symp. on Theory of Computing*, 2000, 697–704.
8. A. Brodsky and N. Pippenger. Characterizations of 1-way quantum finite automata. quant-ph/9903014, 1999.
9. J. Hromkovich. Communication complexity and parallel computing. *Springer-Verlag, Berlin, Heidelberg, New York*, 1997.
10. Ilan Kremer, Noam Nisan, Dana Ron. On randomized one-round communication complexity. In *Proc. of the 27th annual ACM symposium on Theory of computing*, 1995, 596-605.
11. H. Klauck. On quantum and probabilistic communication: La Vegas and one-way protocols. *Proc. of the 32nd ACM Symp. Theory of Computing*, 2000.
12. E. Kushilevitz and N. Nisan. Communication Complexity. *Cambridge University Press*, 1997.
13. A. Kondacs, J. Watrous. On the power of quantum finite state automata. In *Proc. of the 38th IEEE Conference on Foundation of Computer Science*, 1997, 66-75.
14. C. Moore and J. Crutchfield. Quantum automata and quantum grammars, quant-ph/9707031.
15. M. Nakanishi, K. Hamaguchi, and T. Kashiwabara. Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. *Proc. 6th Intl. Conf. on Computing and Combinatorics (COCOON)* Lecture Notes in Computer Science 1858, 2000, 467–476.

16. A. Nayak. Optimal lower bounds for quantum automata and random access codes. *Proc. of the 40th IEEE Conference on Foundation of Computer Science*, 1999, 369-376. See also quant-ph/9904093
17. M.O. Rabin. Probabilistic automata. *Information and Control*, 1963, 6, 230-244.
18. M. Sauerhoff and D. Sieling. Quantum branching programs and space-bounded nonuniform quantum complexity. *ph/0403164, March 2004.*
19. I. Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications, 2000.
20. A. Yao. Some Complexity Questions Related to Distributive Computing. *Proc. of the 11th Annual ACM Symposium on the Theory of Computing*, 1979, 209-213.

Membership and Finiteness Problems for Rational Sets of Regular Languages*

Sergey Afonin and Elena Hazova

Lomonosov Moscow State University, Institute of Mechanics
Moscow, Russia
serg@msu.ru

Abstract. Let Σ be a finite alphabet. A set $\mathcal{R}$ of regular languages over Σ is called *rational* if there exists a finite set $\mathcal{E}$ of regular languages over Σ, such that $\mathcal{R}$ is a rational subset of the finitely generated semigroup $(\mathcal{S}, \cdot) = \langle \mathcal{E} \rangle$ with $\mathcal{E}$ as the set of generators and language concatenation as a product. We prove that for any rational set $\mathcal{R}$ and any regular language $R \subseteq \Sigma^*$ it is decidable (1) whether $R \in \mathcal{R}$ or not, and (2) whether $\mathcal{R}$ is finite or not.

1 Introduction

Let Σ be a finite alphabet and $\mathcal{E} = \{E_1, \dots E_k\}$ be a set of regular languages over Σ. Consider the free semigroup $(\mathcal{S}, \cdot) = \langle \mathcal{E} \rangle$ generated by the elements of $\mathcal{E}$ with respect to language concatenation. Decidability of membership and finiteness problem are one of the most natural questions. The membership problem may be considered as a special form of language factorization: a regular language $R \subseteq \Sigma^*$ belongs to the semigroup $\mathcal{S} = \langle \mathcal{E} \rangle$ if and only if there exists a sequence $i_1, i_2, \dots, i_n$ of integers such that $1 \leqslant i_p \leqslant k$ $(p = 1 \dots n)$ and

$$R = E_{i_1} E_{i_2} \dots E_{i_n} \tag{1}$$

This problem occurs in many practical situations. Let us consider one possible application, namely, view-based query processing in semistructured databases [9].

A powerful mathematical model for semi-structured data, i.e. data with irregular or partially unknown structure, is an edge-labeled directed graph [1]. Nodes of the graph correspond to objects in a subject area, and edges are relations between objects. Complex relationships between objects are represented in this model as *paths* in the database graph and the following problem typically arises: for given regular language R (query), find all the pairs (u, v) of the graph nodes such that there exists at least one labeled path between u and v in the graph and its labels comprise a word in R. Although this problem has polynomial time complexity (one should check non-emptiness of the intersection of some regular languages), the whole database graph may need to be searched, which is inefficient.

* This research was supported in part by the grant No. 10002-251 of the RAS program No. 17 "Parallel computations and multiprocessor systems"

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 88–99, 2005.
© Springer-Verlag Berlin Heidelberg 2005

One possible method toward increasing query processing efficiency is using views [9]. Suppose that we know the results for the queries, corresponding to regular languages $\mathcal{E} = \{E_1, \ldots, E_k\}$. The question is, can this data be used during evaluation of an arbitrary query R? Is it possible to represent the language R in terms of E_i? Formally, this is the language substitution problem. The efficiency of query processing (especially parallel processing) significantly depends on query structure and the decomposition of the form (1) can be effectively evaluated [3].

Rational sets are natural extension of semigroups of regular languages. Such objects appear, for instance, in the described domain if we are given not only a set of views, but also an additional application level constraints on admissible query rewritings. Rational sets are also related to univariable *language equations* [15]. Consider, for example, linear equation $R = EX$, where R and E are given regular languages and X is a variable. It is decidable whether this equation has a solution but in general there are infinitely many regular solutions [13]. Rational sets could be a suitable formalism for their characterization.

The membership problem for the finitely generated semigroup of regular languages was shown decidable by K. Hashiguchi in [11]. More precisely, it was shown that for any finite set $\mathcal{E}$ of regular languages over Σ, and subset $T \subseteq \{\cdot, \cup, ^*\}$ of operations it is decidable whether or not the language R may be constructed from E_i using a finite number of operations from T. The solution in [11] is based on reduction to the limitedness property of distance automata [10] (described in the next section). The latter problem is highly related to finite section problem of finitely generated semigroup of square matrices over the *tropical semiring* (see [19, 23] for a survey). The decidability of the finite section problem for matrices over the semiring of regular languages over unary alphabet was proved in [18]. To our best knowledge, this is the only known result related to the membership problem for rational sets.

As a related work we mention the classical results, such as Krohn-Rhodes decomposition theorem for transformation semigroups [4], or language factorization into a finite number of stars and primes [7], and very recently proved finite language substitution problem [5, 14]. These works, however, deal with "unrestricted" case of the problem, when factors are arbitrary languages of certain classes (i.e. stars and primes or finite languages).

The structure of the paper is as follows. In section 2 we present definitions and briefly describe useful results on distance automata. In section 3 decidability of the membership problem for rational sets of regular languages over arbitrary alphabets is proved. The finiteness problem for finitely generated semigroup and rational set of regular languages is considered in section 4. The conclusion discusses the results and directions for future work.

2 Background and Definitions

An *alphabet* is a finite non-empty set of *symbols*. A finite sequence of symbols from an alphabet Σ is called a *word* in Σ. The *empty* word is denoted by ε. Any

set of words is called a *language* in Σ. Σ^* denotes the set of all words (including the empty word) in a given alphabet, Σ^+ denotes the set of all non-empty words in Σ, $\varnothing$ is an empty language (containing no words), and 2^{Σ^*} is the set of all languages in Σ.

The *union* of languages L_1 and L_2 is the language $L_1 + L_2 = \{w \in \Sigma^* \mid w \in L_1 \vee w \in L_2\}$. The language $L_1L_2 = \{w \in \Sigma^* \mid \exists w_1 \in L_1, w_2 \in L_2 : w = w_1w_2\}$ is called a *concatenation* of L_1 and L_2. $L^k = LL^{k-1}$ is L to the power k. By definition, L to the power zero is the empty word: $L^0 = \{\varepsilon\}$. The Kleene closure (or star) of L is the language $L^* = \cup_{k=0}^{\infty} L^k$. A language is *regular* if it can be obtained from letters of the alphabet, empty language, and $\{\varepsilon\}$ using a finite number of operations of concatenation, union and closure.

A *semigroup* is a set S equipped with associative binary product. A semigroup is called *finitely generated* if there exists a finite set $G \subseteq S$ of *generators*, such that any element of S is a finite product of elements of G. Let Σ be an alphabet and $\mathcal{E} = \langle E_1, \dots E_k \rangle$ be a set of regular languages in Σ. The concatenation of regular languages is regular and we write $(\mathcal{S}, \cdot) = \langle \mathcal{E} \rangle$ for a finitely generated semigroup, generated by $\mathcal{E}$ with concatenation as a semigroup product.

Let Δ be an alphabet and $(S, \cdot)$ be a semigroup. A morphism $\varphi : \Delta^+ \to S$ is any function satisfying $\varphi(uv) = \varphi(u)\varphi(v)$ for all $u, v \in \Delta^+$. The morphism $\varphi : \Delta^+ \to 2^{\Sigma^*}$ is called a *regular language substitution* if $\varphi(\delta)$ is a regular language over Σ for all $\delta \in \Delta$. Sometimes it is more convenient to consider φ as a morphism between free monoids Δ^* and $\mathcal{S}^1$, so we extend φ by $\varphi(\varepsilon) = \{\varepsilon\}$. For regular language $L \subseteq \Delta^+$ by $\varphi(L)$ we mean the language $\varphi(L) = \bigcup\limits_{w \in L} \varphi(w)$.

With every semigroup $(\mathcal{S}, \cdot) = \langle E_1, \dots, E_k \rangle$ of regular languages we can associate a language substitution φ. In this paper we will use specific morphisms, that are natural extensions of isomorphisms between Δ and the set of generators of a finitely generated semigroup.

Definition 2.1. *Let Δ be a finite alphabet and $S = \langle G \rangle$ be a finitely generated semigroup. A* choice of generators *for S is a morphism $\varphi : \Delta^+ \to S$, such that $\langle \varphi(\Delta) \rangle = S$ and $\varphi(\delta_1) = \varphi(\delta_2)$ if and only if $\delta_1 = \delta_2$.*

Conversely, every regular language substitution $\varphi : \Delta^+ \to 2^{\Sigma^*}$ generates the semigroup $\mathcal{S}_\varphi = \langle \{\varphi(\delta) \mid \delta \in \Delta)\} \rangle$.

Definition 2.2 (Rational set). *A set $\mathcal{R}$ of regular languages over Σ is called* rational *if there exists a finite alphabet Δ, a regular language $K \subseteq \Delta^+$, and a regular language substitution $\varphi : \Delta^+ \to 2^{\Sigma^*}$, such that*

$$\mathcal{R} = \{\varphi(w) \mid w \in K\}.$$

Thus, rational sets of regular languages are rational subsets of finitely generated semigroups of regular languages. We shall say that the pair (K, φ) is a *representation* of a rational set and write $\mathcal{R} = (K, \varphi)$.

Definition 2.3. *Let $\varphi : \Delta^+ \to 2^{\Sigma^*}$ be a regular language substitution. The* maximal rewriting *of a regular language $R \subseteq \Sigma^*$ with respect to φ is the set*

$$M_\varphi(R) = \{w \in \Delta^+ \mid \varphi(w) \subseteq R\}.$$

Any subset of $M_\varphi(R)$ *is called a* rewriting *of* R. *The rewriting* M *of* $R \subseteq \Sigma^*$ *is called* exact *if* $\varphi(M) = R$.

Theorem 2.1 (Calvanese et al. [8]). *Let* $\varphi : \Delta^+ \to 2^{\Sigma^*}$ *be a regular language substitution. For any regular language* $R \subseteq \Sigma^*$ *the maximal rewriting* $M_\varphi(R)$ *is a regular language over* Δ.

A *(non-deterministic) automaton* is a tuple $\mathcal{A} = \langle \Delta, Q, \rho, q_\circ, F \rangle$, where Q is a finite set of states, Δ is an input alphabet, $\rho \subseteq Q \times \Delta \times Q$ is a set of transitions, $q_\circ \in Q$ is the initial state and $F \subseteq Q$ is a set of final states. A successful path π in $\mathcal{A}$ is a sequence $(q_1, \delta_1, q_2)(q_2, \delta_2, q_3) \dots (q_{m-1}, \delta_m, q_m)$ of transitions such that $q_1 = q_\circ$ and $q_m \in F$. We call the word $\delta_1 \delta_2 \dots \delta_m$ the *label* of π. The language $L(\mathcal{A})$ of the automaton $\mathcal{A}$ is the set of all words w such that there exists a successful path π labeled by w.

A *distance automaton* over an alphabet Δ is a tuple $\mathcal{A} = \langle \Delta, Q, \rho, q_\circ, F, d \rangle$ where $\langle \Delta, Q, \rho, q_\circ, F \rangle$ is an automaton and $d : \rho \to \{0, 1\}$ is a distance function, which can be extended to a function on words as follows. The distance $d(\pi)$ of a path π is the sum of the distances of all edges in π. The distance $\mu(w)$ of a word $w \in L(\mathcal{A})$ is the minimum of $d(\pi)$ for all successful paths π labeled by w.

Definition 2.4. *A distance automaton* $\mathcal{A}$ *is called* limited *if there exists a constant* M *such that* $d(w) < M$ *for all words* $w \in L(\mathcal{A})$.

Theorem 2.2 (Hashiguchi [10, 16]). *For any distance automaton* $\mathcal{A}$ *it is decidable whether* $\mathcal{A}$ *is limited or not.*

3 Decidability of the Membership Problem

In this section we prove that the following

Theorem 3.1. *For any rational set* $\mathcal{R} = (K, \varphi)$ *of regular languages over* Σ *and any regular language* $R \subseteq \Sigma^*$ *is it decidable whether* $R \in \mathcal{R}$ *or not.*

The proof is based on reduction to the limitedness problem for distance automata and essentially is the same as the proof by K. Hashiguchi [11] for the membership problem for finitely generated semigroup of regular languages.

Let us start with the following observation.

Proposition 3.1. *Let* $\mathcal{R} = (K, \varphi)$ *be a rational set of regular languages and* $\varphi : \Delta^+ \to \mathcal{S}$ *be a regular language substitution. A regular language* $R \subseteq \Sigma^*$ *is in* $\mathcal{R}$ *only if its rewriting* $M = M_\varphi(R) \cap K$ *is exact.*

Let $M_\varphi(R) \cap K$ be an exact rewriting of R. It is well known that any regular language can be represented as *finite union* of union-free languages. (Recall, that a regular language is called *union-free* if it can be obtained form the letters of the alphabet and the empty word using only concatenation and iteration.) Let

$$M_\varphi(R) \cap K = \bigcup_{i=1}^{n} M_i, \tag{2}$$

be such representation. Clearly, $R \in \mathcal{R}$ only if $\varphi(M_i) = R$ for some i. Hence the original membership problem can be reformulated as follows.

Problem 3.1. Given a regular language $R \subseteq \Sigma^*$, a set of regular languages $\mathcal{E} = \{E_1, \ldots, E_k\}$ over Σ, and a union-free regular language $M \subseteq \Delta^*$ such that $\varphi(M) = R$ one should decide whether there exists a word $w \in M$ such that $\varphi(w) = R$.

Consider the following example.

Example 3.1. Let $\Sigma = \{a, b\}$, $R = a^* + a^*ba^*b(a+b)^*$ and $\varphi(\delta_1) = a^* + a^*b(a+b)^*b$, $\varphi(\delta_2) = (bab)^*$. The maximal rewriting of R wrt $\mathcal{E}$ is the language

$$M_\varphi(R) = (\delta_1 + \delta_2)^*$$

The union-free decomposition of this language contains only one component:

$$M = (\delta_1^*\delta_2^*)^*$$

This language contains the word $w = \delta_1\delta_2\delta_2\delta_1\delta_1$, and one can verify that $\varphi(w) = R$. □

We shall say that a word $v \in \Delta^+$ is a *shuffle extension* of a word $u \in \Delta^+$, in notation $u \rightsquigarrow v$, if there exist $\alpha_0, \alpha_1, \ldots, \alpha_{|u|} \in \Delta^*$ such that $v = \alpha_0 u_1 \alpha_1 u_2 \ldots \alpha_{n-1} u_n \alpha_n$ and $u = u_1 u_2 \ldots u_n$.

Proposition 3.2. *Let M be a union-free language over Δ. Then, for any pair of words $u, v \in M$ there exists a word $w \in M$ such that w is a shuffle extension of both u and v.*

Proof. If M is finite, then it contains only one word and the statement is trivial. Let M be an infinite language. Since M is union free it is either a star or a concatenation of union-free languages. If M is a star, then for any words $u, v \in M$ the concatenation $w = uv$ is also in M.

Let M be a concatenation of union-free languages:

$$M = M_1 M_2 \ldots M_p,$$

where M_i $(1 \leqslant i \leqslant p)$ is either a letter, or a star language. Every word $x \in M$ has a factorization of the form $x = x_1 x_2 \ldots x_p$, where $x_i \in M_i$ for all $i \in [1, p]$. Let $u = u_1 u_2 \ldots u_p$ and $v = v_1 v_2 \ldots v_p$. Consider the word

$$w = \mathrm{ext}(u_1, v_1)\mathrm{ext}(u_2, v_2) \ldots \mathrm{ext}(u_p, v_p),$$

where $\mathrm{ext}(u_i, v_i) = u_i$ if M_i is a letter, and $\mathrm{ext}(u_i, v_i) = u_i v_i$ if M_i is a star. Clearly, $w \in M$ because $\mathrm{ext}(u_i, v_i) \in M_i$ for all $i \in [1, p]$ and this word is a shuffle extension of u and v. □

Corollary 3.1. *For any finite subset F of a union-free language M there exists a word $w \in M$, such that w is a shuffle extension for every word in F.*

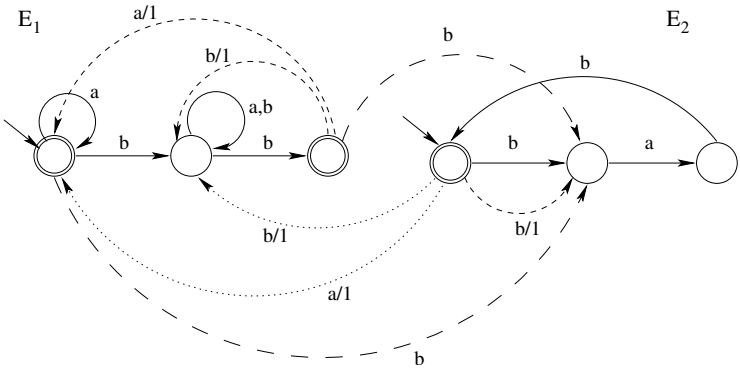


Fig. 1. Distance automaton construction for Example 3.1

The above proposition states that union-free languages has a grid structure with respect to shuffle extension relation.

Lemma 3.1. *Let M be a union-free language over Δ, and $\varphi : \Delta^+ \to 2^{\Sigma^*}$ be a regular language substitution. If $\varepsilon \in \varphi(\delta)$ for all $\delta \in \Delta$, then the following statements are equivalent:*

1. *there exists a word $w \in M$ such that $\varphi(w) = \varphi(M)$*
2. *there exists a constant $C \in \mathbb{N}$ such that*

$$\forall u \in \varphi(M)\ \exists w' \in M\ : |w'| < C, u \in \varphi(w') \tag{3}$$

Proof. It is evident that 2 follows from 1. We simply can choose C to be equal to the length of w. Consider the inverse implication.

Condition (3) means that there exists a finite language $F \subseteq M$ satisfying $\varphi(F) = \varphi(M)$. Since M is union-free then, according to the Corollary 3.1, there exists a word $w \in M$ which is a shuffle extension of all the words in F. The statement of the lemma follows from the fact that if $\varepsilon \in \varphi(\delta)$ for all $\delta \in \Delta$, then $u \rightsquigarrow w \Rightarrow \varphi(u) \subseteq \varphi(w)$. □

Condition (3) of the above lemma can be checked by straightforward reduction to the limitedness property of a distance automaton. Let us illustrate this reduction by Example 3.1 (see Fig. 1). Consider the standard procedure for construction of Σ-automaton recognizing the language $\varphi(M) = (\varphi(\delta_1)^*\varphi(\delta_2)^*)^*$. First, build up automata $\mathcal{A}_1$ and $\mathcal{A}_2$ recognizing $E_1 = \varphi(\delta_1)$ and $E_2 = \varphi(\delta_2)$, respectively. On Fig. 3.1 these automata are shown by solid transitions. Then, convert $\mathcal{A}_1$ to $\mathcal{A}'_1$ by addition "loop transitions" connecting the final states of $\mathcal{A}_1$ with the successors of the initial state (shown by dashed transitions). We assign 0 to all the edges of $\mathcal{A}_{1,2}$, and 1 to all "loop" transitions. Now, connect the final states of $\mathcal{A}'_1$ with the successors of the initial state of $\mathcal{A}'_2$ (long dashed). Thus, we have the automaton recognizing $E_1^*E_2^*$. Finally, by adding "loop" transitions (marked by 1 and shown as dotted transition) we get the desired automaton.

For any union-free regular language $L \subseteq \Delta^+$ by $\mathcal{A}_L$ we mean a distance Σ-automaton constructed as described above from a union-free regular expression for L.

Lemma 3.2. *Let $\mathcal{R} = (K, \varphi)$ be a rational set of regular languages over Σ, $R \subseteq \Sigma^*$ be a regular language, and $M_1 + M_2 + \ldots + M_n$ be a union free decomposition of the rewriting $M = M_\varphi(R) \cap K$. If $\varepsilon \in \varphi(\delta)$ for all $\delta \in \Delta$, then $R \in \mathcal{R}$ if and only if there exists $i \in [1, n]$ such that*

1. *the rewriting M_i is exact, i.e. $\varphi(M_i) = R$, and*
2. *the distance Σ-automaton $\mathcal{A}_{M_i}$ is limited.*

The following example illustrates that the limitedness of the automaton $\mathcal{A}_M$ itself is not sufficient if elements of $\mathcal{E}$ may not contain the empty word.

Example 3.2. Let Σ be an unary alphabet, say $\Sigma = \{a\}$. Consider the semigroup generated by languages $\varphi(\delta_1) = \{\varepsilon\} + \{a^2\}$ and $\varphi(\delta_2) = \{a^8 + a^{64}\} + \{a^{64+2i},\ i \geqslant 1\}$. Let $R = \{a^{2i},\ i \geqslant 4\}$ and $M = \delta_1^* \delta_2 \delta_2^*$.

The distance Σ-automaton $\mathcal{A}_M$ is limited by $p = 6$, because

$$R = \bigcup_{i,j=0}^{6} \varphi(\delta_1^i \delta_2 \delta_2^j),$$

but the shortest word $w \in \Delta^*$ satisfying $\varphi(w) = R$ is $w = \delta_1^{29}\delta_2$. Note, that $|w| = 30 > 6 + 1 + 6$ and we can not find the desired word by substituting p instead of all star operations in the corresponding expression. □

Proposition 3.3. *Let $M = M_1 S_1^* M_2 S_2^* \ldots M_n S_n^* M_{n+1}$ be a union-free language over Δ, $\varphi : \Delta^+ \to 2^{\Sigma^*}$ be a regular language substitution, and $R \subseteq \Sigma^*$ be a regular language. If $\varepsilon \notin \varphi(S_i)$ for some $i \in [1, n]$, then there exists a natural number k_{max} such that for every word $w \in M$ satisfying $\varphi(w) = R$*

$$w \in M_1 S_1^* \ldots S_{i-1}^* M_i S_i^k M_{i+1} S_{i+1}^* \ldots M_n S_n^* M_{n+1}$$

for some $k \leqslant k_{max}$.

Proof. The length of the shortest word in $\varphi(w)$ grows as we increase k, so k_{max} is bounded by the length of the shortest word in R. □

Summing up Lemma 3.2 and Proposition 3.3 we get Theorem 3.1. The outline of the resulting algorithm is the following. First, for a given rational set $\mathcal{R} = (K, \varphi)$ and a regular language R we construct a union-free decomposition of the rewriting $M_\varphi(R) \cap K$. If this decomposition contains no exact rewritings, then $R \notin \mathcal{R}$. Otherwise, we apply Proposition 3.3 to each exact union-free rewriting and obtain a finite number of union-free languages satisfying conditions of Lemma 3.1. Then, by Lemma 3.2, it is sufficient to check finiteness property of corresponding distance automata.

4 Finiteness of Semigroup and Rational Set

In this section we will consider the finiteness problem for rational set $\mathcal{R} = (K, \varphi)$ of regular languages. This problem has two important specific cases. When $K = \Delta^+$, the finiteness problem for $\mathcal{R}$ is equivalent to the finiteness problem of finitely generated semigroup. If $K = \Delta^+$ and $|\Delta| = 1$ then we have the *finite power property* problem of regular language. Let us consider the latter problem more closely because it plays a crucial role in the following proof.

A regular language L is said to have the *finite power property* if there exists a natural number $k \in \mathbb{N}$ such that $L^* = L^k$. In 1966 J. Brzozowski raised a question whether this property is decidable for regular languages. The positive answer was independently given by K. Hashiguchi [10] and I. Simon [22]. We shall write $fpp(L) = p$ or $fpp(L) < \infty$ if $L^* = L^p$, and $fpp(L) = \infty$ if no such number exists. The following characterizations of regular languages satisfying finite power property are well known (c.f. [20]):

Proposition 4.1. *Let L be a regular language. Then*

1. *if $fpp(L) < \infty$, then either L is infinite and $\varepsilon \in L$, or $L = \{\varepsilon\}$,*
2. *$fpp(L) < \infty$ if and only if there exist $k, n \in \mathbb{N}$, such that $L^k = L^{k+n+1}$,*
3. *$fpp(L) = \infty$ if and only if there exists $w \in L^*$, such that $w^n \notin L^n$ for all $n \in \mathbb{N}$.*

4.1 Finiteness of Free Semigroup

Consider the finiteness problem for the semigroup $(\mathcal{S}, \cdot) = \langle \mathcal{E} \rangle$. If $|\mathcal{E}| = 1$ then finiteness problem is equivalent to the finite power property, and thus decidable. But what can we say if $|\mathcal{E}| > 1$? As we know form semigroup theory [17], a finitely generated semigroup S is finite if and only if there exists a positive number m such that, for any sequence $s_1, \ldots, s_m$ of elements of S, there exist integers $j, k,\ 1 \leqslant j \leqslant k \leqslant m$, such that $s_1 \ldots s_k = s_1 \ldots s_{j-1}(s_j \ldots s_k)^2$. This condition, however, does not provide an upper bound for m and can not be used as a decision procedure.

Proposition 4.2. *Let $(\mathcal{S}, \cdot) = \langle \mathcal{E} \rangle$ be a finitely generated semigroup of regular languages in Σ. The semigroup $\mathcal{S}$ is finite only if $fpp(E) < \infty$ for all $E \in \mathcal{E}$.*

Proof. If $fpp(E) = \infty$ for some $E \in \mathcal{E}$, then all the languages of the form E^k $k \in \mathbb{N}$ are distinct and the semigroup is infinite. □

This condition, however, is not sufficient if $|\mathcal{E}| > 1$. For example, if $E_1 = a^*$ and $E_2 = b^*$, the semigroup is infinite because $(a^*b^*)^* = (a + b)^*$, but for any $p \in \mathbb{N}$ the language $(a^*b^*)^p$ consists of all words with at most p "letter switches". Moreover, it is known [6] that the semigroup given by presentation $S = \langle \Delta \mid x^3 = x^2 \text{ for all } x \in \Delta^* \rangle$ is infinite even if $|\Delta| = 2$.

Existence of the empty word allows to prove the decidability of finiteness problem. Let us setup some useful relations between finite power property and concatenation operation.

Proposition 4.3. *Let $E_1, \ldots, E_k$ be regular languages such that $\varepsilon \in E_i$ for all $i \in [1,k]$. For any permutation σ of natural numbers $1, 2, \ldots, k$*

$$(E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k})^* = (E_1 + E_2 + \ldots + E_k)^*.$$

Proof. Clearly, $(E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k})^* \subseteq (E_1 + E_2 + \ldots + E_k)^*$ for any permutation σ of natural numbers $1, 2, \ldots, k$. We have to prove the inverse inclusion.

Let w be a word in $(E_1 + E_2 + \ldots + E_k)^*$. There exists $p \in \mathbb{N}$ such that $w \in (E_1 + E_2 + \ldots + E_k)^p$, and the word w can be represented as $w = w_1 w_2 \ldots w_p$, where each w_i belongs to E_j for some j. Note, that since $\varepsilon \in E_i$ for all $i \in [1,k]$ then $E_i \subseteq (E_{\sigma_1} \ldots E_{\sigma_k})$. Thus, $w \in (E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k})^{pk}$ and $(E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k})^* \supseteq (E_1 + E_2 + \ldots + E_k)^*$. □

Proposition 4.4. *Let $E_1, \ldots, E_k$ be regular languages, and σ be an arbitrary permutation of natural numbers $1, 2, \ldots, k$. Then,*

$$fpp(E_1 E_2 \ldots E_k) < \infty \Rightarrow fpp(E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k}) < \infty.$$

Proof. Consider the languages $L = (E_{\sigma_1} \ldots E_{\sigma_k})^{pk}$ and $L' = (E_{\sigma_1} \ldots E_{\sigma_k})^{pk+1}$. Since $fpp(E_1 E_2 \ldots E_k) = p < \infty$ the language $E_1 E_2 \ldots E_k$ and all the languages $E_1, \ldots, E_k$ contain the empty word. The following inclusions hold:

$$(E_1 E_2 \ldots E_k)^p \subseteq L \subseteq L' \subseteq (E_1 + \ldots + E_k)^*$$

According to Proposition 4.3, $(E_1 E_2 \ldots E_k)^* = (E_1 + \ldots + E_k)^p$, so $L = L'$ and $fpp(E_{\sigma_1} E_{\sigma_2} \ldots E_{\sigma_k}) < \infty$ by Proposition 4.1. □

Theorem 4.1. *Let $\mathcal{S} = \langle \mathcal{E} \rangle$ be a finitely generated semigroup of regular languages over Σ and $\varphi : \Delta^+ \to \mathcal{S}$ be a choice of generators for $\mathcal{S}$. The semigroup $\mathcal{S}$ is finite if and only if for any m-subset $\{\delta_1, \ldots, \delta_m\} \subseteq \Delta$ $(m = 1, \ldots, |\Delta|)$*

$$fpp(\varphi(\delta_1 \delta_2 \ldots \delta_m)) < \infty.$$

Proof. Clearly, if $fpp(\varphi(\delta_1 \delta_2 \ldots \delta_m)) = \infty$ for some $\{\delta_1, \ldots, \delta_m\} \subseteq \Delta$, then the semigroup $\mathcal{S}$ is infinite. We prove now, by induction on k – the cardinality of Δ, that if the conditions of the theorem hold, then for every k there exists a constant $C_k \in \mathbb{N}$, such that for any word $w \in \Delta^+$ there exists a word $w' \in \Delta^+$ that satisfy $\varphi(w') = \varphi(w)$ and $|w'| \leqslant C_k$.

Let $k = 1$. As we mentioned above, in this case we have finite power property and the statement is correct.

Assume that the statement is correct for $k = n - 1$ and consider the case $k = n$. Let $fpp(\varphi(\delta_1 \delta_2 \ldots \delta_n)) = p$. According to Proposition 4.3 every word $w \in \{\delta_1, \ldots, \delta_n\}^*$ may be presented as:

$$w = w_{11} \delta_1 w_{12} \delta_2 \ldots w_{1n} \delta_n w_{21} \delta_1 w_{22} \delta_2 \ldots w_{2n} \delta_n \ldots\ldots w_{pn} \delta_n \tilde{w}, \qquad (4)$$

where $w_{ij} \in (\Delta \setminus \{\delta_j\})^*$ for all $i = 1, \ldots, p$ and $j = 1, \ldots, n$, and $\tilde{w} \in \{\delta_1, \ldots, \delta_n\}^*$. According to inductive assumption for every word w_{ij} there exists a word w'_{ij}, such that $\varphi(w'_{ij}) = \varphi(w_{ij})$ and $|w'_{ij}| \leqslant C_{n-1}$. Let us assume that $|w_{ij}| \leqslant C_{n-1}$. It is follow from (4) that $\varphi((\delta_1 \delta_2 \ldots \delta_n)^p) \subseteq \varphi(w)$, but $\varphi((\delta_1 \delta_2 \ldots \delta_n)^p) \supseteq \varphi(w)$ by Proposition 4.3. Note, that if $\varepsilon \in \varphi(\delta)$ for all $\delta \in \Delta$, then $\varphi(u) \subseteq \varphi(uv)$ for all $u, v \in \Delta^+$. Thus, we can choose C_n to be equal to $(C_{n-1} + 1)p$. □

The following example demonstrates that there exist non-commutative regular languages satisfying condition of the above theorem, i.e. there exist non-trivial finite semigroups of regular languages.

Example 4.1. Let $\Sigma = \{a, b\}$, $E_1 = b^*a + (b^*ab^*ab^*)^*$, and $E_2 = (b + ab^*a)^*$. The languages E_1 and E_2 satisfy the finite power property and do not commute ($E_1E_2 \neq E_2E_1$). One can verify that $(E_1E_2)^* = (E_1E_2)^2$.

4.2 Finiteness of a Rational Set

Finiteness of a rational set $\mathcal{R} = (K, \varphi)$ is harder to establish than finiteness of the corresponding semigroup $\mathcal{S}_\varphi$, because not all possible products (concatenations) of generators are allowed by K. For instance, let Σ be an unary alphabet, $\Sigma = \{a\}$, and substitution φ is defined as $\varphi(\delta_1) = (aaa)^*$, $\varphi(\delta_2) = (aa + \varepsilon)$, and $\varphi(\delta_3) = (aaaaa)^*$. The semigroup $\mathcal{S}_\varphi$ is, obviously, infinite, but the set $\mathcal{R} = (K, \varphi)$, where $K = \delta_1\delta_2^*\delta_3$ is finite due to equality

$$\varphi(\delta_1)\varphi(\delta_2)^2\varphi(\delta_3) = \varphi(\delta_1)\varphi(\delta_2)^*\varphi(\delta_3).$$

For the lack of simplicity assume that K is a union-free language of star height one, hence,

$$K = K_1S_1^*K_2S_2^* \dots K_mS_m^*K_{m+1}, \tag{5}$$

where $K_i \in \Delta^*$ and $S_i \in \Delta^+$.

Proposition 4.5. *Let $\mathcal{R} = (K, \varphi)$ be a rational set of regular languages in Σ, where K has the form (5). If $\varepsilon \notin \varphi(S_i)$ for some $i \in [1, m]$ then $\mathcal{R}$ is infinite.*

Proof. Similarly to Proposition 3.3 consider the length of the shortest word. □

Define the mapping $\gamma : (\mathbb{N} \cup \{*\})^m \to 2^{\Sigma^*}$ by the rule $\mathbf{v} = (v_1, v_2, \dots, v_m) \mapsto K_1S_1^{v_1} \dots K_mS_m^{v_m}K_{m+1}$. By $d(\mathbf{v})$ denote the bound for distance function of distance Σ-automaton $\mathcal{A}_{\gamma(\mathbf{v})}$. If $\mathcal{A}_{\gamma(\mathbf{v})}$ is not limited, then define $d(\mathbf{v}) = \infty$. For any k-subset $I = \{i_1, \dots, i_k\} \subseteq \{1, \dots, m\}$ and any number $p \in \mathbb{N}$ by $\mathbf{V}(I, p)$ denote the following subset of $(\mathbb{N} \cup \{*\})^m$:

$$\mathbf{V}(I, p) = \{\mathbf{v} \in (\mathbb{N} \cup \{*\})^m : v_i \in [0, p] \text{ if } i \in I, \text{ and } v_i = * \text{ if } i \notin I\} \tag{6}$$

Let $D(I, p) = \max\{d(\mathbf{v}) \mid \mathbf{v} \in \mathbf{V}(I, p)\}$. By p_0 denote $d((*, \dots, *))$. Consider the sequence $p_0, \dots, p_m$ of $\mathbb{N} \cup \infty$ elements, defined for any $k \in [0, m-1]$ as

$$p_{k+1} = \begin{cases} \max\{D(I_{k+1}, p_k) \mid I_{k+1} \subseteq \{1, \dots, m\},\ |I_{k+1}| = k+1\}, & p_k \in \mathbb{N} \\ \infty, & p_k = \infty. \end{cases} \tag{7}$$

Proposition 4.6. *Let $\mathcal{R} = (K, \varphi)$ be a rational set of regular languages over Σ. If $\varepsilon \in \varphi(\delta)$ for all $\delta \in \Delta$, then the set $\mathcal{R}$ is finite if and only if p_m defined by (7) is a natural number.*

Proof. Similarly to the proof of Lemma 3.1. □

Theorem 4.2. *For any rational set $\mathcal{R} = (K, \varphi)$ of regular languages it is decidable whether $\mathcal{R}$ is finite or not.*

5 Conclusion

In this paper we extended the results by K.Hashiguchi [11] and proved that membership and finiteness problems for rational sets of regular languages are decidable. An obvious drawback of the solution proposed is its computational complexity. Limitedness property of a distance automaton has exponential complexity (wrt the number of states of the automaton) and exponential lower bound for maximal rewriting construction was proved in [8]. Note, that even for finite languages only exponential time complexity factorization algorithms are known [21]. A possible solution is a reduction to the membership problem over unary alphabet, which is simpler then the original problem [2]. Consider a language morphism $f : \Sigma \to \{a\}$ and reduce the problem into a one-letter case, when only lengths of words are taken into account. If corresponding problem for one-letter alphabet has no solution then the original problem has no solution either.

We conclude this paper with a discussion on some open problems. First of all, are the *equivalence* and *intersection* problems for rational sets $\mathcal{R}_1 = (K_1, \varphi_1)$ and $\mathcal{R}_2 = (K_2, \varphi_2)$ decidable? That is, is it decidable whether or not $\mathcal{R}_1 = \mathcal{R}_2$ and is the intersection $\mathcal{R} = \mathcal{R}_1 \cap \mathcal{R}_2$ a rational set. The problem is, that even if $\mathcal{R}_1$ and $\mathcal{R}_2$ are equivalent, some of generators of $\mathcal{R}_1$, i.e. the languages $\varphi(\delta),\ \delta \in \Delta_1$, may not be representable in terms of generators for $\mathcal{R}_2$, and vice verse.

Another problem is Δ- and K-minimality of a representation. We say that a representation (K, φ) of a rational set $\mathcal{R}$ is *K-minimal* if $(K', \varphi) \subset \mathcal{R}$ for any regular language $K' \subset K$. Similarly, we say that a representation (K, φ) is *Δ-minimal* if $\mathcal{R}$ has no representation (K', φ') in a smaller alphabet Δ'. Is it decidable whether a given representation is Δ- or K-minimal, and is it possible to compute Δ- or K-minimal representation for a given rational set?

It is worth noting that these problems are interesting from practical point of view. For example, in a view-based query processing system, we are interested in finding a set of views, i.e. $\mathcal{E}$, that covers a given (finite) set $\mathcal{Q}$ of "the most popular" queries, i.e. $\mathcal{Q} \subseteq \mathcal{S} = \langle \mathcal{E} \rangle$. Clearly, we would like to maintain as little views as possible, that leads to Δ-minimality problem for free semigroup.

Finally, it is interesting whether or not similar problems are decidable if the concatenation is replaced by some other binary language operation, such as synchronized insertion or shuffle. Linear language equations in this case remain decidable [12] and one can expect that the so is the membership problem for the finitely generated semigroup.

References

1. Abiteboul, S. Querying semi-structured data. In *Database Theory – ICDT'97, 6th International Conference* (Delphi, Greece, 8–10 Jan. 1997), F. N. Afrati and P. Kolaitis, Eds., vol. 1186 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.
2. Afonin, S., Hazova, E., and Shundeev, A. On regular language factorisation: a complete solution for unary case. Tech. Rep. 252, CDMTCS, Auckland, 2004.

3. Afonin, S., Shundeev, A., and Roganov, V. Semistructured data search using dynamic parallelisation technology. In *Proceedings of the 26th International Convention MIPRO-2003, Opatija, Croatia* (2003), pp. 152–157.
4. Arbib, M. *Algebraic Theory of Machines, Languages, and Semigroups.* Academic Press, 1968.
5. Bala, S. Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. In *Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science STACS-2004* (2004), V. Diekert and M. Habib, Eds., vol. 2996 of *Lecture Notes in Computer Science*, pp. 596–607.
6. Brzozowski, J., Culik, K., and Gabrielian, A. Classification of noncounting events. *Journal of computer and system sciences 5* (1971), 41–53.
7. Brzozowski, J. A., and Cohen, R. On decompositions of regular events. *Journal of the ACM 16*, 1 (Jan. 1969), 132–144.
8. Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M. Rewriting of regular expressions and regular path queries. *Journal of Computer and System Sciences 64* (May 2002), 443–465.
9. Calvanese, D., Giacomo, G. D., Lenzerini, M., and Vardi, M. Y. Answering regular path queries using views. In *ICDE* (2000), pp. 389–398.
10. Hashiguchi, K. Limitedness theorem on finite automata with distance functions. *Journal of computer and system sciences 24* (1982), 233–244.
11. Hashiguchi, K. Representation theorems on regular languages. *Journal of computer and system sciences 27* (1983), 101–115.
12. Kari, L. On language equations with invertible operations. *Theoretical Computer Science 132*, 1–2 (Sept. 1994), 129–150.
13. Kari, L., and Thierrin, G. Maximal and minimal solutions to language equations. *Journal of Computer and System Sciences 53* (December 1996), 487–496.
14. Kirsten, D. Desert automata and the finite substitution problem. In *Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science STACS-2004* (2004), V. Diekert and M. Habib, Eds., vol. 2996 of *Lecture Notes in Computer Science*, pp. 305–316.
15. Leiss, E. *Language Equations.* Springer-Verlag, 1999.
16. Leung, H., and Podolskiy, V. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science 310*, 1–3 (Jan. 2004), 147–158.
17. Luca, K. d., and Revisto, A. A finiteness condition for finitely generated semigroups. *Semigroup forum 28* (1984), 123–134.
18. Mascle, J.-P. Torsion matrix semigroups and recognizable transductions. In *Automata, Languages and Programming* (Berlin, 1986), L. Kott, Ed., vol. 226 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 244–253.
19. Pin, J.-E. Tropical semirings. In *Idempotency* (1998), J. Gunawardena, Ed., Cambridge University Press, pp. 50–69.
20. Salomaa, A. *Jewels of Formal Language Theory.* Computer science press, 1981.
21. Salomaa, A., and Yu, S. On the decomposition of finite languages. In *Proceedings of the Developments of Language Theory 1999* (2000), World Scientific.
22. Simon, I. Limited subsets of a free monoid. In *Proceedings of the 19st Annual Symposium on Foundations of Computer Science* (1978), pp. 143–150.
23. Simon, I. Recognizable sets with multiplicities in the tropical semiring. In *Mathematical Foundations of Computer Science* (1988), M. Chytil, Ed., vol. 324 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 107–120.

Tissue P Systems with Antiport Rules and Small Numbers of Symbols and Cells

Artiom Alhazov[1,2], Rudolf Freund[3], and Marion Oswald[3]

[1] Research Group on Mathematical Linguistics,
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
artiome.alhazov@estudiants.urv.es
[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
artiom@math.md
[3] Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9–11, A-1040 Vienna, Austria
{rudi,marion}@emcc.at

Abstract. We consider tissue P systems with antiport rules and investigate their computational power when using only a (very) small number of symbols and cells. Even when using only one symbol, any recursively enumerable set of natural numbers can be generated with at most seven cells. On the other hand, with only one cell we can only generate regular sets when using one channel with the environment, whereas one cell with two channels between the cell and the environment obtains computational completeness with at most five symbols. Between these extreme cases of one symbol and one cell, respectively, there seems to be a trade-off between the number of cells and the number of symbols, e.g., for the case of tissue P systems with two channels between a cell and the environment we show that computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively.

Keywords: antiport rules, cells, membrane computing, tissue P systems

1 Introduction

Membrane systems with a hierarchical (tree-like) structure (P systems) were introduced in the original paper of Gheorghe Păun (see [11]) with the rules being applied in a maximally parallel manner; tissue P systems with cells arranged in an arbitrary graph structure (see [8]) allow only one rule to be applied in each channel between two cells or a cell and the environment, but all channels work together in a maximally parallel manner. We here consider tissue P systems using symport/antiport rules (these communication rules first were investigated in [10]) for the communication between two cells or a cell and the environment.

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 100–111, 2005.
© Springer-Verlag Berlin Heidelberg 2005

It is well known that equipped with the maximally parallel derivation mode P systems with only one membrane (one cell) already reach computational completeness, even with antiport rules of weight two (e.g., see [3], [5], [7]); the same result also holds true for P systems with one cell and two channels (working in opposite directions) between the cell and the environment, whereas with only one channel between the environment and the single cell only regular sets of natural numbers can be generated as will be proved in Section 5.

Considering the generation of recursively enumerable sets of natural numbers we may also ask the question how many symbols we need for obtaining computational completeness, especially for small numbers of membranes (in P systems) and cells (in tissue P systems), respectively. In [13], the quite surprising result was proved that three symbols and four membranes are enough in the case of P systems with symport/ antiport rules. The specific type of maximally parallel application of at most one rule in each channel between two cells or a cell and the environment, respectively, in tissue P systems allowed for an even more surprising result proved in [4]: The minimal number of one symbol is already sufficient to obtain computational completeness, e.g., it was shown that any recursively enumerable set of natural numbers can be generated by a tissue P system with at most seven cells using symport / antiport rules of only one symbol. On the other hand, for "classical" P systems using symport / antiport rules it was shown in [1] that computational completeness can already be obtained in one membrane by using only five symbols.

We here further investigate the power of tissue P systems with symport / antiport rules as well as small numbers of symbols and cells. After some preliminary definitions, we recall the definition of tissue P systems as they are considered in this paper. We then first consider tissue P systems with only one symbol and recall the completeness result from [4] in Section 3. Afterwards, in Section 4 we elaborate computational completeness results for tissue P systems with two and three symbols, respectively. On the other hand, for tissue P systems with only one cell, computational completeness cannot be achieved when using only one channel with the environment, instead with at least two symbols we obtain only regular sets as is shown in Section 5, whereas one cell with two channels between the cell and the environment gains computational completeness with at most five symbols (a similar result for P systems is established in [1]). In Section 6 we finally give an overview about the results obtained in this paper.

2 Preliminaries

For the basic elements of formal language theory needed in the following, we refer to [2] and [15]. We just list a few notions and notations: $\mathbf{N}$ denotes the set of natural numbers (i.e., of non-negative integers). V^* is the free monoid generated by the alphabet V under the operation of concatenation and the empty string, denoted by λ, as unit element; by NRE, $NREG$, and $NFIN$ we denote the family of recursively enumerable sets, regular sets, and finite sets of natural numbers, respectively.

2.1 Register Machines

The proofs of the main results established in this paper are based on the simulation of register machines; we refer to [9] for original definitions, and to [3] for definitions like those we use in this paper:

A *(non-deterministic) register machine* is a construct $M = (n, R, l_0, l_h)$, where n is the number of registers, R is a finite set of instructions injectively labelled with elements from a given set $lab(M)$, l_0 is the initial/start label, and l_h is the final label.

The instructions are of the following forms:

- $l_1 : (A(r), l_2, l_3)$ – add 1 to the contents of register r and proceed to one of the instructions (labelled with) l_2 and l_3. (We say that we have an ADD instruction.)
- $l_1 : (S(r), l_2, l_3)$ – if register r is not empty, then subtract 1 from its contents and go to instruction l_2, otherwise proceed to instruction l_3. (We say that we have a SUB instruction.)
- l_h : *halt* – stop the machine. The final label l_h is only assigned to this instruction.

A register machine M is said to generate a vector $(s_1, \ldots, s_k)$ of natural numbers if, starting with the instruction with label l_0 and all registers containing the number 0, the machine stops (it reaches the instruction l_h : *halt*) with the first k registers containing the numbers $s_1, \ldots, s_k$ (and all other registers being empty).

Without loss of generality, in the succeeding proofs we will assume that in each ADD instruction $l_1 : (A(r), l_2, l_3)$ and in each SUB instruction $l_1 : (S(r), l_2, l_3)$ the labels l_1, l_2, l_3 are mutually distinct (for a short proof see [6]).

The (non-deterministic) register machines are known to be computationally complete, equal in power to (non-deterministic) Turing machines; especially we know that three registers are enough to generate any recursively enumerable set of natural numbers, where, moreover, the only instructions on the first register are ADD instructions (see [3], [9]).

2.2 Tissue P Systems with Symport / Antiport Rules

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [12]; comprehensive information can be found on the P systems web page `http://psystems.disco.unimib.it`.

Tissue P systems were introduced in [8], and tissue-like P systems with channel states were investigated in [6]. Here we deal with the following type of systems (omitting the channel states):

A tissue P system (of degree $m \geq 1$) with symport / antiport rules is a construct

$$\Pi = \left(m, O, w_1, \ldots, w_m, ch, (R(i,j))_{(i,j) \in ch}\right),$$

where m is the number of cells, O is the alphabet of *objects*, $w_1, \ldots, w_m$ are strings over O representing the *initial* multiset of *objects* present in the cells of

the system (it is assumed that the m cells are labelled with $1, 2, \ldots, m$, and, moreover, we assume that all objects from O appear in an unbounded number in the environment), $ch \subseteq \{(i,j) \mid i,j \in \{0,1,2,\ldots,m\}, (i,j) \neq (0,0)\}$ is the set of links (*channels*) between cells (these were called *synapses* in [6]; 0 indicates the environment), $R(i,j)$ is a finite set of antiport rules of the form x/y, for some $x, y \in O^*$, associated with the channel $(i,j) \in ch$.

An *antiport rule* of the form $x/y \in R(i,j)$ for the ordered pair (i,j) of cells means moving the objects specified by x from cell i (from the environment, if $i = 0$) to cell j, at the same time moving the objects specified by y in the opposite direction. The rules with one of x, y being empty are, in fact, *symport rules*, but we do not always explicitly consider this distinction here, as it is not relevant for what follows. For short, we shall also speak of a *tissue P system* only when dealing with a *tissue P system with symport / antiport rules* as defined above.

The computation starts with the multisets specified by $w_1, \ldots, w_m$ in the m cells; in each time unit, a rule is used on each channel for which a rule can be used (if no rule is applicable for a channel, then no object passes over it). Therefore, the use of rules is sequential at the level of each channel, but it is parallel at the level of the system: all channels which can use a rule must do it (the system is synchronously evolving). The computation is successful if and only if it halts.

The result of a halting computation is the number described by the multiplicity of objects present in cell 1 in the halting configuration. The set of all natural numbers computed in this way by the system Π is denoted by $N(\Pi)$. The family of sets $N(\Pi)$ of natural numbers computed as above by systems with at most n symbols and m cells is denoted by $NO_ntP'_m$. When any of the parameters m, n is not bounded, it is replaced by $*$.

In [6], only channels (i,j) with $i \neq j$ are allowed, and, moreover, for any i, j only one channel out of $\{(i,j),(j,i)\}$ is allowed, i.e., between two cells (or one cell and the environment) only one channel is allowed (as we shall see in Section 5, this technical detail may influence considerably the computational power). The family of sets $N(\Pi)$ of vectors computed as above by such tissue P systems with at most n symbols and at most m cells is denoted by NO_ntP_m.

In the following we will not distinguish between a language $L \subseteq \{a\}^*$ and the corresponding set of natural numbers $Ps(L) = \{k \mid a^k \in L\}$, the Parikh set of L.

3 Computational Completeness with One Symbol

In [4] it was shown that one symbol is enough for obtaining computational completeness when using at least seven cells:

Theorem 1. *$NRE = NO_1tP_n$ for all $n \geq 7$.*

Omitting the condition that for any i, j only one channel out of $\{(i,j),(j,i)\}$ is allowed, at least one cell can be saved (i.e., the one used as the trap, see [4]):

Theorem 2. *$NRE = NO_1tP'_n$ for all $n \geq 6$.*

4 Computational Completeness with Two and Three Symbols

As we are going to prove in this section, there seems to be a trade-off between the number of cells and the number of symbols: as our main result, we show that in the case of allowing two channels between a cell and the environment (we can restrict ourselves to only one channel between cells) computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively. We first show that when allowing only two symbols we need at most three cells for obtaining computational completeness:

Theorem 3. *$NRE = NO_n tP'_m$ for all $n \geq 2$ and $m \geq 3$.*

Proof. We only prove $NRE \subseteq NO_2 tP'_3$.

Let us consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; we now construct the tissue P system (of degree 3)

$$\Pi = \left(3, \{a, b\}, w_1, \lambda, \lambda, ch, \left(R\left(i,j\right)\right)_{(i,j)\in ch}\right),$$
$$ch = \{(0,1), (0,2), (0,3), (1,0), (1,2), (2,0), (2,3), (3,0), (3,1)\},$$

which simulates the actions of M in such a way that Π halts if and only if M halts, thereby representing the final contents of register 1 of M by the corresponding multisets of symbols in the first cell (and no other symbols contained there). Throughout the computation, cell i of Π represents the contents of register i by the corresponding number of symbols a, whereas specific numbers of the symbol b represent the instructions to be simulated; moreover, b also has the function of a trap symbol, i.e., in case of the wrong choice for a rule to be applied we take in so many symbols b that we can never again get rid of them and therefore get "trapped" in an infinite loop.

An important part of the proof is to define a suitable encoding c for the instructions of the register machine: Without loss of generality we assume the labels of M to be positive integers such that the labels assigned to ADD and SUB instructions have the values $di + 1$ for $0 \leq i < t - 1$, as well as $l_0 = 1$ and $l_h = d(t-1) + 1$, for some $t > 1$ and some constant $d > 1$ which allows us to have d consecutive codes for each instruction. As we shall see, in this proof it suffices to take $d = 7$.

We now define the encoding c on natural numbers in such a way that $c : \mathbf{N} \to \mathbf{N}$ is a linear function that has to obey to the following additional conditions:

- For any i, j with $1 \leq i, j < dt$, $c(i) + c(j) > c(dt)$, i.e., the sum of the codes of two instruction labels has to be larger than the largest code $c(dt)$ we will ever use for the given M.
- The distance g between any two codes $c(i)$ and $c(i+1)$ has to be large enough to allow one copy of the symbol b to be used for appearance checking as well as to allow specific numbers between 1 and g of copies of b to detect an incorrect application of rules.

As we shall see in the construction of the rules below, we may take $g = 2$. A function c fulfilling all the conditions stated above then, for example, is

$$c(x) = gx + gdt = 2x + 14t \text{ for } x \geq 0.$$

With $l_0 = 1$ we therefore obtain $c(l_0) = 14t + 2$ and $w_1 = b^{c(l_0)} = b^{14t+2}$.

Finally, we have to find a number f which is so large that after the introduction of f symbols we inevitably enter an infinite loop with the rule b^{2f}/b^2; as we shall see below, we can take $f = 2c(dt)$.

Equipped with this coding function c and the constants defined above we are now able to define the following sets of symport / antiport rules for simulating the actions of the given register machine M:

$$\begin{aligned}
R_{(0,1)} &= R_{(0,2)} = R_{(0,3)} = \left\{b^{2f}/b^2\right\},\\
R_{(1,0)} &= \left\{b^{c(l_1)}/b^{c(l_2)}a, b^{c(l_1)}/b^{c(l_3)}a \mid l_1 : (A(1), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_1+1)}a, b^{c(l_1+2)}/b^{c(l_2)}, b^{c(l_1+2)}/b^{c(l_3)} \mid\right.\\
&\quad\quad \left. l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)}/b^{c(l_1+3)}, b^{c(l_1+4)}/b^{c(l_3)},\right.\\
&\quad\quad b^{c(l_1)}/b^{c(l_1+5)}, b^{c(l_1+6)}/b^{c(l_2)} \mid\\
&\quad\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \cup \left\{b^{c(l_h)}/\lambda\right\},\\
R_{(1,2)} &= \left\{b^{c(l_1+1)}a/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\}\\
&\quad \cup \left\{b^{c(l_1+1)+1}/\lambda, b^{c(l_1+3)}/\lambda, b^{c(l_1+5)}/\lambda \mid\right.\\
&\quad\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\},\\
R_{(2,0)} &= \left\{ba/b^{2f}\right\},\\
R_{(2,3)} &= \left\{b^{c(l_1+1)}/\lambda \mid l_1 : (A(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}a/\lambda \mid l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}/\lambda, b^{c(l_1+3)+1}/\lambda, b^{c(l_1+5)}a/\lambda \mid l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)+1}/\lambda, b^{c(l_1+3)}/\lambda, b^{c(l_1+5)}/\lambda \mid l_1 : (S(3), l_2, l_3) \in R\right\},\\
R_{(3,0)} &= \left\{b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}/b^{c(l_1+2)}, b^{c(l_1+3)+1}/b^{c(l_1+4)},\right.\\
&\quad\quad \left. b^{c(l_1+5)}a/b^{c(l_1+6)} \mid l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\}\\
&\quad \cup \left\{ba/b^{2f}\right\},\\
R_{(3,1)} &= \left\{b^{c(l_1+2)}/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\}\\
&\quad \cup \left\{b^{c(l_1+2)}/\lambda, b^{c(l_1+4)}/\lambda, b^{c(l_1+6)}/\lambda \mid\right.\\
&\quad\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\}.
\end{aligned}$$

The correct work of the rules in Π can be described as follows:

1. Throughout the whole computation in Π, the application of rules is directed by the code $b^{c(l)}$ for some $l \leq l_h$; in order to guarantee the correct sequence of encoded rules, superfluous symbols b in case of a wrong choice guarantee an infinite loop with the symbols b by the "trap rule" b^{2f}/b^2 in the rule sets $R_{(0,i)}$, $i \in \{1,2,3\}$.
   The number $2f$ is so large that even in cell 1 which allows for the elimination of $c(l_h)$ symbols b enough symbols b remain to repeat the "trap rule" b^{2f}/b^2.

2. Each ADD instruction $l_1 : (A(1), l_2, l_3)$ of M is directly simulated by the rules
$$b^{c(l_1)}/b^{c(l_2)}a \text{ or } b^{c(l_1)}/b^{c(l_3)}a$$
in $R_{(1,0)}$ in one step. The ADD instructions $l_1 : (A(r), l_2, l_3)$ of M, $r \in \{2, 3\}$, are simulated in six steps in such a way that the new symbol a is transported to the corresponding cell r and, moreover, in cell 3 the code $c(l_1)$ is exchanged with the code $c(l_1 + 1)$ in order to guarantee that when returning to cell 1 a different code arrives which does not allow for misusing a symbol a representing the contents of register 1 in cell 1 to start a new cycle with the original code. For example, the simulation of an ADD instruction $l_1 : (A(2), l_2, l_3)$ is accomplished by applying the following sequence of rules:
$$b^{c(l_1)}/b^{c(l_1+1)}a \text{ from } R_{(1,0)},$$
$$b^{c(l_1+1)}a/\lambda \text{ from } R_{(1,2)},$$
$$b^{c(l_1+1)}/\lambda \text{ from } R_{(2,3)},$$
$$b^{c(l_1+1)}/b^{c(l_1+2)} \text{ from } R_{(3,0)},$$
$$b^{c(l_1+2)}/\lambda \text{ from } R_{(3,1)},$$
$$b^{c(l_1+2)}/b^{c(l_2)} \text{ or } b^{c(l_1+2)}/b^{c(l_3)} \text{ from } R_{(1,0)}.$$
If we do not choose one of the correct rules, then an infinite loop will be entered by the rule b^{2f}/b^2 from the rule sets $R_{(0,i)}$, $i \in \{1, 2, 3\}$, as the coding function has been chosen in such a way that instead of the correct rule for a label l only rules for labels $l' < l$ could be chosen, whereas on the other hand, the number of symbols b is not large enough for allowing the remaining rest being interpreted as the code of another instruction label.
3. For simulating the decrementing step of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $b^{c(l_1+5)}$ to the corresponding cell r, where a correct continuation is only possible if this cell contains at least one symbol a. For example, decrementing register 2 is accomplished by applying the following sequence of six rules:
$$b^{c(l_1)}/b^{c(l_1+5)} \text{ from } R_{(1,0)},$$
$$b^{c(l_1+5)}/\lambda \text{ from } R_{(1,2)},$$
$$b^{c(l_1+5)}a/\lambda \text{ from } R_{(2,3)},$$
$$b^{c(l_1+5)}a/b^{c(l_1+6)} \text{ from } R_{(3,0)},$$
$$b^{c(l_1+6)}/\lambda \text{ from } R_{(3,1)},$$
$$b^{c(l_1+6)}/b^{c(l_2)} \text{ from } R_{(1,0)}.$$
Again we notice that if at some moment we do not choose the correct rule, then the application of the rule b^{2f}/b^2 will cause an infinite loop.
4. For simulating the zero test, i.e., the case where the contents of register r is zero, of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $b^{c(l_1+1)}$ together with one additional copy of the symbol b to cell r, where this additional symbol b may cause the application of the rule ba/b^{2f} which then will lead to an infinite computation. In cell 3, the code $b^{c(l_1+1)}$ is exchanged with the code $b^{c(l_1+2)}$, which is exchanged with $b^{c(l_1+3)}$ in cell 1. This code $b^{c(l_1+3)}$ then captures the additional symbol b left back in cell r, and in cell

3 this additional symbol b goes out together with code $b^{c(l_1+3)}$, instead, code $b^{c(l_1+4)}$ continues and in cell 1 allows for replacing it with $b^{c(l_2)}$. For example, for testing register 3 for zero we take the following rules:

$b^{c(l_1)}/b^{c(l_1+1)+1}$ from $R_{(1,0)}$,
$b^{c(l_1+1)+1}/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+1)+1}/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+1)}/b^{c(l_1+2)}$ from $R_{(3,0)}$,
$b^{c(l_1+2)}/\lambda$ from $R_{(3,1)}$,
$b^{c(l_1+2)}/b^{c(l_1+3)}$ from $R_{(1,0)}$,
$b^{c(l_1+3)}/\lambda$ from $R_{(1,2)}$,
$b^{c(l_1+3)}/\lambda$ from $R_{(2,3)}$,
$b^{c(l_1+3)+1}/b^{c(l_1+4)}$ from $R_{(3,0)}$,
$b^{c(l_1+4)}/\lambda$ from $R_{(3,1)}$,
$b^{c(l_1+4)}/b^{c(l_3)}$ from $R_{(1,0)}$.

Once again we notice that if at some moment we do not choose the correct rule, then the application of the rule b^{2f}/b^2 will cause an infinite loop.

5. Finally, for the halt label l_h we take the rule $b^{c(l_h)}/\lambda$ from $R_{(1,0)}$, hence, the work of Π will stop exactly when the work of M stops (provided the system has not become overflowed by symbols b due to a wrong non-deterministic choice during the computation).

From the explanations given above we conclude that Π halts if and only if M halts, and moreover, the final configuration of Π represents the final contents of the registers in M (i.e., the number of symbols in cell 1 corresponds with the number generated in register 1). These observations conclude the proof. □

Instead of the second channels $(0,i)$, $i \in \{1,2,3\}$, between the cells and the environment we could also use a fourth cell which then acts as a trap:

Corollary 1. *$NRE = NO_n tP_m$ for all $n \geq 2$ and $m \geq 4$.*

Proof. We only prove $NRE \subseteq NO_2 tP_4$.

Let us consider the tissue P system (of degree 4)

$$\Pi' = \left(4, \{a,b\}, w_1, \lambda, \lambda, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(1,0),(1,2),(2,0),(2,3),(3,0),(3,1),$$
$$(4,0),(4,1),(4,2),(4,3)\},$$

where w_1 and the rule sets $R(i,j)$ for

$$(i,j) \in \{(1,0),(1,2),(2,0),(2,3),(3,0),(3,1)\}$$

are exactly the same as in the proof of Theorem 3; the additional sets $R(4,j)$ for $j \in \{1,2,3\}$ contain the "trap rule" λ/b^2 which starts the trap represented by the rule b/b in $R(4,0)$. Except for the way of trapping the system, Π' works

in the same way as the system Π constructed in the proof of Theorem 3, which observation completes the proof. □

On the other hand, when the number of objects is increased to three, we need one cell less:

Theorem 4. *$NRE = NO_n tP'_m$ for all $n \geq 3$ and $m \geq 2$.*

Proof. We only prove $NRE \subseteq NO_3 tP'_2$.

As in the previous proof, we consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; we now construct the tissue P system (of degree 2)

$$\Pi = \left(2, \{a, b, c\}, w_1, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(0,1), (0,2), (1,0), (1,2), (2,0)\},$$

which simulates the actions of M in such a way that throughout the computation, specific numbers of the symbol b represent the instructions to be simulated, the number of symbols a in cell 1 of Π represents the contents of register 1 by the corresponding number of symbols a and the new symbol c represents the contents of registers 2 and 3 by the corresponding number of symbols c in cells 1 and 2, respectively.

We shall use the same encoding c and the same initial string w_1 as in the previous proof. The zero test now uses the rule bc/b^{2f} in $R_{(1,0)}$ or $R_{(2,0)}$, respectively.

In sum, we define the following sets of symport / antiport rules for simulating the actions of the given register machine M:

$$\begin{aligned}
R_{(0,1)} &= R_{(0,2)} = \left\{b^{2f}/b^2\right\},\\
R_{(1,0)} &= \left\{b^{c(l_1)}/b^{c(l_2)}a, b^{c(l_1)}/b^{c(l_3)}a \mid l_1 : (A(1), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_2)}c, b^{c(l_1)}/b^{c(l_3)}c \mid l_1 : (A(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_1+1)}c, b^{c(l_1+2)}/b^{c(l_2)}, b^{c(l_1+2)}/b^{c(l_3)} \mid\right.\\
&\qquad \left. l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)+1}/b^{c(l_3)}, b^{c(l_1)}c/b^{c(l_2)} \mid\right.\\
&\qquad \left. l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1)}/b^{c(l_1+1)+1}, b^{c(l_1+2)}/b^{c(l_1+3)}, b^{c(l_1+4)}/b^{c(l_3)},\right.\\
&\qquad b^{c(l_1)}/b^{c(l_1+5)}, b^{c(l_1+6)}/b^{c(l_2)} \mid\\
&\qquad \left. l_1 : (S(3), l_2, l_3) \in R\right\} \cup \left\{b^{c(l_h)}/\lambda, bc/b^{2f}\right\},\\
R_{(1,2)} &= \left\{b^{c(l_1+1)}c/\lambda, \lambda/b^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}/\lambda, \lambda/b^{c(l_1+2)} \mid l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)+1}/\lambda, \lambda/b^{c(l_1+2)}, b^{c(l_1+3)}/\lambda,\right.\\
&\qquad \lambda/b^{c(l_1+4)}, b^{c(l_1+5)}/\lambda, \lambda/b^{c(l_1+6)} \mid\\
&\qquad \left. l_1 : (S(3), l_2, l_3) \in R\right\} \cup \left\{b^4/\lambda, \lambda/b^2\right\},\\
R_{(2,0)} &= \left\{b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}/b^{c(l_1+2)} \mid l_1 : (S(2), l_2, l_3) \in R\right\}\\
&\quad \cup \left\{b^{c(l_1+1)}/b^{c(l_1+2)}, b^{c(l_1+3)+1}/b^{c(l_1+4)},\right.\\
&\qquad \left. b^{c(l_1+5)}c/b^{c(l_1+6)} \mid l_1 : (S(3), l_2, l_3) \in R\right\} \cup \left\{bc/b^{2f}\right\}.
\end{aligned}$$

As one can easily see, Π halts if and only if M halts, and moreover, in the final configuration of Π cell 1 represents the final contents of register 1 in M. If at some moment we do not use the correct rule, then an infinite loop will be entered by applying the rule b^{2f}/b^2 from the rule sets $R_{(0,i)}$, $i \in \{1,2\}$. These observations conclude the proof. □

The next result follows from Theorem 4 in a similar way as the proof of Corollary 1 followed from the proof of Theorem 3; hence, we omit the proof:

Corollary 2. *$NRE = NO_ntP_m$ for all $n \geq 3$ and $m \geq 3$.*

5 Tissue P Systems with One Cell

In this section we investigate the remaining variant of using only one cell, in which case it turns out that the definition of the tissue P system is essential, i.e., computational completeness can only be obtained with two channels between the cell and the environment, whereas we can only generate regular sets when using only one channel between the cell and the environment.

The proof of the following completeness result can be obtained following the construction given in [1] for P systems and therefore is omitted:

Theorem 5. *$NRE = NO_ntP_1'$ for all $n \geq 5$.*

We now consider the case of tissue P systems with only one channel between the cell and the environment. In the simplest case of only one symbol, we only get finite sets:

Example 1. To each finite one-letter language L we can construct the tissue P system $\Pi = (1, \{a\}, w_1, \{(1,0)\}, R(1,0))$ where $w_1 = a^m$ with $m = \max\{i \mid a^i \in L\}$ and $R(1,0) = \{a^m/a^j \mid a^j \in L, j < m\}$. Obviously, $Ps(L) = N(\Pi)$.

The proof of the following theorem is obvious and therefore omitted:

Theorem 6. *$NFIN = NO_1tP_1 = NO_1tP_1'$.*

The following result shows that with only one cell and one channel between the single cell and the environment only regular sets can be generated (due to lack of space we have to omit the proof):

Theorem 7. *$NREG = NO_ntP_1$ for all $n \geq 2$.*

6 Conclusion

In sum, for tissue P systems with only one channel between two cells and between a cell and the environment we could show the results listed in Table 1 (we have omitted the proof of the completeness result $NRE = NO_ntP_m$ for all $n \geq 4$ and

Table 1. Families NO_mtP_n

symbols							
4	$NREG$	NRE	NRE	NRE	NRE	NRE	NRE
3	$NREG$	?	NRE	NRE	NRE	NRE	NRE
2	$NREG$	?	?	NRE	NRE	NRE	NRE
1	$NFIN$	?	?	?	?	?	NRE
	1	2	3	4	5	6	7 cells

$m \geq 2$, which needs a different proof technique following the construction given in [1] for P systems).

The main open question concerns a characterization of the sets of natural numbers in NO_2tP_2, NO_2tP_3, and NO_3tP_2. Further, it would be interesting to find the minimal number l such that NO_1tP_l contains all recursively enumerable sets of natural numbers, whereas the families NO_1tP_j with $j < l$ do not fulfill this condition. Finally, it remains to find characterizations of the sets of natural numbers in those families NO_1tP_j that do not contain all recursively enumerable sets of natural numbers.

Table 2. Families $NO_mtP'_n$

symbols						
5	NRE	NRE	NRE	NRE	NRE	NRE
4	?	NRE	NRE	NRE	NRE	NRE
3	?	NRE	NRE	NRE	NRE	NRE
2	?	?	NRE	NRE	NRE	NRE
1	$NFIN$	?	?	?	?	NRE
	1	2	3	4	5	6 cells

The most interesting open problems for the families $NO_ntP'_m$ are to find the minimal number k as well as the minimal number l such that $NO_ktP'_1$ and $NO_1tP'_l$, respectively, contain all recursively enumerable sets of natural numbers, whereas the families $NO_itP'_1$ and $NO_1tP'_j$ with $i < k$ and $j < l$, respectively, do not fulfill this condition. Moreover, it remains to find characterizations of the sets of natural numbers in those families $NO_ntP'_m$ that do not contain all recursively enumerable sets of natural numbers.

Related open problems concern the families NO_mP_n of sets of natural numbers generated by P systems with symport / antiport rules as well as n symbols and m membranes. The first result proving computational completeness for P systems with three symbols and four membranes was obtained in [13] and continued in [1], where P systems with five symbols in only one membrane were shown to be computationally complete. The main open problem in the case of P systems is the question whether one symbol is sufficient to obtain computational completeness as was shown for the case of tissue P systems in [4].

Acknowledgement

The work of Marion Oswald was supported by FWF-project T225-N04.

References

1. Alhazov, A., Freund, R.: P systems with one membrane and symport/ antiport rules of five symbols are computationally complete. To appear in the Proceedings of the Third Brainstorming Week on Membrane Computing, Sevilla (2005)
2. Dassow, J., Păun, Gh.: Regulated Rewriting in Formal Language Theory. Springer-Verlag, Berlin (1989)
3. Freund, R., Oswald, M.: P Systems with activated/prohibited membrane channels. In [14] (2003) 261–268
4. Freund, R., Oswald, M.: Tissue P systems with symport / antiport rules of one symbol are computationally complete. Downloadable from [16] (2004)
5. Freund, R., Păun, A.: Membrane systems with symport/ antiport rules: universality results. In [14] (2003) 270–287
6. Freund, R., Păun, Gh., Pérez-Jiménez, M.J.: Tissue-like P systems with channel states. Second Brainstorming Week on Membrane Computing, Sevilla, February 2004, TR 01/04 of Research Group on Natural Computing, Sevilla University (2004) 206–223 and Theoretical Computer Science 330 (2005) 101–116
7. Frisco, P., Hoogeboom, H.J.: Simulating counter automata by P systems with symport/antiport. In [14] (2003) 288–301
8. Martín-Vide, C., Pazos, J., Păun, Gh., Rodríguez-Patón, A.: Tissue P systems. Theoretical Computer Science **296** (2) (2003) 295–326
9. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice Hall, Englewood Cliffs, New Jersey, USA (1967)
10. Păun, A., Păun, Gh.: The power of communication: P systems with symport/ antiport. New Generation Computing **20** (3) (2002) 295–306
11. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences **61** (1) (2000) 108–143 and
TUCS Research Report 208 (1998) (`http://www.tucs.fi`)
12. Păun, Gh.: Membrane Computing: An Introduction. Springer-Verlag, Berlin (2002)
13. Păun, Gh., Pazos, J., Pérez-Jiménez, M.J., Rodríguez-Patón, A.: Symport/ antiport P systems with three objects are universal. Downloadable from [16] (2004)
14. Păun, Gh., Rozenberg, G., Salomaa, A., Zandron C. (Eds.): Membrane Computing. International Workshop WMC 2002, Curtea de Argeş, Romania, Revised Papers. Lecture Notes in Computer Science **2597**, Springer-Verlag, Berlin (2003)
15. Rozenberg, G., Salomaa, A. (Eds.): Handbook of Formal Languages (3 volumes). Springer-Verlag, Berlin (1997)
16. The P Systems Web Page: `http://psystems.disco.unimib.it`

The Mortality Threshold for Partially Monotonic Automata

Dmitry S. Ananichev*

Ural State University,
620083 Ekaterinburg, Russia
Dmitry.Ananichev@usu.ru

Abstract. A deterministic incomplete automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is partially monotonic if its state set Q admits a linear order such that each partial transformation $\delta(\_, a)$ with $a \in \Sigma$ preserves the restriction of the order to the domain of the transformation. We show that if $\mathscr{A}$ possesses a 'killer' word $w \in \Sigma^*$ whose action is nowhere defined, then $\mathscr{A}$ is 'killed' by a word of length $|Q| + \left\lfloor \frac{|Q|-1}{2} \right\rfloor$.

1 Background and Motivation

A *deterministic incomplete automaton* $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is defined by specifying a finite *state set* Q, a finite *input alphabet* Σ, and a partial *transition function* $\delta : Q \times \Sigma \to Q$. The partial function δ extends in a unique way to a partial action $Q \times \Sigma^* \to Q$ of the free monoid Σ^* over Σ; this extension is still denoted by δ. Thus, each word $w \in \Sigma^*$ defines a partial transformation of the set Q denoted by $\delta(\_, w)$.

Given a deterministic incomplete automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, it may happen that for some word $w \in \Sigma^*$ the partial transformation $\delta(\_, w)$ is nowhere defined. The intuition is that the automaton $\mathscr{A}$ breaks when being fed with the input w; we therefore say that the word w *kills* $\mathscr{A}$.

There are several rather natural questions concerning the notion of a killer word: how *mortal* automata (that is, incomplete automata possessing a killer word) can be recognized, how long a killer word for a given mortal automaton may be, etc. These questions are tightly related to the synchronization problems for complete automata with 0. Recall that a deterministic automaton is said to be *complete* if its transition function is totally defined. A complete automaton $\mathscr{A} = \langle Q, \Sigma, \zeta\rangle$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter at which state in Q it started: $\zeta(q, w) = \zeta(q', w)$ for all $q, q' \in Q$. Any word w with this property is said to be a *reset word* for the automaton.

Given a complete automaton $\mathscr{A} = \langle Q, \Sigma, \zeta\rangle$, we say that $s \in Q$ is a *sink state* if $\zeta(s, a) = s$ for all $a \in \Sigma$. It is clear that any synchronizing automaton may

* The work was supported by the Federal Education Agency of Russia, grant 49123 and 04.01.437, the President Program of Leading Scientific Schools, grant 2227.2003.1, and the Russian Foundation for Basic Research, grant 05-01-00540

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 112–121, 2005.
© Springer-Verlag Berlin Heidelberg 2005

have at most one sink state and any word that resets a synchronizing automaton possessing a sink state brings all states to the sink. In such a situation, we denote the unique sink state by 0 and refer to the automaton as a *synchronizing automaton with* 0.

Every incomplete automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ can be completed in the following obvious way. First one adds a new state 0 to the state set Q; let Q^0 stand for the resulting set $Q \cup \{0\}$. Then one extends the partial function $\delta : Q \times \Sigma \to Q$ to a total function $\zeta : Q^0 \times \Sigma \to Q^0$ setting for all $p \in Q^0$ and all $a \in \Sigma$

$$\zeta(p,a) = \begin{cases} \delta(p,a) & \text{if } \delta(p,a) \text{ is defined,} \\ 0 & \text{otherwise.} \end{cases}$$

We call the automaton $\langle Q^0, \Sigma, \zeta\rangle$ the 0-*completion* of the incomplete automaton $\mathscr{A}$ and denote this completion by $\mathscr{A}^0$. It is then clear that the 0-completion of a mortal automaton is a synchronizing automaton with 0, and vice versa, every mortal automaton can be obtained from a synchronizing automaton with 0 by removing the zero state and all arrows leading to it.

Recall that the general problem of determining the *synchronization threshold* (that is, the maximum length of the shortest reset word) for a synchronizing automaton with a given number n of states still remains open. (The famous *Černý conjecture* [3] claiming that this threshold is equal to $(n-1)^2$ is arguably the most longstanding open problem in the combinatorial theory of finite automata.) In contrast, the restriction of this problem to the case of synchronizing automata with 0 admits an easy solution: the synchronization threshold for n-state synchronizing automata with 0 is known to be equal to $\frac{n(n-1)}{2}$ (see, for instance, [5, Theorem 6.1]). Applying this result to 0-completions of mortal automata, one readily obtains the value $\frac{n(n+1)}{2}$ for the *mortality threshold* (that is, the maximum length of the shortest killer word) for mortal automata with n states. However, the situation becomes much more intricate for the important subcase of aperiodic automata.

Recall that a deterministic finite automaton $\mathscr{A}$ (complete or not) is said to be *aperiodic* if all subgroups of its transition monoid are singletons. (In view of a celebrated theorem of Schützenberger [6] this amounts to saying that $\mathscr{A}$ can recognize only star-free languages.) It is to be expected that for mortal aperiodic automata with n states the mortality threshold is smaller than in the general case but up to now no bound better than $\frac{n(n+1)}{2}$ has been found. On the other hand, all known examples of mortal aperiodic automata possess short killer words so that it even was conjectured that such an automaton always can be killed by a word whose length does not exceed the number of states of the automaton, see a discussion in [2, Section 3].

Determining the mortality threshold for aperiodic automata is especially important in view of recent results on synchronization of aperiodic automata due to Trahtman [7] and Volkov (unpublished). Without going into detail, we mention that the problem of finding the synchronization threshold for arbitrary synchronizing aperiodic automata can be easily reduced to the cases of strongly connected automata and synchronizing automata with 0. The currently known

upper bound of the synchronization threshold for strongly connected aperiodic automata with n states is $\lfloor \frac{n(n+1)}{6} \rfloor$ which is less than $\frac{n(n-1)}{2}$ for all $n > 2$. Therefore any improved upper bound of the synchronization threshold for synchronizing aperiodic automata with 0 will immediately yield a corresponding improvement for arbitrary synchronizing aperiodic automata.

In the present paper we study incomplete automata of a special kind which we call partially monotonic. A deterministic incomplete automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *partially monotonic* if its state set Q admits a linear order such that each partial transformation $\delta(\_, a)$ with $a \in \Sigma$ preserves the restriction of the order to the domain of the transformation. This means that for all $q, q' \in Q$ such that $q \leq q'$ and both $\delta(q, a)$ and $\delta(q', a)$ are defined, one has $\delta(q, a) \leq \delta(q', a)$. It is well known and easy to verify that each partially monotonic automaton is aperiodic (but the converse, generally speaking, is not true). Our main result gives a linear upper bound for the mortality threshold for partially monotonic automata:

Theorem 1. *If a partially monotonic automaton with n states is mortal, then it has a killing word of length at most $n + \lfloor \frac{n-1}{2} \rfloor$.*

We also present a series of examples of mortal partially monotonic automata showing that this bound is tight for $n \geq 6$. The proof that the n^{th} automaton in the series cannot be killed by a shorter word is quite long and therefore is not included here due to length limitations. This proof will be published elsewhere.

Our proof of Theorem 1 is based on a careful analysis of certain properties of **complete** monotonic automata which for brevity will be called *monotonic* in the sequel. (The term 'monotonic automaton' was also used in [4] but in a different sense.) Basic synchronization properties of monotonic automata have been already studied in [1] but here we need some refinements of the results of that paper.

Throughout the paper we assume that the state set Q of automata under consideration is the set $\{1, 2, 3, \ldots, n\}$ of the first n positive integers with the usual order $1 < 2 < 3 < \cdots < n$.

2 Strongly Connected Monotonic Automata

We call a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *strongly connected* if the graph of $\mathscr{A}$ is strongly connected, that is, for any states $p, q \in Q$ there is a word $w \in \Sigma^*$ such that $\delta(p, w) = q$.

We need the following property of strongly connected monotonic automata.

Lemma 1. *Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a strongly connected monotonic automaton. Then for any state $q \in Q \setminus \{1\}$ there is a letter $a \in \Sigma$ such that $\delta(q, a) < q$ and for any state $q \in Q \setminus \{n\}$ there is a letter $b \in \Sigma$ such that $\delta(q, b) > q$.*

Proof. Consider a word w such that $\delta(q, w) = 1 < q$. (Such a word w exists because the automaton $\mathscr{A}$ is strongly connected.) We find the shortest prefix u of the word w such that $\delta(q, u) < q$ and take the last letter of u as a.

By the choice of the letter a we see that $u = va$, $p = \delta(q, v) \geq q$ and $\delta(p, a) < q$. The automaton $\mathscr{A}$ is monotonic whence $\delta(q, a) \leq \delta(p, a) < q$.

The letter b with $\delta(q, b) > q$ can be found in a similar way. □

Let X be a subset of the state set of the automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$. We define the *full preimage of degree* k of the set X as

$$P^k(X) = \{q \in Q \mid (\exists w \in \Sigma^*)\ |w| \leq k \text{ and } \delta(q, w) \in X\}.$$

Let $P(X) = P^1(X)$. Observe that $P^k(X) \subseteq P^{k+1}(X)$ for any k. Moreover, if the automaton $\mathscr{A}$ is strongly connected then the equality $P^k(X) = P^{k+1}(X)$ implies that $P^k(X) = Q$. Therefore, the unequality $P^k(X) \neq Q$ implies that $|P^k(X)| \geq |X| + k$.

Let $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ be a strongly connected monotonic automaton. We say that a subset X of the set Q is a *special set* if $|P(X)\setminus X| = 1$ and $X \cap \{1, n\} = \varnothing$.

Our next lemma points out a useful property of some special sets.

Lemma 2. *Let $p \in Q$ and let $X = P^k(\{p\})$ be a special set in a strongly connected monotonic automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$. Then there is a word $w \in \Sigma^*$ of length at most $n - 2 - k$ such that either $\delta(1, w) = n$ or $\delta(n, w) = 1$.*

Proof. Let q denote the unique state of the difference $P(X) \setminus X$. We construct two sequences $q = q_0 > q_1 > \cdots > q_\alpha = 1$ and $q = p_0 < p_1 < \cdots < p_\beta = n$ in the following way. Using Lemma 1 for each $i \in \{0, 1, 2, \ldots, \alpha - 1\}$ we find a letter $a_i \in \Sigma$ such that $\delta(q_i, a_i) < q_i$. We denote $\delta(q_i, a_i)$ by q_{i+1}. Also for each $j \in \{0, 1, 2, \ldots, \beta - 1\}$ we find a letter $b_j \in \Sigma$ such that $p_{j+1} = \delta(p_j, b_j) > p_j$.

Let $f = \begin{cases} \min\{i | q_i \notin P(X)\} & \text{if } q_0 \neq 1, \\ 0 & \text{if } q_0 = 1, \end{cases}$

and $g = \begin{cases} \min\{i | p_i \notin P(X)\} & \text{if } p_0 \neq n, \\ 0 & \text{if } p_0 = n. \end{cases}$

Then $\{p_1, p_2, \ldots, p_{g-1}, q_1, q_2, \ldots, q_{f-1}\} \subseteq X$, whence $|X| \geq f + g - 2$.

Now we assume that $f \leq g$ and using Lemma 1 find a word w such that $\delta(n, w) = 1$. Then we show that the length of this word is at most $n - 2 - k$. By symmetry, in case $g \geq f$ we find a word w of length at most $n - 2 - k$ such that $\delta(1, w) = n$.

By Lemma 1 there are the chain $n = r_1 > r_2 > \cdots > r_{s+1} = 1$ and a word $w = c_1 c_2 \cdots c_s$ such that $\delta(r_i, c_i) = r_{i+1}$ for each $i \in \{1, 2, \ldots, s\}$. If the intersection $\{r_1, r_2, \ldots, r_{s+1}\} \cap X$ is empty, then

$$|w| = s \leq n - 1 - |X| = n - |P^k(\{p\})| - 1 \leq n - k - 2.$$

In the opposite case we consider the first element r_h of this chain that lies in $P(X)$. Observe that $r_h = q$. Indeed, if $r_h \neq q$, then $r_h \in X$ whence $r_{h-1} \in P(X)$.

The ways of constructing the chains $r_1 > r_2 > \cdots > r_{s+1}$ and $q_0 > q_1 > \cdots > q_\alpha$ are identical and $r_h = q_0$. Therefore we can take $c_h = a_0$ and obtain $r_{h+1} = q_1$, take $c_{h+1} = a_1$ and obtain $r_{h+2} = q_2$ and so on. Finally, we take $c_{h+f-1} = a_{f-1}$ and obtain $r_{h+f} = q_f$. Observe that $r_i \notin P(X)$ for any $i \geq h + f$. Indeed, if we consider the first element $r_i \in P(X)$ for $i \geq h + f$ we come to a contradiction

because as above $r_i = q = q_0 > q_f = r_{h+f}$ but $i \geq h + f$ implies $r_i \leq r_{h+f}$. Summarizing, we have obtained that $\{r_1, r_2, \ldots, r_{s+1}\} \cap P(X) = \{q_0, q_1, \ldots q_{f-1}\}$ whence $|w| = s \leq n - 1 - |P(X)| + f = n - 2 + (f - |X|)$.

To complete the proof it suffices to show that $f - |X| \leq -k$. Observe that if $p_i \in P^u(\{p\})$ for some $i > 0$ and $u \leq k$, then $p_{i-1} \in P^{u+1}(\{p\})$. Therefore $p_i \in P^{k-f+1}(\{p\})$ for some $i \in \{1, 2, \ldots, f-1\}$ contradicts to $p_0 \notin P^k(\{p\})$, whence $p_i \notin P^{k-f+1}(\{p\})$ for $i \in \{1, 2, \ldots, f-1\}$ and, similarly, $q_i \notin P^{k-f+1}(\{p\})$ for $i \in \{1, 2, \ldots, f-1\}$. It means that $\{p_1, p_2, \ldots, p_{f-1}, q_1, q_2, \ldots, q_{f-1}\} \subseteq X \setminus P^{k-f+1}(\{p\})$. Hence $2f - 2 \leq |X| - |P^{k-f+1}(\{p\})| \leq |X| - (k - f + 2)$, and therefore, $f - |X| \leq -k$. □

The next fact is crucial for the proof of our main result.

Proposition 1. *Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a strongly connected monotonic automaton. Given a state $p \in Q$, there is a word w of length at most $\frac{3}{2}(n-1)$ such that $\delta(q, w) = p$ for all $q \in Q$.*

Proof. Consider the chain of the full preimages of the set $\{p\}$:

$$\{p\} = P^0(\{p\}) \subseteq P^1(\{p\}) \subseteq P^2(\{p\}) \subseteq \cdots \subseteq P^t(\{p\}).$$

Let $Y = P^s(\{p\})$ be the minimal set of this chain containing 1 or n. We suppose that $1 \in Y$. The case $n \in Y$ is symmetric.

If there is no special sets in this chain, then $|P^i(\{p\}) \setminus P^{i-1}(\{p\})| \geq 2$ for each $i \in \{1, ..., s\}$, hence $2s + 1 \leq |Y| \leq |Q| = n$. By Lemma 1 there is a word u of length at most $n-1$ such that $\delta(n, u) = 1$. Since the automaton $\mathscr{A}$ is monotonic, $\delta(q, u) = 1$ for any $q \in Q$. The inclusion $1 \in Y = P^s(\{p\})$ implies that there is a word v of length at most $s \leq \frac{1}{2}(n-1)$ such that $\delta(1, v) = p$. If $w = uv$ then $|w| \leq \frac{3}{2}(n-1)$ and $\delta(q, w) = p$ for any $q \in Q$.

Now let $X = P^k(\{p\})$ be the maximal special set of the chain.

By the choice of k and s we obtain that $|Y| - |P(X)| \geq 2(s - k - 1)$, hence $|Y| \geq |P(X)| + 2(s-k-1) = |P^{k+1}(\{p\})| + 2(s-k-1) \geq k + 2 + 2(s-k-1) = 2s - k$.

By Lemma 2 there is a word $u \in \Sigma^*$ of length at most $n - 2 - k$ such that either $\delta(1, u) = n$ or $\delta(n, u) = 1$.

Case 1: $\delta(1, u) = n$.

Since the automaton $\mathscr{A}$ is monotonic, $\delta(q, u) = n$ for any $q \in Q$. Since the automaton $\mathscr{A}$ is strongly connected, there is a word u_1 of length at most $n - |Y|$ such that $\delta(n, u_1) \in Y$. By the definition of $Y = P^s(\{p\})$ there is a word v_1 of length at most s such that $\delta(\delta(n, u_1), v_1) = p$. Denote uu_1v_1 by w_1. Observe that $|w_1| = |u| + |u_1| + |v_1| \leq (n-2-k) + (n-|Y|) + s \leq (n-2-k) + (n-2s+k) + s = 2(n-1) - s$ and $\delta(q, w_1) = p$ for any $q \in Q$.

Furthermore, by Lemma 1 there is a word u_2 of length at most $n-1$ such that $\delta(n, u_2) = 1$. Since the automaton $\mathscr{A}$ is monotonic, $\delta(q, u_2) = 1$ for any $q \in Q$. The inclusion $1 \in Y = P^s(\{p\})$ implies that there is a word v_2 of length at most s such that $\delta(1, v_2) = p$. Denote u_2v_2 by w_2. It is easy to see that $|w_2| \leq n - 1 + s$ and $\delta(q, w_2) = p$ for any $q \in Q$.

Let w be the shortest word in the pair w_1, w_2. The inequality $n - 1 + s \leq 2(n-1) - s$ implies that $s \leq \frac{1}{2}(n-1)$, therefore $|w| \leq \frac{3}{2}(n-1)$.

Case 2: $\delta(n,u)=1$.

The inclusion $1 \in Y = P^s(\{p\})$ implies that there is a word v of length at most s such that $\delta(1,v)=p$. Let $w=uv$. It is easy to see that $|w| \leq n-k-2+s$ and $\delta(q,w)=p$ for any $q \in Q$. Observe that $-k \leq 0$ and $s-k \leq |Y|-s \leq n-s$, therefore $|w| \leq \min\{n+s-2, 2n-s-2\} \leq \frac{3}{2}n-2 < \frac{3}{2}(n-1)$. □

Next propositions describe the structure of monotonic automata. We use these statements to generalize Proposition 1 to the set of all synchronizing monotonic automata and to prove Theorem 1.

We need some definitions to formulate the propositions.

Given a word $w \in \Sigma^*$ and non-empty subset $X \subseteq Q$, we write $X.w$ for the set $\{\delta(x,w) \mid x \in X\}$ Given an automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, we define the *rank* of a word $w \in \Sigma^*$ with respect to $\mathscr{A}$ as the cardinality of the image of the transformation $\delta(\_, w)$ of the set Q that is $|Q.w|$. (Thus, in this terminology reset words are precisely words of rank 1.) Define the *rank* $r(\mathscr{A})$ of an automaton $\mathscr{A}$ as the minimum rank of words with respect to $\mathscr{A}$. (Thus, synchronizing automata are precisely automata of rank 1.)

A subset X of a set Q is said to be *invariant with respect to a transformation* $\varphi : Q \to Q$ if $X\varphi \subseteq X$. A subset of the state set of an automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is called *invariant* if it is invariant with respect to all the transformations $\delta(\_, a)$ with $a \in \Sigma$. If X is an invariant subset, we define the *restriction* of $\mathscr{A}$ to X as the automaton $\mathscr{A}_X = \langle X, \Sigma, \delta_X\rangle$ where δ_X is the restriction of the transition function δ to the set $X \times \Sigma$.

To prove the next propositions we use some constructions and arguments taken from the proof of Theorem 1 in [1]. In particular, the next two lemmas coincide with respectively Lemmas 1 and 2 in [1].

Lemma 3. *Let X be a non-empty subset of Q such that $\max(X.w) \leq \max(X)$ for some $w \in \Sigma^*$. Then for each $p \in [\max(X.w), \max(X)]$ there exists a word $\mathcal{D}(X,w,p)$ of length at most $\max(X)-p$ such that $\max(X.\mathcal{D}(X,w,p)) \leq p$.*

The dual statement is

Lemma 4. *Let X be a non-empty subset of Q such that $\min(X.w) \geq \min(X)$ for some $w \in \Sigma^*$. Then for each $p \in [\min(X), \min(X.w)]$ there exists a word $\mathcal{U}(X,w,p)$ of length at most $p-\min(X)$ such that $\min(X.\mathcal{U}(X,w,p)) \geq p$.*

For $x, y \in Q$ with $x \leq y$ we denote by $[x,y]$ the interval $\{x, x+1, x+2, \ldots, y\}$.

Proposition 2. *Let $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ be a synchronizing monotonic automaton. Then there is an invariant subset P of the set Q such that the restriction $\mathscr{A}_P$ is a strongly connected automaton, and there is a word $w_{\mathscr{A}}$ of length at most $|Q|-|P|$ such that $\delta(q, w_{\mathscr{A}}) \in P$ for any $q \in Q$.*

Proof. Let $P = \{q \mid \{q\} = Q.w \text{ for some } w \in \Sigma^*\}$. In other words, P is the set of all states to which the automaton $\mathscr{A}$ can be reset. Observe that the set P is invariant. Indeed, arguing by contradiction, suppose that there are $q \in P$ and $w \in \Sigma^*$ such that $\delta(q,w) \notin P$. By the definition of P there is a word u such that

$Q.u = \{q\}$. Hence $Q.uw$ is an one-element set $\{\delta(q, w)\}$. It contradicts to the definition of P. The automaton $\mathscr{A}_P$ is strongly connected because for any states $p, q \in P$ there is a word $w \in \Sigma^*$ such that $Q.w = \{q\}$, in particular, $\delta(p, w) = q$.

Consider the interval $I_1 = [\min(P), n]$. It is invariant. Indeed, arguing by contradiction, suppose that there are $q \in I$ and $w \in \Sigma^*$ such that $\delta(q, w) \notin I$. Since the transformation $\delta(\_, w)$ is order preserving, $\delta(\min(P), w) \leq \delta(q, w) < \min(P)$. That is $\delta(\min(P), w) \notin P$. It contradicts to the fact that P is an invariant set. Similarly, we obtain the dual fact that the interval $I_2 = [1, \max(P)]$ is invariant. Therefore the intersection $I = I_1 \cap I_2$ is also invariant. We can apply Lemma 4 to the set I_1, the state 1 and a synchronizing word $u \in \Sigma^*$. Let $w_1 = \mathcal{U}(I_1, u, 1)$; then the length of w_1 is at most $|Q| - |I_1|$ and $\delta(1, w_1) \in I_1$. Since the transformation $\delta(\_, w_1)$ is order preserving, it means that $Q.w_1 \subseteq I_1$. By symmetry, we can find a word w_2 of length at most $|Q| - |I_2|$ such that $Q.w_2 \subseteq I_2$. The concatenation $w_1 w_2$ has the length at most $|Q| - |I|$. Since the interval $|I_1|$ is invariant, $Q.w_1 w_2 \subseteq I_1 \cap I_2 = I$.

Now fix a synchronizing word v such that $Q.v = \max(P)$. Let $|I \setminus P| = s$. We enumerate the elements $q_1, q_2, \ldots, q_s$ of the difference $I \setminus P$ such that $q_1 < q_2 < q_3 < \cdots < q_s$.

We prove that for each $k \in \{0, 1, \ldots, s\}$ there is a word u_k of length at most k such that $I.u_k \subseteq \{q_{k+1}, q_{k+2}, \ldots, q_s\} \cup P$. We induct on k with the obvious base $k = 0$ (u_0 is the empty word). Let $k > 0$. We find the last letter a in v such that $v = v_1 a v_2$, $\delta(\max(P), v_1) \leq q_k$ and $\delta(\max(P), v_1 a) \geq q_k$. Observe that $\delta(\max(P), v_1 a) \neq q_k$, because the set P is invariant. Since the transformation $\delta(\_, a)$ is order preserving, it means that $\delta(q_i, a) > q_k$ for all $i \in \{k, k+1, \ldots, s\}$. Let $u_k = u_{k-1} a$. By the induction assumption $I.u_{k-1} \subseteq \{q_k, q_{k+1}, \ldots, q_s\} \cup P$. Hence $I.u_k \subseteq (\{q_k, q_{k+1}, \ldots, q_s\} \cup P).a \subseteq \{q_{k+1}, q_{k+2}, \ldots, q_s\} \cup P$ as required.

We obtain that $Q.w_1 w_2 u_s \subseteq I.u_s \subseteq P$ and the length of the word $w_1 w_2 u_s$ is at most $|Q| - |I| + |I \setminus P| = |Q| - |P|$. Thus, the word $w_1 w_2 u_s$ can be chosen to play the role of $w_{\mathscr{A}}$ from the formulation of the proposition. □

Now we generalize Proposition 2 to arbitrary monotonic automaton.

Proposition 3. *Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a monotonic automaton of rank k. Then there are pairwise disjoint invariant subsets $P_1, P_2, \ldots, P_k$ of the set Q such that all restrictions $\mathscr{A}_{P_i}$ for $i \in \{1, \ldots, k\}$ are strongly connected automata, and there is a word $w_{\mathscr{A}}$ of length at most $|Q| - |\bigcup_{i=1}^{k} P_i|$ such that $\delta(q, w_{\mathscr{A}}) \in \bigcup_{i=1}^{k} P_i$ for any $q \in Q$.*

Proof. Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a monotonic automaton of rank k. We induct on k.

If $k = 1$, then the automaton $\mathscr{A}$ is synchronizing and we can apply Proposition 2.

Let $k > 1$. Consider the set $X = \{\min(Q.w) \mid w \in \Sigma^*,\ |Q.w| = k\}$. (This set is not empty because by the condition of the proposition there exists a word of rank k with respect to $\mathscr{A}$.)

Let $M = \max(X)$ and let $v \in \Sigma^*$ be such that $\min(Q.v) = M$ and $|Q.v| = k$. Observe that the interval $Y = [1, M]$ is invariant. Indeed, arguing by contradiction, suppose that there are $q \in Y$ and $w \in \Sigma^*$ such that $\delta(q, w) > M$. Since the transformation $\delta(\_, w)$ is order preserving, $\min(Q.vw) = \delta(M, w) \geq \delta(q, w) >$

M. At the same time, $|Q.vw| \leq |Q.v| \leq k$ whence $\min(Q.vw)$ belongs to the set X (since the rank of $\mathscr{A}$ is k, the unequality $|Q.wv| \leq k$ means that $|Q.wv| = k$). It contradicts to the choice of M. Observe that the restriction $\mathscr{A}_Y$ is a synchronizing automaton. Indeed, there is a word $v \in \Sigma^*$ such that $\min(Q.v) = M$ (and $|Q.v| = k$) by the choice of M but the interval $Y = [1, M]$ is invariant whence $Y.v = \{M\}$.

Consider the set $Z = \{q \in Q \mid \delta(q, w) \leq M \text{ for some } w \in \Sigma^*\}$. Observe that Z is an interval and that $Y \subseteq Z$ since for $q \in Y$ the empty word can serve as w with $\delta(q, w) \leq M$. Therefore $\max(Z) \geq M$. Fix a word $u \in \Sigma^*$ such that $\delta(\max(Z), u) \leq M$. Then $\delta(q, u) \leq M$ for each $q \in Z$ as the transformation $\delta(\_, u)$ is order preserving.

Now consider the interval $T = [\max(Z)+1, n] = Q \setminus Z$. It is invariant. Indeed, suppose that there exist $q \in T$ and $w \in \Sigma^*$ such that $\delta(q, w) \leq \max(Z)$. This means that $\delta(q, wu) \leq M$ whence $q \in Z$, in a contradiction to the choice of q.

Let v_1 be a word of minimal rank (k) with respect to the automaton $\mathscr{A}$, v_2 is a word of minimal rank with respect to the automaton $\mathscr{A}_T$, v_3 is a reset word for the automaton $\mathscr{A}_Y$. Denote $v_1v_2v_3$ by v. Then the word uv also has rank k with respect to $\mathscr{A}$, has a minimal rank with respect to $\mathscr{A}_T$ and the word v resets $\mathscr{A}_Y$. We have $Q.uv = Z.uv \cup T.uv$ and $Z.uv \subseteq Y.v$. Since $Y \subseteq Z$ and $|Y.v| = 1$ we obtain that $Z.uv = Y.v$. The sets Y and T are invariant, therefore $k = |Q.uv| = |Y.v| + |T.uv| = 1 + |T.uv|$. Hence the rank of the restriction $\mathscr{A}_T$ is $k - 1$.

By the induction assumption there are pairwise disjoint invariant subsets $P_1, P_2, \ldots, P_{k-1}$ of the set T and an invariant subset P_k of the set Y such that all automata $\mathscr{A}_{P_i}$ for $i \in \{1, \ldots, k\}$ are strongly connected. There is a word $w_1 = w_{\mathscr{A}_T}$ of length at most $|T| - |\bigcup_{i=1}^{k-1} P_i|$ such that $T.w_1 \subseteq \bigcup_{i=1}^{k-1} P_i$. There is a word $w_2 = w_{\mathscr{A}_Y}$ of length at most $|Y| - |P_k|$ such that $Y.w_2 \subseteq P_k$.

We apply Lemma 3 to the set Y, the state $\max(Z)$ and the word u that was fixed above. Let $w_3 = \mathcal{D}(Y, u, \max(Z))$; then the length of w_3 is at most $|Z| - |Y| = |Q| - |T| - |Y|$ and $Z.w_3 \subseteq Y$. Since T is an invariant set we obtain that $Q.w_3 \subseteq Y \cup T$.

We obtain that $Q.w_3w_1w_2 \subseteq (Y \cup T)w_1w_2 \subseteq \bigcup_{i=1}^{k} P_i$ and the length of the word $w_3w_1w_2$ is at most $|Q| - |\bigcup_{i=1}^{k} P_i|$. Thus, the word $w_3w_1w_2$ can be chosen to play the role of $w_{\mathscr{A}}$ from the formulation of the proposition. □

3 Proof of Theorem 1

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a mortal partially monotonic automaton. Then any $\delta(\_, a)$ with $a \in \Sigma$ is an order preserving partial transformation on the set Q. Each order preserving partial transformation can be extended to the whole set Q so that the resulting total transformation remains order preserving. Do such completions (in an arbitrary way) for all transformations $\delta(\_, a)$ with $a \in \Sigma$. Then we obtain a monotonic automaton $\mathscr{B} = \langle Q, \Sigma, \zeta \rangle$.

We mark every element $q \in Q$ such that $\delta(q, a)$ is not defined for some letter $a \in \Sigma$. Observe that each invariant set of the automaton $\mathscr{B}$ contains a marked element. Indeed, in the opposite case we can find an invariant set I without

marked elements. It means that the restriction $\mathscr{A}_I$ is a complete automaton, hence $\delta(q,w)$ is defined for any $q \in I$ and $w \in \Sigma^*$. It contradicts to the assumption that the automaton $\mathscr{A}$ is mortal.

Let k be a rank of the automaton $\mathscr{B}$. We apply Proposition 3 to this automaton. We find the pairwise disjoint invariant subsets $P_1, P_2, \ldots, P_k$ of the set Q such that all automata $\mathscr{B}_{P_i}$ for $i \in \{1,\ldots,k\}$ are strongly connected and the word $w_{\mathscr{B}}$ of length at most $|Q| - |\bigcup_{i=1}^k P_i|$ such that $\zeta(q, w_{\mathscr{B}}) \in \bigcup_{i=1}^k P_i$ for any $q \in Q$. Observe that all sets $P_1, P_2, \ldots, P_k$ are 'half-invariant' in the automaton $\mathscr{A}$ in the following sense: there is no elements $x \in P_i$ and $y \notin P_i$ such that $\delta(x,w) = y$ for some word $w \in \Sigma^*$.

For each $i \in \{1,\ldots,k\}$ we apply Proposition 1 to the automaton $\mathscr{B}_{P_i}$ and a marked element $p_i \in P_i$. We find a word w_i of length at most $\frac{3}{2}(|P_i|-1)$ such that $\zeta(q, w_i) = p_i$ for any $q \in P_i$. Since the state p_i is marked there is a letter $a_i \in \Sigma$ such that $\delta(p_i, a_i)$ is not defined.

Consider the word $v = w_{\mathscr{B}} w_1 a_1 w_2 a_2 \cdots w_k a_k$. For any $q \in Q$ we then have $\zeta(q, w_{\mathscr{B}}) \in P_i$ for some $i \in \{1,\ldots,k\}$. Hence $\zeta(q, w_{\mathscr{B}} w_1 a_1 \ldots w_i) = p_i$. It means that either $\delta(q, w_{\mathscr{B}} w_1 a_1 \ldots w_i) = p_i$ or $\delta(q, w_{\mathscr{B}} w_1 a_1 \ldots w_i)$ is not defined. In both cases $\delta(q, w_{\mathscr{B}} w_1 a_1 \ldots w_i a_i)$ is not defined. Therefore $\delta(q,v)$ is not defined for any $q \in Q$. Thus, v is a killer word for the automaton $\mathscr{A}$.

The length of the word v is at most

$$
\begin{aligned}
|Q| - |\bigcup_{i=1}^k P_i| + \sum_{i=1}^k \left(\frac{3}{2}(|P_i|-1)+1\right) & \\
= |Q| - |\bigcup_{i=1}^k P_i| + \sum_{i=1}^k |P_i| + \sum_{i=1}^k \left(\frac{1}{2}(|P_i|-1)\right) & \\
= |Q| + \sum_{i=1}^k \left(\frac{1}{2}(|P_i|-1)\right) & \\
= |Q| - \frac{k}{2} + \frac{1}{2}\sum_{i=1}^k |P_i| \le |Q| - \frac{1}{2} + \frac{1}{2}|Q| & \\
= |Q| + \frac{1}{2}(|Q|-1), &
\end{aligned}
$$

whence the length of the word v is at most $|Q| + \lfloor (|Q|-1)/2 \rfloor$ as required. The theorem is proved.

4 The Tightness of the Bound

We present a series of examples of partially monotonic automata $\mathscr{A}_n = \langle Q, \Sigma, \delta \rangle$, where $n = 6, 7, \ldots$, such that the automaton $\mathscr{A}_n$ is killed by a word of length $|Q| + \lfloor (|Q|-1)/2 \rfloor$ but is not killed by any shorter word. The state set Q_n of the automaton $\mathscr{A}_n$ is the chain $1 < 2 < 3 < \cdots < n$. We denote $\lfloor (n+1)/2 \rfloor$ by ℓ. The input alphabet Σ of $\mathscr{A}_n$ contains two letters a and b.

The action of the letter a on the set Q_n is defined as follows:

$$\delta(j,a) = \begin{cases} 1 & \text{if } j = 1, \\ \ell & \text{if } j = \ell + 2, \\ \text{is not defined} & \text{if } j = \ell + 1, \\ j - 1 & \text{in all other cases.} \end{cases}$$

The action of the letter b is defined as follows:

$$\delta(j,b) = \begin{cases} n & \text{if } j = n, \\ j + 1 & \text{in all other cases.} \end{cases}$$

The following picture shows the automaton $\mathscr{A}_9$.

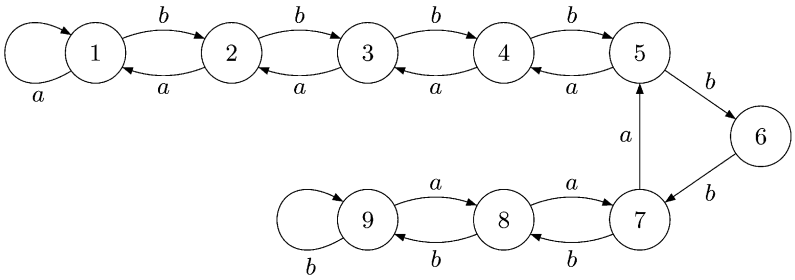


Fig. 1. The automaton $\mathscr{A}_9$

It is easy to verify that the automata $\mathscr{A}_n$ are indeed partially monotonic and that the word $w_n = a^{n-2}b^\ell a$ kills the automaton $\mathscr{A}_n$ for each n. The proof that the automaton $\mathscr{A}_n$ has no killer word of length less than $n + \ell - 1$ is far too long to be reproduced here.

Acknowledgments

Several useful comments of the anonymous referees of this paper are gratefully acknowledged.

References

1. D. S. Ananichev, M. V. Volkov, *Synchronizing monotonic automata*, Theoret. Comput. Sci. **327** (2004) 225–239.
2. D. S. Ananichev, M. V. Volkov, *Synchronizing generalized monotonic automata*, Theoret. Comput. Sci. **330** (2005) 3–13.
3. J. Černý, *Poznámka k homogénnym eksperimentom s konecnými automatami*, Mat.-Fyz. Cas. Slovensk. Akad. Vied. **14** (1964) 208–216 [in Slovak].
4. D. Eppstein, *Reset sequences for monotonic automata*, SIAM J. Comput. **19** (1990) 500–510.
5. I. Rystsov, *Reset words for commutative and solvable automata*, Theoret. Comput. Sci. **172** (1997) 273–279.
6. M. P. Schützenberger, *On finite monoids having only trivial subgroups*, Inf. Control **8** (1965) 190–194.
7. A. N. Trahtman, *Černý conjecture for DFA accepting star-free languages*, submitted.

Sturmian Words: Dynamical Systems and Derivated Words

Isabel M. Araújo[1,2] and Véronique Bruyère[3]

[1] Departamento de Matemática, Universidade de Évora,
Rua Romão Ramalho, 59, 7000–671 Évora, Portugal
[2] Centro de Álgebra da Universidade de Lisboa,
Avenida Professor Gama Pinto, 2, 1649–003 Lisboa, Portugal
[3] Institut d'Informatique, Université de Mons–Hainaut,
Avenue du Champ de Mars, 6, 7000 Mons, Belgium

Abstract. In a preceding article, we have studied the family of words derivated from characteristic Sturmian words. This study has lead to a new proof of the characterization of characteristic Sturmian words which are fixed points of morphisms. In this article, we extend this approach to all Sturmian words. The Sturmian words viewed as dynamical systems play an important role in obtaining this generalization.

1 Introduction

The concepts of return words and derivated words were introduced independently by Durand in [8], and by Holton and Zamboni in [9]. Given a word x and a factor w of x, a *return word* of w in x is a word that starts at an occurrence of w and ends exactly before the next occurrence of w. The derivated word $D_w(x)$ encodes the unique decomposition of x in terms of the return words of w.

These concepts have been recently investigated for the family of Sturmian words. Vuillon has proved in [14] that each factor w of a Sturmian word has exactly two return words, and that this property characterizes Sturmian words. In [2] and [3], we have given the exact form of the two return words and the related derivated words. This study has been limited to the family of Sturmian words which are characteristic. That permitted to answer negatively a question posed by Michaux et Villemaire in [13], and to propose a new proof for the characterization of characteristic words which are fixed points of morphisms given in [7]. In [10], Justin and Vuillon have studied the return words in Sturmian and episturmian words.

In this article, we extend our study to the family of all Sturmian words. In Section 2, we recall some basic notions on Sturmian words, in particular the dynamical system $\mathcal{S}_\alpha$ formed by all the Sturmian words with slope α. We introduce in Section 3 the notions of return words and derivated words in the context of Sturmian words. We recall the results obtained in [2], [3] for characteristic words. Among these results, let us mention that words derivated from a characteristic word are again characteristic, and that a characteristic word is a fixed point of

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 122–133, 2005.
© Springer-Verlag Berlin Heidelberg 2005

a morphism if and only if it can be derivated from itself. In Section 4, we show that if the characteristic word s_α with slope α has a derivated word equal to the characteristic word s_β with slope β, i.e., $D_w(s_\alpha) = s_\beta$, then the derivation operator D_w defines a map from the dynamical system $\mathcal{S}_\alpha$ onto the dynamical system $\mathcal{S}_\beta$. Conversely if a morphism φ verifies $\varphi(s_\beta) = s_\alpha$, then it defines a map from $\mathcal{S}_\beta$ into $\mathcal{S}_\alpha$ which is in a certain sense the inverse image of some derivation operator D_w. In Section 5, we state and prove adequate generalizations of the two theorems given in [3]. For instance, the characterization of characteristic words which are fixed points of morphisms is generalized to all Sturmian words in the following way. Given a Sturmian word x with slope α, there exists a morphism φ such that $\varphi(x)$ has the same slope α if and only if there exists a word $D_w(x)$ derivated from x with slope α.

2 Sturmian Words

In this section we recall some basic notions on Sturmian words. The interested reader is referred to Chapter 2 in [12] and the survey [5] for more details. Reference [1] also contains two chapters on the subject.

There are several ways to define Sturmian words. We begin with the definition of particular Sturmian words which are the characteristic words. Let α be an irrational number such that $0 < \alpha < 1$, and let $[0, a_1 + 1, a_2, \ldots, a_n, \ldots]$ be its continued fraction expansion ($a_1 \geq 0$ and $a_n \geq 1$ for all $n \geq 2$). Then the *characteristic* word s_α with *slope* α is defined by

$$s_\alpha = \lim_{n\to\infty} t_n$$

where the finite words t_n are inductively defined by

$$t_0 = 0,\ t_1 = 0^{a_1}1,\ t_n = t_{n-1}^{a_n} t_{n-2} \quad (n \geq 2).$$

The sequence $(t_n)_n$ is called the *characteristic sequence* associated with s_α [1]. It is usual to define $t_{-1} = 1$ in a way to write $t_1 = t_0^{a_1} t_{-1}$. We also associate to s_α the sequence $(q_n)_n$ of the lengths of the words t_n. Clearly $(q_n)_n$ is given by $q_0 = 1$, $q_1 = a_1 + 1$, and $q_n = a_n q_{n-1} + q_{n-2} (n \geq 2)$.

Let us now introduce the notion of Sturmian words. We begin with some useful notations : $\mathrm{Pref}(x)$ (resp. $\mathrm{Fact}(x)$) is the set of prefixes (resp. factors) of a word $x \in \{0,1\}^\omega$. These notations are naturally extended to subsets $X \subseteq \{0,1\}^\omega$. Consider

$$\mathcal{S}_\alpha = \{x \in \{0,1\}^\omega \mid \mathrm{Fact}(x) = \mathrm{Fact}(s_\alpha)\}$$

the *dynamical system* generated by the characteristic word s_α of slope α. Every element of $\mathcal{S}_\alpha$ is called a *Sturmian* word of *slope* α. The dynamical system $\mathcal{S}_\alpha$ can also be defined as the topological closure of the set

$$R_\alpha = \{S^i(s_\alpha) \mid i \geq 0\},$$

[1] Notice that the first letter of s_α depends on the value a_1 in the continued fraction expansion $[0, a_1 + 1, a_2, \ldots]$ of α. Indeed, it is equal to 0 if $a_1 > 0$, and equal to 1 if $a_1 = 0$

where S is the shift, and the closure of R_α is equal to $\{x \in \{0,1\}^\omega \mid \mathrm{Pref}(x) \subseteq \mathrm{Pref}(R_\alpha)\}$. The following property can be proved.

Proposition 1. *Let $x, y \in \{0,1\}^\omega$. If x is a Sturmian word of slope α and $\mathrm{Fact}(y) \subseteq \mathrm{Fact}(x)$, then y is a Sturmian word. Moreover $\mathrm{Fact}(y) = \mathrm{Fact}(x)$ and y has slope α.*

Example 1. Consider the characteristic word s_α, where α has the continued fraction expansion $[0, 3, 2, 3, 2, 3, 2, \ldots]$. The first elements of the characteristic sequence are $t_0 = 0$, $t_1 = 001$ and $t_2 = 0010010$. The word

$$0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0$$

is a prefix of s_α. The set R_α is formed by all the suffixes of s_α. It is known that $0s_\alpha, 1s_\alpha \in \mathcal{S}_\alpha$ (see [12]). Each of these words belong to $\mathcal{S}_\alpha \setminus R_\alpha$, otherwise they are periodic, in contradiction with the irrationality of α.

3 Return Words and Derivated Words

In this section, we recall the notions of return word and derivated word in the context of Sturmian words. These notions were first defined by Durand in [8], and Holton and Zamboni in [9].

Given a non empty factor w of a Sturmian word $x = x_0x_1 \cdots x_n \cdots$ (with $x_n \in \{0,1\}$ for all n), an integer i is said to be an *occurrence* of w in x if $w = x_i x_{i+1} \cdots x_{i+|w|-1}$. For adjacent occurrences i, j of w in x, the word $x_i x_{i+1} \cdots x_{j-1}$ is call a *return word* of w in x. A return word of w has either w as a prefix, or is a prefix of w (the latter happens when the two occurrences of w overlap).

In [14], Vuillon has showed that an infinite word $x \in \{0,1\}^\omega$ is Sturmian if and only if each non empty factor of x has exactly two return words (see [10] for a shorter proof). In [2] and [3], we gave the precise form of these two return words in the particular case of a prefix w of a characteristic word s_α. Their form is related to the characteristic sequence $(t_n)_n$ associated with s_α and depends on the length of the prefix w. The return words of characteristic Sturmian and episturmian words have been also studied in [10], in terms of their palindromic prefixes (instead of the sequence $(t_n)_n$).

Proposition 2. [2, 3] *Let s_α be a characteristic word with slope $\alpha = [0, a_1 + 1, a_2, \ldots]$. Let $w \in \mathrm{Pref}(s_\alpha)$ be such that $|w|$ belongs to the interval $]iq_n + q_{n-1} - 2, (i+1)q_n + q_{n-1} - 2]$, with $n \geq 1$ and $i \in \{0, \ldots, a_{n+1} - 1\}$, or $n = 0$ and $i \in \{1, \ldots, a_{n+1} - 1\}$. Then the two return words of w are t_n and $t_n^i t_{n-1}$.*

Notice that the results of Proposition 2 naturally extend to all Sturmian words in the following way. We recall that a factor w of a Sturmian word x is *left special* if $0w$ and $1w$ are factors of x. We denote by $\mathrm{Left}(x)$ the set of left special factors of x. It is well-known that x has exactly one left special factor of each length. Moreover when x is characteristic, $\mathrm{Left}(x) = \mathrm{Pref}(x)$. Remember

that Sturmian words that belong to the dynamical system $\mathcal{S}_\alpha$ have the same set of factors. Therefore if t_n and $t_n^i t_{n-1}$ are the return words of a prefix w of s_α, then they are the return words of the left special factor w in any Sturmian word $x \in \mathcal{S}_\alpha$.

Corollary 1. *Let u, v be the return words of $w \in \mathrm{Pref}(s_\alpha)$ in the characteristic word s_α. Let u', v' be the return words of $w' \in \mathrm{Left}(x)$ in a Sturmian word $x \in \mathcal{S}_\alpha$. If $|w| = |w'|$, then $w = w'$, $u = u'$ and $v = v'$.*

From now on, we consider return words with respect to the *left special factors* of a Sturmian word.

Example 2. We consider the Sturmian word $0s_\alpha$ with slope $\alpha = [0, 3, 2, 3, 2, \ldots]$ (see Example 1). Let us look for the return words of the left special factor $w = 001$. Occurrences of that factor are underlined. The return words are $u = 001$ and $v = 0010$.

$$0\ \underbrace{\underline{001}}_{u}\ \underbrace{\underline{0010}}_{v}\ \underbrace{\underline{001}}_{u}\ \underbrace{\underline{0010}}_{v}\ \underbrace{\underline{001}}_{u}\ \underbrace{\underline{0010}}_{v}\ \underbrace{\underline{001}}_{u}\ \underbrace{\underline{001}}_{u}\ \underbrace{\underline{0010}}_{v}\ \underbrace{\underline{001}}_{u}\ \underline{001}0. \tag{1}$$

These are exactly the return words t_n, $t_n^i t_{n-1}$ of Proposition 2 with $n = i = 1$.

We now introduce the concept of derivated word. Our presentation is a little more general than in [8].

Let $x = x_0 x_1 \cdots x_n \cdots$ be a Sturmian word of slope α, and w be a non empty word in $\mathrm{Left}(x)$. Consider u, v the two return words of w in x (or equivalently in s_α). Let us write $x = px'$ such that the first occurrence of w is equal to $|p|$. Then x can be written in a *unique* way as a concatenation of the words u and v (excluding the prefix p), that is

$$x = px' = pz_1 z_2 z_3 \cdots \quad \text{with } z_i \in \{u, v\}. \tag{2}$$

Denote by $h(s_\alpha)$ the first letter of s_α [2]. We define a bijection $\Delta : \{u, v\} \to \{0, 1\}$ by putting[3]

$$\Delta(u) = h(s_\alpha), \quad \text{and} \quad \Delta(v) = 1 - h(s_\alpha). \tag{3}$$

Then the word

$$D_w(x) = \Delta(z_1)\Delta(z_2)\Delta(z_3)\cdots$$

is called the *derivated word* of x with respect to w. The derivated word $D_w(x)$ is a renaming by 0 and 1 of the occurrences of u and v in the decomposition of x in terms of its return words.

We also define the morphism Θ such that

$$\Theta(h(s_\alpha)) = u \quad \text{and} \quad \Theta(1 - h(s_\alpha)) = v. \tag{4}$$

[2] Recall that the first letter of s_α equals 0 if and only if $a_1 > 1$ in the continued fraction expansion $[0, a_1 + 1, a_2, \ldots]$ of α

[3] This definition of Δ depending on the first letter of s_α will be motivated later

Clearly

$$\Theta(D_w(x)) = x', \tag{5}$$

that is, Θ is the "decoding" morphism that allows to recover x' from $D_w(x)$. Moreover, if we denote $y = D_w(x) = D_w(x')$, we have

$$D_w(\Theta(y)) = y. \tag{6}$$

Note that when w is prefix of x, that is, $x = x'$, then $\Theta(D_w(x)) = x$.

Example 3. Once again consider the Sturmian word $x = 0s_\alpha$ with slope $\alpha = [0, 3, 2, 3, 2, \ldots]$ and the return words $u = 001$ and $v = 0010$ computed in Example 2. Thus we set $\Delta(u) = h(s_\alpha) = 0$ and $\Delta(v) = 1 - h(s_\alpha) = 1$. From (1), we see that the derivated word $D_w(x)$ of x starts with 0101010010 and that $\Theta(D_w(x)) = s_\alpha = \Theta(D_w(s_\alpha))$.

In [3], we have studied the derivated word of a characteristic word s_α with respect to a prefix w of s_α. This study has lead to the characterization of characteristic words which are substitutive, and fixed points of morphisms. These results are stated in the Proposition 3 and Theorems 1, 2 hereafter. We recall that an infinite word $x \in \{0,1\}^\omega$ is *substitutive* if it is the image by a literal morphism of a morphic word, and that x is a *morphic* word if there exists a non erasing morphism φ such that $\varphi(a) = ap$ with $a \in \{0,1\}$, $p \neq \epsilon$, and $x = \lim_{n\to\infty} \varphi^n(a)$. We say that a morphism is *non trivial* if it differs from both the identity and the morphism E defined by $E(0) = 1, E(1) = 0$.

The next proposition indicates that the derivated words of a characteristic word are also characteristic.

Proposition 3. [3] *Let s_α be a characteristic word with slope $\alpha = [0, a_1 + 1, a_2, a_3, \ldots]$. Let $w \in \mathrm{Pref}(s_\alpha)$ be such that its two return words are $t_n, t_n^i t_{n-1}$, with $n \geq 1$ and $i \in \{0, \ldots, a_{n+1} - 1\}$, or $n = 0$ and $i \in \{1, \ldots, a_{n+1} - 1\}$. Then the derivated word $D_w(s_\alpha)$ is the characteristic word s_β of slope*[4]

1. $[0, a_{n+1} + 1 - i, a_{n+2}, a_{n+3}, \ldots]$ *if* $a_1 > 0$;
2. $[0, 1, a_{n+1} - i, a_{n+2}, a_{n+3}, \ldots]$ *if* $a_1 = 0$.

The next corollary is easily proved from this proposition. Given a number α, we recall that its continued fraction expansion $[0, a_1 + 1, a_2, \ldots]$ is *ultimately periodic* if there exist $n, m \geq 1$ such that $a_{n+k} = a_{n+m+k}$ for all $k \geq 1$, and we write it as $[0, a_1 + 1, \ldots, a_n, \overline{a_{n+1}, \ldots, a_{n+m}}]$. The number α is called a *Sturm number* if its continued fraction expansion is of one of the following types:

1. $[0, a_1 + 1, \overline{a_2, \ldots, a_n}]$, $\quad a_n \geq a_1 \geq 1$;
2. $[0, 1, a_1, \overline{a_2, \ldots, a_n}]$, $\quad a_n \geq a_1$.

[4] By this proposition, if we denote $\beta = [0, b_1 + 1, b_2, b_3, \ldots]$, we see that $a_1 > 1$ if and only if $b_1 > 1$ (recall that $a_1 \geq 0$ and $a_n \geq 1$, for $n \geq 2$). In other words, s_α and its derivated words begin with the same letter. This behavior is due to the definition of Δ depending on the first letter of s_α

Corollary 2. [3] *Let s_α be a characteristic word with slope α.*

1. *The continued fraction expansion of α is ultimately periodic if and only if the set $\{D_w(s_\alpha) \mid w \in \mathrm{Pref}(s_\alpha), w \neq \epsilon\}$ is finite.*
2. *The number α is a Sturm number if and only if $D_w(s_\alpha) = s_\alpha$ for some non empty word $w \in \mathrm{Pref}(s_\alpha)$.*

We now state the two announced theorems. The equivalence between (b) and (c) of Theorem 1 is proved by Durand in [8] in a more general context. The equivalence between (a) and (c) of Theorem 2 was first proved by Crisp, Moran, Pollington and Shiue in [7]. Alternative proofs were proposed in [6], [11] and [3].

Theorem 1. *For a characteristic word s_α, the following are equivalent:*

(a) the continued fraction expansion of α is ultimately periodic;
(b) the set $\{D_w(s_\alpha) \mid w \in \mathrm{Pref}(s_\alpha), w \neq \epsilon\}$ is finite;
(c) s_α is substitutive.

Theorem 2. *For a characteristic word s_α, the following are equivalent:*

(a) α is a Sturm number;
(b) there exists $w \in \mathrm{Pref}(s_\alpha)$, $w \neq \epsilon$, such that $D_w(s_\alpha) = s_\alpha$;
(c) s_α is a fixed point of a non trivial morphism.

In this paper, we want to propose adequate generalizations of Theorems 1 and 2 for *all* Sturmian words.

4 Maps Between Two Dynamical Systems

Let us consider the dynamical system $\mathcal{S}_\alpha$ of all Sturmian words of slope α. We recall that for all $x \in \mathcal{S}_\alpha$, $\mathrm{Left}(x) = \mathrm{Pref}(s_\alpha)$. Moreover, we know by Corollary 1 that each $x \in \mathcal{S}_\alpha$ has the same return words u and v, with respect to a given prefix w of s_α. Hence, the two functions Δ and Θ defined in Section 3 by (3) and (4) are the same for every $x \in \mathcal{S}_\alpha$. Therefore, in the sequel, we will use the following notation: $u_{\alpha,w}$, $v_{\alpha,w}$, $\Delta_{\alpha,w}$ and $\Theta_{\alpha,w}$.

We propose in this section two families of maps between dynamical systems $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$. The properties of these two families will be essential in the proofs of the next section. We say that a map $\varphi : \mathcal{S}_\alpha \to \mathcal{S}_\beta$ *respects the structure* of the dynamical systems $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ if whenever $\varphi(x) = y$, then

1. $x = s_\alpha \;\Rightarrow\; y = s_\beta$;
2. $x \in R_\alpha \setminus \{s_\alpha\} \;\Rightarrow\; y \in R_\beta \setminus \{s_\beta\}$.

The first family of maps is related to the derivation operator D_w. Let s_α be a characteristic word and $w \in \mathrm{Pref}(s_\alpha)$. By Proposition 3, $D_w(s_\alpha)$ is a characteristic word; let β be its slope. In the next proposition, we show that the derivated word $D_w(x)$ of a Sturmian word x of slope α is a Sturmian word of slope β. Moreover we prove that the D_w operator defines a surjective map between $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ which respects the structure of the two dynamical systems.

Proposition 4. *Let s_α be a characteristic word and $w \in \mathrm{Pref}(s_\alpha)$. Let s_β be the characteristic word such that $D_w(s_\alpha) = s_\beta$. Then the D_w operator defines a surjective map from $\mathcal{S}_\alpha$ to $\mathcal{S}_\beta$ that respects the structure of the dynamical systems.*

Proof. 1. Let us first prove that D_w defines a map from $\mathcal{S}_\alpha$ to $\mathcal{S}_\beta$, that is, for any $x \in \mathcal{S}_\alpha$, the derivated word $D_w(x)$ is a Sturmian word in $\mathcal{S}_\beta$. Let us verify that $\mathrm{Fact}(D_w(x)) \subseteq \mathrm{Fact}(s_\beta)$. We will conclude that $D_w(x) \in \mathcal{S}_\beta$ by Proposition 1.

By Proposition 1, $\mathrm{Fact}(x) = \mathrm{Fact}(s_\alpha)$. We consider the unique decomposition of s_α in terms of the return words $u_{\alpha,w}$ and $v_{\alpha,w}$ of w (see (2)), that is, $s_\alpha = z_1 z_2 z_3 \cdots$ with $z_i \in \{u_{\alpha,w}, v_{\alpha,w}\}$. We have $\Theta_{\alpha,w}(D_w(s_\alpha)) = \Theta_{\alpha,w}(s_\beta) = s_\alpha$ (see (5)). By (5), we also have $\Theta_{\alpha,w}(D_w(x)) = x'$ where $x = px'$ and $|p|$ is the first occurrence of w in x.

Let $f \in \mathrm{Fact}(D_w(x))$, then $\Theta_{\alpha,w}(f) \in \mathrm{Fact}(x') \subseteq \mathrm{Fact}(x)$. As $\Theta_{\alpha,w}(f)$ is a concatenation of the return words $u_{\alpha,w}$ and $v_{\alpha,w}$, it follows that $\Theta_{\alpha,w}(f)w \in \mathrm{Fact}(x)$. Therefore $\Theta_{\alpha,w}(f)w \in \mathrm{Fact}(s_\alpha)$. Notice that this word has w as prefix and suffix. This implies that $\Theta_{\alpha,w}(f) = z_i z_{i+1} \cdots z_j$, for some indices i, j. It follows that $\Delta_{\alpha,w}(z_i \cdots z_j) = f$ and $f \in \mathrm{Fact}(s_\beta)$. Hence $\mathrm{Fact}(D_w(x)) \subseteq \mathrm{Fact}(s_\beta)$.

2. Let us now prove that the map defined by D_w from $\mathcal{S}_\alpha$ to $\mathcal{S}_\beta$ is surjective. Let $y \in \mathcal{S}_\beta$ and $x = \Theta_{\alpha,w}(y)$. We are going to show that $x \in \mathcal{S}_\alpha$ and $D_w(x) = y$.

By Proposition 1, $\mathrm{Fact}(y) = \mathrm{Fact}(s_\beta)$. Remember that $\Theta_{\alpha,w}(s_\beta) = s_\alpha$.

We are going to verify that $\mathrm{Fact}(x) \subseteq \mathrm{Fact}(s_\alpha)$. Let $f \in \mathrm{Fact}(x)$, then there exists $g \in \mathrm{Fact}(y)$ such that f is factor of $\Theta_{\alpha,w}(g)$ and $\Theta_{\alpha,w}(g) \in \mathrm{Fact}(x)$. Since $g \in \mathrm{Fact}(s_\beta)$, we have $\Theta_{\alpha,w}(g) \in \mathrm{Fact}(s_\alpha)$ and thus $f \in \mathrm{Fact}(s_\alpha)$.

By Proposition 1, x is a Sturmian word in $\mathcal{S}_\alpha$. Clearly $D_w(x) = y$ by (6).

3. Finally let us show that D_w respects the structure of the dynamical systems. Let $x \in \mathcal{S}_\alpha$, $y \in \mathcal{S}_\beta$ be such that $D_w(x) = y$. We already know that $D_w(s_\alpha) = s_\beta$. Suppose that $x = S^i(s_\alpha)$ with S the shift and $i \geq 1$. Then it is easy to verify that $D_w(x) = S^k(D_w(s_\alpha)) = S^k(s_\beta)$ where $k \geq 1$ is the number of occurrences of w in s_α less than i.

This completes the proof.

Example 4. Let $\mathcal{S}_\alpha$ be the dynamical system with slope $\alpha = [0, \overline{3, 2}]$. Consider the word $w = 001 \in \mathrm{Pref}(s_\alpha)$. By Proposition 3, $D_w(s_\alpha) = s_\beta$ with $\beta = [0, \overline{2, 3}]$. It is clear that if $x = S^7(s_\alpha)$, then $D_w(x) = S^2(s_\beta)$. On the other hand, as we have seen in Example 3, $D_w(0s_\alpha) = D_w(s_\alpha) = s_\beta$.

In Proposition 4, notice that if $\alpha = [0, a_1+1, a_2, \ldots]$ and $\beta = [0, b_1+1, b_2, \ldots]$, then either $a_1, b_1 > 0$, or $a_1 = b_1 = 0$ (see footnote of Proposition 3). A way to define a map between two dynamical systems $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ such that either $a_1 > 0, b_1 = 0$, or $a_1 = 0, b_1 > 0$ is to combine the derivation operator D_w with the morphism E such that $E(0) = 1, E(1) = 0$. Indeed

$$E(s_\alpha) = s_{1-\alpha}, \tag{7}$$

and more generally $E(x)$ is a Sturmian word with slope $1 - \alpha$ if x is a Sturmian word with slope α (see [12]). Thus E defines a map from $\mathcal{S}_\alpha$ to $\mathcal{S}_{1-\alpha}$ which is

bijective and respects the dynamical systems. The way D_w and E commute is described by the next equality (see [3])

$$(E \circ D_w)(x) = D_{E(w)}(E(x)). \tag{8}$$

The second family of maps we want to study is the family of locally characteristic morphisms. We recall that a morphism φ is *locally characteristic* if there exists a characteristic word x such that $\varphi(x)$ is a characteristic word. It is called *characteristic* if $\varphi(x)$ is a characteristic word for all characteristic words x. It is known that every locally characteristic morphism is characteristic [12].

Notice that every morphism $\Theta_{\alpha,w}$ is a locally characteristic morphism.

Proposition 5. *Let φ be a locally characteristic morphism which is non trivial. Let s_α, s_β be characteristic words such that $\varphi(s_\beta) = s_\alpha$. Then for some non empty word $w \in \mathrm{Pref}(s_\alpha)$,*

$$\varphi = \Theta_{\alpha,w} \quad or \quad \varphi = \Theta_{\alpha,w} \circ E \ [5],$$

and φ defines an injective map from $\mathcal{S}_\beta$ to $\mathcal{S}_\alpha$ that respects the structure of the dynamical systems.

Proof. In [3] (see proof of Theorem 21), we have proved that $\varphi(0)$, $\varphi(1)$ are the return words $u_{\alpha,w}$ and $v_{\alpha,w}$ of some $w \in \mathrm{Pref}(s_\alpha)$. It follows that either $\varphi = \Theta_{\alpha,w}$, or $\varphi = \Theta_{\alpha,w} \circ E$.

In the first case, $D_w(s_\alpha) = s_\beta$. The properties of Proposition 5 are then easy consequences of Proposition 4 and equations (5), (6). Clearly, φ defines a map from $\mathcal{S}_\beta$ to $\mathcal{S}_\alpha$ since D_w defines a map from $\mathcal{S}_\alpha$ to $\mathcal{S}_\beta$ which is surjective. Let us show that it respects the structure of the dynamical systems. By hypothesis $\varphi(s_\beta) = s_\alpha$. Write $s_\beta = x_0x_1 \cdots x_ix_{i+1} \cdots$ (with $x_n \in \{0,1\}$) and consider $x \in R_\beta$ such that $x = S^i(s_\beta) = x_ix_{i+1} \cdots$. Define $y = S^k(s_\alpha) \in R_\alpha$ where k is equal to $|\Theta_{\alpha,w}(x_0x_1 \cdots x_{i-1})|$. Then $\varphi(x) = y$. Finally, φ is injective because the decomposition of a Sturmian word in terms of the return words is unique.

The case $\varphi = \Theta_{\alpha,w} \circ E$ is solved similarly. By (7), $E(s_\beta) = s_{1-\beta}$ and thus $\Theta_{\alpha,w}(s_{1-\beta}) = s_\alpha$. Hence φ defines an injective map from $\mathcal{S}_\beta$ into $\mathcal{S}_\alpha$ that respects the dynamical systems since $E : \mathcal{S}_\beta \to \mathcal{S}_{1-\beta}$ and $\Theta_{\alpha,w} : \mathcal{S}_{1-\beta} \to \mathcal{S}_\alpha$ both respect the dynamical systems.

To end this section, we give a lemma that will be useful in the next section. We recall that a morphism φ is called *locally Sturmian* if there exists a Sturmian word x such that $\varphi(x)$ is a Sturmian word, and that φ is *Sturmian* if $\varphi(x)$ is a Sturmian word for all Sturmian words x. Moreover, every locally Sturmian morphism is Sturmian [12].

Lemma 1. *Let φ be a Sturmian morphism such that $\varphi(x) = y$ with $x \in \mathcal{S}_\alpha$ and $y \in \mathcal{S}_\beta$. Then there exists a characteristic morphism φ' such that $\varphi'(x) = y'$ and $y' \in \mathcal{S}_\beta$.*

[5] Let $\alpha = [0, a_1 + 1, a_2, \ldots]$ and $\beta = [0, b_1 + 1, b_2, \ldots]$. Then $\varphi = \Theta_{\alpha,w}$ if $a_1, b_1 > 0$ or $a_1 = b_1 = 0$, otherwise $\varphi = \Theta_{\alpha,w} \circ E$

Proof. Let us recall some properties (see [12]). Consider the morphisms E and

$$G : \begin{matrix} 0 \mapsto 0 \\ 1 \mapsto 01 \end{matrix}, \quad \tilde{G} : \begin{matrix} 0 \mapsto 0 \\ 1 \mapsto 10 \end{matrix}.$$

These morphisms are Sturmian. If x has slope α, then $E(x)$ has slope $1-\alpha$ and $G(x), \tilde{G}(x)$ both have slope $\frac{\alpha}{1+\alpha}$. A morphism φ is Sturmian (resp. characteristic) if and only if φ is obtained by composition of E, G and $\tilde{G}$ (resp. of E and G).

Let us now prove the lemma. Given a Sturmian morphism φ, suppose that $\varphi(x) = y$ with $x \in \mathcal{S}_\alpha$ and $y \in \mathcal{S}_\beta$. Decompose $\varphi = \phi_1 \circ \phi_2 \circ \cdots \circ \phi_n$ such that $\phi_i \in \{E, G, \tilde{G}\}$ for all i. Define $\varphi' = \phi'_1 \circ \phi'_2 \circ \cdots \circ \phi'_n$ such that $\phi'_i = \phi_i$ if $\phi_i \in \{E, G\}$, and $\phi'_i = G$ if $\phi_i = \tilde{G}$. Then φ' is a characteristic morphism and $\varphi'(x)$ has the same slope as $\varphi(x)$.

5 Generalizations

In this section, we propose adequate generalizations of Theorems 1 and 2 to all Sturmian words. We begin to generalize the second one.

Theorem 3. *For a Sturmian word x with slope α, the following conditions are equivalent:*

(a) α is a Sturm number;
(b) there exists a word $w \in \mathrm{Left}(x)$, $w \neq \epsilon$, such that $D_w(x)$ has slope α;
(c) there exists a non trivial Sturmian morphism φ such that $\varphi(x)$ has slope α.

Proof. By Corollary 2, Condition (a) is equivalent to Condition (a') that there exists a non empty word $w \in \mathrm{Pref}(s_\alpha)$ such that $D_w(s_\alpha) = s_\alpha$.

$(a') \Leftrightarrow (b)$ Suppose that Condition (a') holds. By Proposition 4, the word $D_w(x)$ belongs to $\mathcal{S}_\alpha$ and thus has slope α. Hence Condition (b) holds.

Conversely suppose that there exists a non empty word $w \in \mathrm{Left}(x)$ such that $D_w(x)$ has slope α. Again by Proposition 4, $D_w(s_\alpha)$ has slope α, that is $D_w(s_\alpha) = s_\alpha$. This proves Condition (a').

$(a') \Leftrightarrow (c)$ If Condition (a') is satisfied, then $\Theta_{\alpha,w}(s_\alpha) = s_\alpha$ (see (5)). We apply Proposition 5 with $\varphi = \Theta_{\alpha,w}$. Hence φ defines a map from $\mathcal{S}_\alpha$ into itself, showing that $\varphi(x)$ has slope α. Therefore Condition (c) holds.

Conversely suppose that there exists a non trivial Sturmian morphism φ such that $\varphi(x)$ has slope α. We can suppose that φ is characteristic by Lemma 1. By Proposition 5, $\varphi(s_\alpha)$ must be equal to the characteristic word s_α. Moreover, there exists $w \neq \epsilon$ such that $\varphi = \Theta_{\alpha,w}$ [6]. In particular, $\Theta_{\alpha,w}(s_\alpha) = s_\alpha$. By (6), $D_w(s_\alpha) = s_\alpha$, showing that Condition (a') holds.

Let us verify that Theorem 3 is a generalization of Theorem 2. Suppose that $x = s_\alpha$. By Proposition 3, $D_w(x)$ is a characteristic word. Since $D_w(s_\alpha)$ has

[6] The other case $\varphi = \Theta_{\alpha,w} \circ E$ cannot occur since x and $\varphi(x)$ have the same slope

slope α, we get $D_w(s_\alpha) = s_\alpha$. Therefore Condition (b) is equivalent to say that $D_w(s_\alpha) = s_\alpha$ for some $w \in \mathrm{Pref}(s_\alpha)$, $w \neq \epsilon$. In Condition (c), we can suppose that φ is characteristic by Lemma 1. Since $x = s_\alpha$, it follows that $\varphi(x)$ is a characteristic word with slope α, that is, $\varphi(s_\alpha) = s_\alpha$. Hence Condition (c) is equivalent to say that s_α is the fixed point of a non trivial morphism φ.

When $x \in R_\alpha \setminus \{s_\alpha\}$, we can rephrase Theorem 3 thanks to Propositions 4 and 5. Indeed, Condition (b) is replaced by the existence of $w \in \mathrm{Left}(x)$, $w \neq \epsilon$, such that $D_w(x) \in R_\alpha \setminus \{s_\alpha\}$. Similarly, Condition (c) is replaced by the existence of a non trivial Sturmian morphism φ such that $\varphi(x) \in R_\alpha \setminus \{s_\alpha\}$.

We now turn to the generalization of Theorem 1.

Theorem 4. *For a Sturmian word x with slope α, the following conditions are equivalent:*

(a) the continued fraction expansion of α is ultimately periodic;
(b) there exist a Sturmian word y and two non empty words $w \in \mathrm{Left}(x)$, $w' \in \mathrm{Left}(y)$ such that y, $D_w(x)$ and $D_{w'}(y)$ have the same slope;
(c) there exist a Sturmian word y and two non trivial Sturmian morphisms φ, ψ such that
- *x and $\psi(y)$ have the same slope,*
- *y and $\varphi(y)$ have the same slope.*

Proof. $(a) \Leftrightarrow (b)$. Let $\alpha = [0, a_1 + 1, a_2, \ldots]$. In this part of the proof, we only treat the case $a_1 > 1$, the other being similar.

Suppose that α is ultimately periodic, that is, there exist $n, m \geq 1$ such that $a_{n+k} = a_{n+m+k}$ for all $k \geq 1$. By Proposition 3, let us consider $w, w_1 \in \mathrm{Pref}(s_\alpha)$ such that $D_w(s_\alpha), D_{w_1}(s_\alpha)$ have slope respectively equal to $[0, a_{n+1}+1, a_{n+2}, \ldots]$ and $[0, a_{n+m+1} + 1, a_{n+m+2}, \ldots]$ (it is the same slope, say β). It follows that $D_w(s_\alpha) = D_{w_1}(s_\alpha) = s_\beta$. Moreover, again by Proposition 3, there exists $w' \in \mathrm{Pref}(D_w(s_\alpha))$ such that $D_{w'}(D_w(s_\alpha)) = D_{w_1}(s_\alpha)$, that is, $D_{w'}(s_\beta) = s_\beta$. We define $y = D_w(x)$. By Proposition 4, y and $D_{w'}(y)$ have the same slope β.

Conversely, suppose that Condition (b) holds. Let β be the slope of y. By using Proposition 4, since $D_w(x)$ has the same slope as y, we have that $D_w(s_\alpha) = s_\beta$. Moreover, as $D_{w'}(y)$ also has slope β, then $D_{w'}(s_\beta) = s_\gamma$ with $\beta = \gamma$. On the other hand, by Proposition 3, we have $\beta = [0, a_{n+1} + 1 - i, a_{n+2}, \ldots]$ for some n, i, and $\gamma = [0, a_{n+m+1} + 1 - j, a_{n+m+2}, \ldots]$ for some m, j. Since $m \geq 1$ and $\beta = \gamma$, it follows that the continued fraction expansion of α is ultimately periodic.

$(b) \Leftrightarrow (c)$ Suppose that there exist a Sturmian word y and two non empty words $w \in \mathrm{Left}(x)$, $w' \in \mathrm{Left}(y)$ such that y, $D_w(x)$ and $D_{w'}(y)$ with the same slope β. By Theorem 3, there exists a non trivial Sturmian morphism φ such that y and $\varphi(y)$ have the same slope β.

By (2) and (5), we have $x = px'$ such that $\Theta_{\alpha,w}(D_w(x)) = x'$. Notice that x' has slope α. We define $\psi = \Theta_{\alpha,w}$. By Proposition 5, as $D_w(x), y \in \mathcal{S}_\beta$, then $x', \psi(y) \in \mathcal{S}_\alpha$. Thus x and $\psi(y)$ have the same slope. Hence Condition (c) holds.

Conversely suppose that there exist a Sturmian word y of slope β and two non trivial Sturmian morphisms φ, ψ such that $x, \psi(y) \in \mathcal{S}_\alpha$ and $y, \varphi(y) \in \mathcal{S}_\beta$. We can suppose that ψ is characteristic by Lemma 1.

By Theorem 3, there exists $w' \in \mathrm{Left}(y)$ such that $D_{w'}(y)$ has slope β. By Proposition 5, we have that $\psi(s_\beta) = s_\alpha$. Moreover there exists $w \in \mathrm{Pref}(s_\alpha)$ such that either $\psi = \Theta_{\alpha,w}$ or $\psi = \Theta_{\alpha,w} \circ E$. Suppose first that $\psi = \Theta_{\alpha,w}$. By Proposition 4, as $x, \psi(y) \in \mathcal{S}_\alpha$, then $D_w(x), D_w(\psi(y)) = y \in \mathcal{S}_\beta$. It follows that Condition (b) holds.

Now if $\psi = \Theta_{\alpha,w} \circ E$, we define $y' = E(y)$ and $w'' = E(w')$. To get Condition (b), let us show that $y', D_w(x), D_{w''}(y')$ all have slope $1 - \beta$. Due to the form of ψ, D_w defines a map from $\mathcal{S}_\alpha$ to $\mathcal{S}_{1-\beta}$. Recall that E defines a map from $\mathcal{S}_\beta$ to $\mathcal{S}_{1-\beta}$. It follows that $y', D_w(x) \in \mathcal{S}_{1-\beta}$. Finally since $y, D_{w'}(y) \in \mathcal{S}_\beta$, we have $y', D_{w''}(y') \in \mathcal{S}_{1-\beta}$ by (8).

Let us now show that this theorem generalizes Theorem 1. Suppose that $x = s_\alpha$. Let us consider Condition (c). By Lemma 1, the two morphisms ψ and φ can be supposed to be characteristic. By Proposition 5, if y has slope β, then $\varphi(s_\beta) = s_\beta$. Moreover $\psi(s_\beta) = s_\alpha$. Therefore, s_α is the image by the morphism ψ of the morphic word s_β. Notice that ψ is not necessarily literal. Nevertheless, it is proved in [8] that the image by a non erasing morphism of a morphic word is a substitutive word. Hence Condition (c) is equivalent to say that s_α is substitutive. Let us now consider Condition (b). When $x = s_\alpha$, this means that $D_w(s_\alpha) = s_\beta$ for some slope β and $D_{w'}(s_\beta) = s_\beta$ (see Proposition 4). As done in the previous proof (part $(b) \Rightarrow (a)$), it follows that the continued fraction of α is ultimately periodic. Thus by Corollary 2, Condition (b) is equivalent to the finiteness of the set $\{D_w(s_\alpha) \mid w \in \mathrm{Pref}(s_\alpha), w \neq \epsilon\}$.

To end this section, let us compare Condition (b) in Theorem 1 and Condition (b) in Theorem 4. When $x = s_\alpha$, we have just proved that they are equivalent. The next proposition indicates that this is also true for $x \in R_\alpha$.

Proposition 6. *Let x be a Sturmian word in R_α. Then the continued fraction expansion of α is ultimately periodic if and only if the set $\{D_w(x) \mid w \in \mathrm{Pref}(s_\alpha), w \neq \epsilon\}$ is finite.*

Proof. Suppose that $x = S^i(s_\alpha)$ with $i \geq 0$. Let $w \in \mathrm{Pref}(s_\alpha)$ and β be such that $D_w(s_\alpha) = s_\beta$. We recall (see part 3. in the proof of Proposition 4) that $D_w(x) = S^k(s_\beta)$ where $k \geq 0$ is the number of occurrences of w in s_α less than i. In particular, k is bounded by i. The proof is completed thanks to Corollary 2.

The equivalence of Proposition 6 is no longer true for $x \in \mathcal{S}_\alpha \setminus R_\alpha$. We were able to construct a counter-example thanks to a very nice theorem in [4]: the Sturmian words of $\mathcal{S}_\alpha$ are the infinite developments in a certain numeration system, under a restrictive condition given by an automaton.

Example 5. Let $\alpha = [0, \overline{2}]$. Given the *non* ultimately periodic sequence $(k_n)_n$

$$0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, \ldots$$

define

$$x = 0^{k_0}1\varphi(0^{k_1}1\varphi(0^{k_2}1\varphi(\ldots)))$$

where $\varphi(0) = 01$ and $\varphi(1) = 001$. It is proved in [4] that x is a Sturmian word of slope α. Moreover, the way x is constructed is related to some "derivates" closed to the derivated words studied here. We can prove that $\{D_w(x) \mid w \in \mathrm{Pref}(s_\alpha), w \neq \epsilon\}$ is infinite.

Acknowledgments

The authors thank Valérie Berthé and Michel Koskas for their suggestions and helpful discussions. The participation of the first author in this work was part of the project POCTI-ISFL-1-143 "Fundamental and Applied Algebra" of Centro de Álgebra da Universidade de Lisboa, financed by FCT and FEDER.

References

1. J.P. Allouche and J. Shallit. *Automatic sequences — Theory, applications, generalizations*. Cambridge University Press, Cambridge, 2003.
2. I.M. Araújo and V. Bruyère. Sturmian words and a criterium by Michaux-Villemaire. *Theoret. Comput. Sci.*, to appear. (*Appeared in* Proceedings of the 4th International Conference on Words (Turku, Finland, 2003), (2003), 83-94.)
3. I.M. Araújo and V. Bruyère. Words derivated from Sturmian words. *Theoret. Comput. Sci.*, to appear.
4. D. Bernardi, A. Guerziz and M. Koskas. Sturmian words: description and orbits. *Preprint*, 30 pages, 2004.
5. J. Berstel. Recent results in Sturmian words. In *Developments in language theory, II (Magdeburg, 1995)*, pages 13–24. World Sci. Publishing, River Edge, NJ, 1996.
6. J. Berstel and P. Séébold. Morphismes de Sturm. *Bull. Belg. Math. Soc. Simon Stevin*, 1(2):175–189, 1994. Journées Montoises (Mons, 1992).
7. D. Crisp, W. Moran, A. Pollington, and P. Shiue. Substitution invariant cutting sequences. *J. Théor. Nombres Bordeaux*, 5(1):123–137, 1993.
8. F. Durand. A characterization of substitutive sequences using return words. *Discrete Mathematics*, 179:89–101, 1998.
9. C. Holton and L. Zamboni. Geometric realizations of substitutions. *Bull. Soc. Math. France*, 126(2):149–179, 1998.
10. J. Justin and L. Vuillon. Return words in Sturmian words and episturmian words. *Theor. Inform. Appl.*, 34(5):343–356, 2000.
11. T. Komatsu and A.J. van der Poorten. Substitution invariant Beatty sequences. *Japan. J. Math. (N.S.)*, 22(2):349–354, 1996.
12. M. Lothaire. *Algebraic Combinatoric on Words*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 2002. Theor. Inform. Appl. 34 (2000), no. 5, 343–356.
13. C. Michaux and R. Villemaire. Presburger arithmetic and recognizability of sets of natural numbers by automata: New proofs of Cobham's and Semenov's theorems. *Annals of Pure and Applied Logic*, 77:251–277, 1996.
14. L. Vuillon. A characterization of Sturmian words by return words. *European J. Combin.*, 22(2):263–275, 2001.

Schützenberger and Eilenberg Theorems for Words on Linear Orderings

Nicolas Bedon and Chloé Rispal

Institut Gaspard Monge, Université de Marne-la-Vallée
5, boulevard Descartes, Champs-sur-Marne
77454 Marne-la-Vallée Cedex 2, France
{Nicolas.Bedon,Chloe.Rispal}@univ-mlv.fr

Abstract. In the case of finite words, a Schützenberger theorem proves that the star-free sets are exactly the languages recognized by finite aperiodic semigroups. This gives an algebraic characterization of star-free sets. The variety theorem of Eilenberg offers a general framework for the algebraic characterization of recognizable sets: it establishes a one-to-one correspondence between varieties of languages of finite words and pseudo-varieties of algebras. This paper contains extensions of those two well-known results to words on countable scattered linear orderings.

1 Introduction

In the finite words case, the algebraic characterization is a powerful tool for the study of rational sets of words. The rational sets are algebraically defined as the sets recognized by finite semigroups. Furthermore, a canonical semigroup, called *syntactic*, can be attached to any recognizable language. The algebraic properties of this semigroup can be used to characterize classes of rational sets. In this direction, Eilenberg [8] has established a one-to-one correspondence between classes of languages and classes of semigroups, known as the *variety* theorem. Schützenberger [21] was the first to consider the class of rational sets of finite words whose syntactic algebras are both finite and aperiodic (without non-trivial groups). This particular sub-class of the rational sets, called *star-free*, is definable by a sub-class of rational expressions, without star iteration but with boolean operations and finite product. Schützenberger has proved that a language is star-free if and only if its syntactic semigroup is finite and aperiodic. The star-free sets play an important role in the logical approach of the rational sets of finite words: the class of rational languages corresponds exactly to the class of languages definable by monadic second order sentences [7] and the class of star-free languages is precisely [10] the class of words defined by first-order sentences equipped with an ordering predicate $<$.

The algebraic approach is especially interesting for words of infinite length. Contrary to the finite words case, a minimal automaton can not any more be attached to a rational set. However, a syntactic algebra can always be attached to a recognizable language. Informally speaking, algebras for infinite words recognizability are semigroups equipped with a product adapted to the length of

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 134–145, 2005.
© Springer-Verlag Berlin Heidelberg 2005

words. When they have a finite number of elements, such algebra can always be finitely described, even if the description of an infinite product is infinite.

The Eilenberg correspondence was extended to words indexed by all the naturals integers (ω-words) by Wilke [22], and to the countable ordinals case by Bedon and Carton [2, 5]. Schützenberger's theorem was also extended to ω-words by Perrin [12], to bi-infinite words by Perrin and Pin [14], to words indexed by countable ordinals by Bedon [3, 4] and to words on scattered linear orderings of finite ranks by Rispal [18].

Rational sets of words on linear orderings have been recently introduced by Bruyère and Carton [6], and their algebraic approach by Rispal and Carton [18, 19]. They have defined a generalization of semigroups, called $\diamond$-semigroups, where the product of any sequence indexed by a scattered ordering is defined. The main results of this paper is an adaptation of the theorems of Eilenberg and Schützenberger to sets of words indexed by countable and scattered linear orderings. This is a first step in the classification of such languages according to the algebraic properties of their syntactic algebras. Those extensions incorporate all the others.

The paper is organized as follows. Section 2 is an introduction to words and rational sets on linear orderings. The $\diamond$-semigroups are presented in Section 3. Star-free sets are introduced in Section 4: we show that a set of words on scattered linear orderings is star-free if and only if it is recognized by a finite aperiodic $\diamond$-semigroup. The proof that any star-free set is recognized by a finite aperiodic $\diamond$-semigroup uses an adaptation of the product of Schützenberger of two semigroups. Conversely, we prove that any set recognized by a finite aperiodic $\diamond$-semigroup S is star-free using an induction on the $\mathcal{D}$-structure of S. We deduce that a rational set is star-free if and only if its syntactic $\diamond$-semigroup is finite and aperiodic. Section 5 is devoted to varieties and Section 6 concludes.

2 Words on Linear Orderings

This section recalls basic definitions on linear orderings but the reader is referred to [20] for a complete introduction. Hausdorff's characterization of countable scattered linear orderings is given and words indexed by linear orderings are introduced.

Let J be a set equipped with an ordering $<$. The ordering J is *linear* if for any j and k in J, either $j < k$ or $k < j$. Let A be a finite set called *alphabet*. A *word* $x = (a_j)_{j \in J}$ indexed by a linear ordering J is a function from J to A. J is called the *length* of x. For instance ω is the length of right-infinite words $a_0 a_1 ...$ and ζ is the length of bi-infinite words $...a_{-1}a_0a_1...$. The *empty word* ϵ is the only word of length 0.

2.1 Product of Words Indexed by Linear Orderings

For any linear ordering J, we denote by $-J$ the backward linear ordering that is the set J equipped with the reverse ordering. For instance, $-\omega$ is the linear ordering of negative integers.

The sum $J + K$ of two linear orderings is the set $J \cup K$ equipped with the ordering $<$ extending the orderings of J and K by setting $j < k$ for any $j \in J$ and $k \in K$. Formally, the *sum* $\sum_{j \in J} K_j$ is the set of all pairs (k, j) such that $k \in K_j$ equipped with the ordering defined by $(k_1, j_1) < (k_2, j_2)$ if and only if $j_1 < j_2$ or ($j_1 = j_2$ and $k_1 < k_2$ in K_{j_1}).

The sum of linear orderings helps to define the product of words. Let J be a linear ordering and let $(x_j)_{j \in J}$ be words of respective length K_j for any $j \in J$. The word $x = \prod_{j \in J} x_j$ obtained by concatenation of the words x_j with respect to the ordering on J has length $L = \sum_{j \in J} K_j$. For instance, if for any $j \in \omega$, we denote by $x_j = a^{\omega^j}$, then $x = \prod_{j \in \omega} x_j$ is the word $x = a^{\omega^\omega}$ of length $\sum_{j \in \omega} \omega^j = \omega^\omega$. The sequence $(x_j)_{j \in J}$ of words is called a *J-factorization* of the word $x = \prod_{j \in J} x_j$. Let $x = \prod_{i \in \omega} x_i$ an ω-factorization. Another factorization $x = \prod_{i \in \omega} y_i$ is called a *superfactorization* if there is a sequence $(k_i)_{i \in \omega}$ of integers such that $y_0 = x_0 \dots x_{k_0}$ and $y_i = x_{k_{i-1}+1} \dots x_{k_i}$ for all $i \geq 1$. Let $J \in \mathcal{S}$ and E be a set of words. Then E^J is the set composed of the words u such that $u = \prod_{j \in J} u_j$ with $u_j \in E$ for any $j \in J$.

2.2 Scattered Linear Orderings

A non-empty linear ordering J is *dense* if for any j and k in J such that $j < k$, there exists an element i of J such that $j < i < k$. It is *scattered* if it contains no dense subordering. The ordering ω of natural integers and the ordering ζ of relative integers are scattered. More generally, ordinals are scattered orderings. We denote by $\mathcal{N}$ the sub-class of finite linear orderings, $\mathcal{O}$ the class of countable ordinals and $\mathcal{S}$ the class of countable scattered linear orderings. Hausdorff has defined the scattered linear orderings using an induction on ordinals. Any scattered linear ordering can be obtained from the finite linear orderings using finite sums, ω-sums and $-\omega$-sums:

Theorem 1 (Hausdorff [9]). *A countable linear ordering J is scattered if and only if J belongs to $\bigcup_{\alpha \in \mathcal{O}} V_\alpha$ where the classes V_α are inductively defined by:*

$$V_0 = \{\mathbf{0}, \mathbf{1}\} \qquad V_\alpha = \{\sum_{j \in J} K_j \mid J \in \mathcal{N} \cup \{\omega, -\omega, \zeta\} \text{ and } K_j \in \bigcup_{\beta < \alpha} V_\beta\}$$

where $\mathbf{0}$ and $\mathbf{1}$ are respectively the orderings with zero and one element.

We denote by $A^\diamond$ the set of all words over A indexed by countable scattered linear orderings, without the empty word.

2.3 Rational Sets of Words on Linear Orderings

Bruyère and Carton [6] have introduced rational expressions and automata for words indexed by countable scattered linear orderings. An automaton on linear

orderings is a classical finite automaton with additional limit transitions of the form $P \longrightarrow q$ or $q \longrightarrow P$ where P is a set of states.

Rational expressions were also defined in [6] for words on scattered linear orderings: the class of rational sets is the smallest class containing the letters and closed under finite product $(\cdot)$, finite union $(\cup)$, finite iteration $(*)$, ω and $-\omega$-product, countable ordinal iteration $(\natural)$ and reverse countable ordinal iteration $(-\natural)$, and a binary operator $\diamond$. A Kleene-like theorem has been established for those words:

Theorem 2 (Bruyère and Carton [6]). *A set of words indexed by countable scattered linear orderings is rational if and only if it is accepted by a finite automata.*

3 Algebraic Characterization of Rational Sets

In [18, 19], Rispal and Carton have given an algebraic characterization of rational sets on scattered linear orderings. They have generalized the semigroups to $\diamond$-semigroups for those words and proved that a set is rational if and only if it is recognizable. In this section, we first recall basic algebraic definitions and then we define the recognizability for those sets of words.

3.1 Algebraic Definitions

A semigroup is a set S equipped with an associative binary product. The semigroup S in which had been added a neutral element is denoted by S^1. An element $e \in S$ is an *idempotent* if $e^2 = e$ and the set of idempotents of S is denoted by $E(S)$. A pair $(s, e) \in S \times E(S)$ is *right linked* (respectively left linked) if $se = s$ (respectively $es = s$). Two right linked pairs (s_1, e_1) and (s_2, e_2) are *conjugated* if there exist a, $b \in S^1$ such that $e_1 = ab$, $e_2 = ba$, $s_1 a = s_2$ and $s_2 b = s_1$. The conjugacy relation is an equivalence relation on right linked pairs [11]. Recall that the Green's relations are defined from the following preorders:

$$s \leq_{\mathcal{R}} t \iff \exists a \in S^1,\ s = ta \qquad s \leq_{\mathcal{L}} t \iff \exists a \in S^1,\ s = at$$
$$s \leq_{\mathcal{J}} t \iff \exists a, b \in S^1,\ s = atb$$

For any $\mathcal{K} \in \{\mathcal{R}, \mathcal{L}, \mathcal{J}\}$, $s\mathcal{K}t$ if and only if $s \leq_{\mathcal{K}} t$ and $t \leq_{\mathcal{K}} s$. $\mathcal{K}$ is an equivalence relation. The equivalence class of $p \in S$ is denoted by $\mathcal{K}(p)$. We also denote by $s <_{\mathcal{K}} t$ iff $s \leq_{\mathcal{K}} t$ and not $t \leq_{\mathcal{K}} s$. Recall that the equivalence relation $\mathcal{D} = \mathcal{RL} = \mathcal{LR}$ is equal to $\mathcal{J}$ when S is finite. A semigroup S is *aperiodic* if there exists an integer n such that for all $s \in S$, $s^n = s^{n+1}$, or equivalently (see [16]), for all $s \in S$, $\mathcal{R}(s) \cap \mathcal{L}(s) = \{s\}$.

3.2 $\diamond$-Semigroups

The product of semigroups is generalized to recognize sets of words indexed by countable scattered linear orderings. A $\diamond$-semigroup is a generalization of an usual semigroup, equipped with a product adapted to scattered orderings:

Definition 1. *A* $\diamond$-semigroup *is a set* S *equipped with a product* $\pi : S^\diamond \longrightarrow S$ *which maps any word of countable scattered linear length over* S *to an element of* S*:*

- *for any element* s *of* S, $\pi(s) = s$;
- *(generalization of associativity) for any word* x *over* S *of countable scattered linear length and for any factorization* $x = \prod_{j \in J} x_j$ *where* $J \in \mathcal{S}$,

$$\pi(x) = \pi(\prod_{j \in J} \pi(x_j))$$

For instance, $A^\diamond$ equipped with the concatenation product is a $\diamond$-semigroup.

Example 1. The set $S = \{0, 1\}$ equipped with the product π defined for any $u \in S^\diamond$ by $\pi(u) = 0$ if u has at least one occurrence of the letter 0 and $\pi(u) = 1$ otherwise is a $\diamond$-semigroup.

For any two elements s and t of a $\diamond$-semigroup (S, π), the finite product $\pi(st)$ is merely denoted by st. The notions of *sub-*$\diamond$*-semigroup*, *morphism of* $\diamond$*-semigroup*, *quotient*, *division* and *congruence* are directly inspired from the usual algebra (see [1] for example). A $\diamond$-semigroup (S, π) is said to be finite if S is finite. Even when S is finite, the function π is not easy to describe because the product of any sequence has to be given. It turns out that the function π can be described using a semigroup structure on S with two additional functions (called τ and $-\tau$) from S to S. This gives a finite description of the function π.

Definition 2. *Let* S *be a semigroup. A function* $\tau : S \longrightarrow S$ *(respectively* $-\tau : S \longrightarrow S$*) is* compatible to the right with S *(respectively* to the left*) if and only if for any* s, t *in* S *and any integer* n *the following properties hold:* $s(ts)^\tau = (st)^\tau$ *and* $(s^n)^\tau = s^\tau$ *(respectively* $(st)^{-\tau}s = (ts)^{-\tau}$ *and* $(s^n)^{-\tau} = s^{-\tau}$*).*

Compatible functions can be used to describe products in $\diamond$-semigroups.

Theorem 3 ([18, 19]). *Let* (S, π) *be a finite* $\diamond$*-semigroup. The binary product defined for any* s, t *in* S *by* $s \cdot t = \pi(st)$ *naturally endows a structure of semigroup and the functions* τ *and* $-\tau$ *respectively defined by* $s^\tau = \pi(s^\omega)$ *and* $s^{-\tau} = \pi(s^{-\omega})$ *are respectively compatible to the right and to the left with* S*.*
Conversely, let S *be a finite semigroup and let* τ *and* $-\tau$ *be functions respectively compatible to the right and to the left with* S*. Then* S *can be uniquely endowed with a structure of* $\diamond$*-semigroup* (S, π) *such that* $s^\tau = \pi(s^\omega)$ *and* $s^{-\tau} = \pi(s^{-\omega})$*.*

It is well known that rational sets of finite words are exactly those recognized by finite semigroups. This result is generalized for words indexed by countable scattered linear orderings.

Definition 3. *Let* S *and* T *be two* $\diamond$*-semigroups. The* $\diamond$*-semigroup* T recognizes *a subset* X *of* S *if and only if there exist a morphism* $\varphi : S \longrightarrow T$ *and a subset* $P \subseteq T$ *such that* $X = \varphi^{-1}(P)$*. A set* $X \subseteq A^\diamond$ *is* recognizable *if and only if there exists a* ***finite*** $\diamond$*-semigroup recognizing it.*

For any alphabet A, $Rec(A^\diamond)$ denotes the set of recognizable subsets of $A^\diamond$.

Theorem 4 (Rispal and Carton [18, 19]). *A set of words indexed by countable scattered linear orderings is rational iff it is recognizable.*

Example 2. Let $A = \{a, b\}$ and $S = \{s, t, e, f, 0\}$ be the $\diamond$-semigroup whose product is defined by $st = e$, $ts = f$, $ee = e$, $ff = f$, $es = s$, $ft = t$, $sf = s$, $te = t$, $e^\tau = e$, $e^{-\tau} = e$, $f^\tau = t$, $f^{-\tau} = s$, and where any other product is equal to 0. Let $\varphi : A^\diamond \to S$ be the morphism defined by $\varphi(a) = s$ and $\varphi(b) = t$. Then S recognizes in particular $(ab)^\diamond = \varphi^{-1}(e)$.

3.3 Syntactic $\diamond$-Semigroups

Contrary to automata, the algebraic characterization associates a canonical object, called *syntactic $\diamond$-semigroup*, to each rational set. Let S be a $\diamond$-semigroup and $P \subseteq S$. The equivalence relation $\sim_P$ is defined for any s,t in S by $s \sim_P t$ if and only if for any integer m,

$$\forall s_1, s_2, \ldots, s_m, t_1, t_2, \ldots, t_m \in S^1 \ , \forall \theta_1, \theta_2, \ldots, \theta_{m-1} \in \{\omega, -\omega\} \cup \mathcal{N},$$

$$\pi(s_m(\ldots(s_2(s_1 s t_1)^{\theta_1} t_2)^{\theta_2} \ldots)^{\theta_{m-1}} t_m) \in P$$

$$\iff \pi(s_m(\ldots(s_2(s_1 t t_1)^{\theta_1} t_2)^{\theta_2} \ldots)^{\theta_{m-1}} t_m) \in P.$$

Note that m is bounded when S is finite. For each rational set X, the $\diamond$-semigroup $A^\diamond/\sim_X$ is the smallest $\diamond$-semigroup recognizing X in the sense of division. It is called the *syntactic $\diamond$-semigroup of X* and is denoted by $S(X)$.

Proposition 1 ([18, 19]). *A subset X of $A^\diamond$ is recognizable if and only if the relation $\sim_X$ is a congruence of $\diamond$-semigroups of finite index.*

4 Star-Free Sets

In this section, we define star-free sets of words on scattered linear orderings and we extend the Schützenberger's theorem which establishes the equivalence between languages recognized by aperiodic finite semigroups and star-free sets of finite words.

Definition 4. *Let A be an alphabet. The class $SF(A^\diamond)$ of* star-free sets of words *on countable scattered linear orderings on A is the smallest set containing all $\{a\}$ for $a \in A$ and closed under finite union, complement with respect to $A^\diamond$ and finite product.*

Example 3. For any alphabet A, $(A^\diamond)^\omega = A^\diamond \setminus A^\diamond A \in SF(A^\diamond)$ and $(A^\diamond)^{-\omega} = A^\diamond \setminus AA^\diamond \in SF(A^\diamond)$.

Theorem 5. *Let A be an alphabet and X a recognizable subset of $A^\diamond$. The following conditions are equivalent:*

1. $X \in SF(A^\diamond)$,
2. $A^\diamond/\sim_X$ *is a finite and aperiodic* $\diamond$*-semigroup.*

Example 4. Let $A = \{a, b\}$ and $L = (ab)^\diamond$. The syntactic $\diamond$-semigroup of L is given by Example 2. One can easily check its aperiodicity. The language L is a star-free set, described by the expression:

$$L = A^\diamond \setminus (A^\diamond aaA^\diamond \cup A^\diamond bbA^\diamond \cup bA^\diamond \cup A^\diamond a \cup (A^\diamond)^\omega bA^\diamond \cup A^\diamond a(A^\diamond)^{-\omega})$$

The remaining of the paper is devoted to a sketch of the proof of Theorem 5.

4.1 From Star-Free Sets to Finite Aperiodic $\diamond$-Semigroups

In this section, we prove that the class of languages recognized by finite aperiodic $\diamond$-semigroups is closed under finite product and boolean operations.

The Schützenberger's product of two semigroups S and T, denoted by $S \diamond T$ is the set $S \times \mathcal{P}(S^1 \times T^1) \times T$ equipped with the finite product defined by:

$$(s_1, P_1, t_1) \cdot (s_2, P_2, t_2) = (s_1 s_2, s_1 P_2 \cup P_1 t_2, t_1 t_2)$$

where $s_1 P_2 = \{(s_1 s_2, t_2) \mid (s_2, t_2) \in P_2\}$ and $P_1 t_2 = \{(s_1, t_1 t_2) \mid (s_1, t_1) \in P_1\}$. An element $(s,\ P,\ t)$ of $S \diamond T$ is represented by the matrix $\begin{bmatrix} s & P \\ 0 & t \end{bmatrix}$ and the finite product is the product of matrices. For any finite $\diamond$-semigroups S and T, $S \diamond T$ is extended to a $\diamond$-semigroup. The function τ is defined by, for any $(s, P, t) \in S \diamond T$,

$$\begin{bmatrix} s & P \\ 0 & t \end{bmatrix}^\tau = \begin{bmatrix} s^{\tau_S} & \{(s^k s', t' t^{\tau_T}) \mid k \in \mathbb{N}, (s', t') \in P\} \cup \{(s^{\tau_S}, 1)\} \\ 0 & t^{\tau_T} \end{bmatrix}$$

where τ_S, τ_T are respectively the functions compatible to the right with S and T. The function $-\tau$ is defined by symmetrical arguments.

Lemma 1. *Let S and T be two finite $\diamond$-semigroups. The functions τ and $-\tau$ are respectively compatible to the right and to the left with $S \diamond T$.*

Lemma 1 shows that the $\diamond$-semigroup $S \diamond T$ is well defined. Moreover, the Schützenberger's product of two finite aperiodic $\diamond$-semigroups is still aperiodic.

Proposition 2. *The set of finite aperiodic $\diamond$-semigroups is closed under Schützenberger's product.*

Let $\varphi : A^\diamond \longrightarrow S$ and $\psi : A^\diamond \longrightarrow T$ be two morphisms of $\diamond$-semigroups. The morphism $\varphi \diamond \psi$ defined for any $a \in A$ by

$$\varphi \diamond \psi(a) = \begin{bmatrix} \varphi(a) & \{(\varphi(a), 1), (1, \psi(a))\} \\ 0 & \psi(a) \end{bmatrix}$$

satisfies the property of Lemma 2.

Lemma 2. *For any word $u \in A^\diamond$,*

$$\varphi \diamond \psi(u) = \begin{bmatrix} \varphi(u) & \{(\varphi(v), \psi(w)) \mid u = vw\} \\ 0 & \psi(u) \end{bmatrix}$$

As a consequence, the class of languages recognized by finite aperiodic $\diamond$-semigroups is closed under finite product and boolean operations.

Proposition 3. *Any star-free subset of $A^\diamond$ is recognized by a finite aperiodic $\diamond$-semigroup.*

4.2 From Finite Aperiodic $\diamond$-Semigroups to Star-Free Sets

Let A be an alphabet and S a finite aperiodic $\diamond$-semigroup. In this section we prove that a language X recognized by a morphism $\varphi : A^\diamond \to S$ of $\diamond$-semigroups belongs to $SF(A^\diamond)$.

As $SF(A^\diamond)$ is closed under finite union it suffices to prove that $\forall p \in S$, $\varphi^{-1}(p) \in SF(A^\diamond)$.

The proof is by induction on the structure in $\mathcal{D}$-classes of S.

For the induction step, we will assume that $\varphi^{-1}(s) \in SF(A^\diamond)$ for every $s \in S$ such that $p <_\mathcal{D} s$ and show that this implies $\varphi^{-1}(p) \in SF(A^\diamond)$. This is the technique used in the original proof [21] of the version of Theorem 5 for finite words, but the proof we present now is an adaptation of the proof from [13]. It is based on the following property of aperiodic semigroups:

Lemma 3. *Let A be an alphabet, S a finite aperiodic $\diamond$-semigroup and $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups. For any $p \in S$,*

$$\varphi^{-1}(p) \quad = \quad \left(\varphi^{-1}(\mathcal{R}(p))A^\diamond \ \cap \ A^\diamond\varphi^{-1}(\mathcal{L}(p))\right) \ \setminus \ \bigcup_{p \not\leq_\mathcal{D} r} \varphi^{-1}(r). \quad (1)$$

In order to prove that $\varphi^{-1}(p) \in SF(A^\diamond)$ we show, step by step, that the right member of the previous equality belongs to $SF(A^\diamond)$. The following lemma will be needed.

Lemma 4. *Let A be an alphabet, n an integer and let $(L_i)_{1 \leq i \leq n}$ be a family of star-free sets: for any $1 \leq i \leq n$, $L_i \in SF(A^\diamond)$. Then $(\bigcup_{1 \leq i \leq n} L_i A^\diamond)^\omega \in SF(A^\diamond)$ and $(A^\diamond \bigcup_{1 \leq i \leq n} L_i)^{-\omega} \in SF(A^\diamond)$.*

In the induction, we also need to prove that, if $p <_\mathcal{D} e$ and if, for every s such that $p <_\mathcal{D} s$, $\varphi^{-1}(s) \in SF(A^\diamond)$, then $\varphi^{-1}(e)^\omega$ is a star-free set:

Lemma 5. *Let S be a finite $\diamond$-semigroup, $p \in S$ and $\varphi : A^\diamond \to S$ be a morphism of $\diamond$-semigroups. Assume that $\varphi^{-1}(s) \in SF(A^\diamond)$ for every $s \in S$ such that $p <_\mathcal{D} s$, and let e be an idempotent of S such that $p <_\mathcal{D} e$. Then $\varphi^{-1}(e)^\omega \in SF(A^\diamond)$ and $\varphi^{-1}(e)^{-\omega} \in SF(A^\diamond)$.*

The previous Lemma is a corollary of the following star-free expression.

Lemma 6. *Let A be an alphabet, S a finite $\diamond$-semigroup and $\varphi : A^\diamond \to S$ be a morphism of $\diamond$-semigroups. For any idempotent $e \in E(S)$,*

$$\varphi^{-1}(e)^\omega = (A^\diamond)^\omega \setminus Z \text{ where } Z = \bigcup_{\substack{s\in S\\ e\leq_{\mathcal{D}} s}} \varphi^{-1}(s)[(A^\diamond)^\omega \setminus \bigcup_{\substack{t\in S\\ st=e}} \varphi^{-1}(t)\varphi^{-1}(e)(A^\diamond)^\omega].$$

A similar expression holds for the $\varphi^{-1}(e)^{-\omega}$ case.

In order to prove that the right member of the equality (1) is star-free, we first prove that $\varphi^{-1}(\mathcal{R}(p))A^\diamond$ and $A^\diamond\varphi^{-1}(\mathcal{L}(p))$ are included in star-free expressions containing also words of $\cup_{p\not\leq_{\mathcal{D}} r}\varphi^{-1}(r)$.

Lemma 7. *Let A be an alphabet, S a finite $\diamond$-semigroup, $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups and p an element of S. Assume that $\varphi^{-1}(s) \in SF(A^\diamond)$ for every $s \in S$ such that $p <_{\mathcal{D}} s$. There exists a star-free expression R_p such that $\varphi^{-1}\big(\mathcal{R}(p)\big)A^\diamond \subseteq R_pA^\diamond \subseteq \varphi^{-1}\big(\mathcal{R}(p)\big)A^\diamond \cup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r)$. There also exists a star-free expression L_p such that $A^\diamond\varphi^{-1}\big(\mathcal{L}(p)\big) \subseteq A^\diamond L_p \subseteq A^\diamond\varphi^{-1}\big(\mathcal{L}(p)\big) \cup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r)$.*

Then it only remains to prove that $\bigcup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r)$ is a star-free set. We need the following "elementary expressions" for words decomposition:

Definition 5. *Let A be an alphabet, S a finite $\diamond$-semigroup and $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups. For any $s \in S$, we define F_s by*

$$F_s = \bigcup_{\substack{a\in A\\ \varphi(a)=s}} \{a\} \bigcup_{\substack{e\in E(S)\\ e^\tau=s}} \varphi^{-1}(e)^\omega \bigcup_{\substack{e\in E(S)\\ e^{-\tau}=s}} \varphi^{-1}(e)^{-\omega}$$

Lemma 8. *Let A be an alphabet, S a finite $\diamond$-semigroup, $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups and $p \in S$. Then*

$$\bigcup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r) = \bigcup_{\substack{p\leq_{\mathcal{D}} ds, sd'\\ p\not\leq_{\mathcal{D}} dsd'}} A^\diamond F_d\varphi^{-1}(s)F_{d'}A^\diamond \bigcup_{p\not\leq_{\mathcal{D}} r} A^\diamond F_rA^\diamond \bigcup_{\substack{p\leq_{\mathcal{D}} d,d'\\ p\not\leq_{\mathcal{D}} dd'}} A^\diamond F_dF_{d'}A^\diamond$$

We can prove that each union of the right member of the equality is included in a star-free set which is itself included in $\bigcup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r)$. For any idempotent e of S, if $p <_{\mathcal{D}} e$, then by Lemma 5, $\varphi^{-1}(e)^\omega \in SF(A^\diamond)$. Otherwise, if $e\,\mathcal{D}\,p$, by Lemma 7, there exists a star-free expression R_e such that $\varphi^{-1}(e)^\omega \subseteq (R_eA^\diamond)^\omega$. Using the property that two right linked pairs of a same $\mathcal{D}$-class are conjugated if and only if their first components are $\mathcal{R}$-equivalent, we prove that $(R_eA^\diamond)^\omega \subseteq \varphi^{-1}(e^\tau) \cup \bigcup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r)$. Moreover, by Lemma 4, $(R_eA^\diamond)^\omega \in SF(A^\diamond)$. Symmetrical arguments are used for $\varphi^{-1}(e)^{-\omega}$.

Lemma 9. *Let A be an alphabet, S a finite aperiodic $\diamond$-semigroup, $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups and $p \in S$. If for every $s \in S$ such that $p <_{\mathcal{D}} s$, $\varphi^{-1}(s) \in SF(A^\diamond)$, then $\bigcup_{p\not\leq_{\mathcal{D}} r} \varphi^{-1}(r) \in SF(A^\diamond)$.*

Finally,

Proposition 4. *Any language of $A^\diamond$ recognized by a finite aperiodic $\diamond$-semigroup is star-free.*

Proof. Let S be a finite aperiodic $\diamond$-semigroup, $\varphi : A^\diamond \to S$ a morphism of $\diamond$-semigroups and P a subset of S. Since $SF(A^\diamond)$ is closed under finite union, we show that for any $p \in P$, $\varphi^{-1}(p) \in SF(A^\diamond)$.

For the initial step of the induction, let $p \in S$ such that there does not exist $s \in S$ with $p <_{\mathcal{D}} s$. According to Lemma 7, there exist star-free expressions R_p and L_p such that

$$\big(\varphi^{-1}\big(\mathcal{R}\,(p)\big)A^\diamond \cap A^\diamond\varphi^{-1}\big(\mathcal{L}\,(p)\big)\big) \setminus \bigcup_{p \not\leq_{\mathcal{D}} r} \varphi^{-1}(r) = (R_pA^\diamond \cap A^\diamond L_p) \setminus \bigcup_{p \not\leq_{\mathcal{D}} r} \varphi^{-1}(r).$$

According to Lemma 9, $\cup_{p \not\leq_{\mathcal{D}} r}\varphi^{-1}(r) \in SF(A^\diamond)$. Finally, Lemma 3 proves that $\varphi^{-1}(p) \in SF(A^\diamond)$.

For the induction step, let $p \in S$ such that for all $s \in S$, $p <_{\mathcal{D}} s$ implies $\varphi^{-1}(s) \in SF(A^\diamond)$. The induction hypothesys can be used together with Lemmas 7, 9 and 3 to prove, similarly to the initial step, that $\varphi^{-1}(p) \in SF(A^\diamond)$. □

Propositions 4 and 3 together prove that a set of words on scattered linear orderings is star-free if and only if it is recognized by a finite aperiodic $\diamond$-semigroup. Since for any rational set X, $S(X)$ divides any $\diamond$-semigroup recognizing X, this proves Theorem 5.

5 Varieties

In this section, we extend the Eilenberg one-to-one correspondence between pseudo-varieties of semigroups and varieties of rational sets of finite words (see [8, 17]) to languages of words recognized by $\diamond$-semigroups. We first define both notions of pseudo-variety of $\diamond$-semigroups and of variety of $\diamond$-languages. All $\diamond$-semigroups considered in this section are finite, except free $\diamond$-semigroups.

A *pseudo-variety* of $\diamond$-semigroups is a class of $\diamond$-semigroups closed under division and finite product. We will denote pseudo-varieties of $\diamond$-semigroups by bold letters.

Example 5. The class of commutative $\diamond$-semigroups (satisfying $xy = yx$) is a pseudo-variety of $\diamond$-semigroups.

Before defining the notion of a variety of $\diamond$-languages, we need the notion of a residual of a language.

Definition 6. *Let S be a $\diamond$-semigroup, P a part of S, $s, t \in S^1$ and $\theta \in \{\omega, -\omega, 1\}$. The* residuals *of P are the subsets defined by:*

$$(sPt)^{-\theta} = \{p \in S \mid (spt)^\theta \in P\}$$

It may be checked that if X is recognized by a $\diamond$-semigroup S, all the residuals of X are also recognized by S. In particular, a rational language has a finite number of residuals.

Definition 7. *A* variety of $\diamond$-language $\mathcal{V}$ *is a function which associates to any alphabet A a class $A^\diamond\mathcal{V}$ of rational $\diamond$-languages of $A^\diamond$ such that:*

- *for any alphabet A, if $L, L' \in A^\diamond\mathcal{V}$, $u, v \in A^\diamond$ and $\theta \in \{\omega, -\omega, 1\}$ then $L \cup L' \in A^\diamond\mathcal{V}$, $A^\diamond \setminus L \in A^\diamond\mathcal{V}$ and $(uLv)^{-\theta} \in A^\diamond\mathcal{V}$;*
- *if $\varphi : A^\diamond \to B^\diamond$ is a morphism of free $\diamond$-semigroups and $L \in B^\diamond\mathcal{V}$ then $\varphi^{-1}(L) \in A^\diamond\mathcal{V}$.*

We now use pseudo-varieties of $\diamond$-semigroups to give a classification of rational $\diamond$-languages by means of the properties of their syntactic $\diamond$-semigroups. If $\mathbf{V}$ is a pseudo-variety of $\diamond$-semigroups and A an alphabet, we denote by $A^\diamond\mathcal{V}$ the set of languages of $A^\diamond$ recognized by an $\diamond$-semigroup of $\mathbf{V}$. Since a variety of $\diamond$-semigroups is closed under division, a language X belongs to $A^\diamond\mathcal{V}$ iff its syntactic $\diamond$-semigroup $S(X)$ belongs to $\mathbf{V}$. It is straightforward to verify that if $\mathbf{V}$ is a pseudo-variety of $\diamond$-semigroups, then $\mathcal{V}$ is a variety of $\diamond$-languages.

We are now able to state the main theorem which extends Eilenberg's theorem to words on countable scattered linear orderings (see [1, p. 65] or [17, Cor. 4.8]). Its proof essentially mimics the proof for the case of finite words.

Theorem 6 (Correspondence theorem). *The map $\mathbf{V} \to \mathcal{V}$ is a bijection between varieties of $\diamond$-semigroups and varieties of $\diamond$-languages.*

As an example, the class $\mathbf{A}$ of finite aperiodic $\diamond$-semigroups is a pseudo-variety of $\diamond$-semigroups, and the languages whose syntactic $\diamond$-semigroups belongs to this pseudo-variety are precisely the star-free sets, which makes up a variety of languages $\mathcal{A}$.

6 Conclusion

This paper extends to words on countable and scattered linear orderings the theorems of Eilenberg and Schützenberger on finite words. Those two results are stated without consideration for the empty word: the empty word can easily be taken into account with slight modifications.

Historically, the recognizable and the star-free languages of finite words were the first classes of languages characterized by algebraic properties. A lot of other sub-classes of recognizable sets have been algebraically characterized since that time. Such results could be extended to words on linear orderings.

Furthermore, the rational languages can also be characterized by their logical properties. Büchi [7] proved that a language of finite words is rational if and only if it is definable by a formula of the weak monadic second order logic equipped with an ordering predicate $<$. A logical characterization of the star-free sets of finite words have been established by McNaughton and Papert [10]: a set is star-free if and only if it is definable by a formula of the same logic restricted to first-order. We are working on extensions of those characterizations to words on countable and scattered linear orderings.

References

1. Jorge Almeida. *Finite semigroups and universal algebra*, volume 3 of *Series in algebra*. World Scientific, 1994.
2. Nicolas Bedon. *Langages reconnaissables de mots indexés par des ordinaux*. PhD thesis, University of Marne-la-Vallée, France, 1998.
3. Nicolas Bedon. Logic over words on denumerable ordinals. *Journal of Computer and System Science*, 63(3):394–431, November 2001.
4. Nicolas Bedon. Star-free sets of words on ordinals. *Inform. Comput.*, 166:93–111, 2001.
5. Nicolas Bedon and Olivier Carton. An Eilenberg theorem for words on countable ordinals. In Cláudio L. Lucchesi and Arnaldo V. Moura, editors, *Latin'98: Theoretical Informatics*, volume 1380 of *Lect. Notes in Comput. Sci.*, pages 53–64. Springer-Verlag, 1998.
6. Véronique Bruyère and Olivier Carton. Automata on linear orderings. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *MFCS'2001*, volume 2136 of *Lect. Notes in Comput. Sci.*, pages 236–247, 2001.
7. J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeit. Math. Logik. Grund. Math.*, 6:66–92, 1960.
8. Samuel Eilenberg. *Automata, languages and machines*, volume B. Academic Press, 1976.
9. Felix Hausdorff. *Set Theory*. Chelsea, New York, 1957.
10. Robert McNaughton and Seymour Papert. *Counter free automata*. MIT Press, Cambridge, MA, 1971.
11. Jean-Pierre Pécuchet. Variétés de semigroupes et mots infinis. In B. Monien and G. Vidal-Naquet, editors, *STACS '86*, volume 210 of *Lect. Notes in Comput. Sci.*, pages 180–191, 1986.
12. Dominique Perrin. Recent results on automata and infinite words. In M. P. Chytil and V. Koubek, editors, *Mathematical foundations of computer science*, volume 176 of *Lect. Notes in Comput. Sci.*, pages 134–148, Berlin, 1984. Springer.
13. Dominique Perrin. *Handbook of theoretical computer science*, volume B, chapter Finite automata, pages 1–53. Elsevier, 1990.
14. Dominique Perrin and Jean-Éric Pin. First order logic and star-free sets. *J. Comput. System Sci.*, 32:393–406, 1986.
15. Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.
16. Jean-Éric Pin. *Variétés de langages formels*. Masson, Paris, France, 1984. English version: Varieties of formal languages, Plenum Press, New-York, 1986.
17. Jean-Éric Pin. *Handbook of formal languages*, volume 1, chapter Syntactic semigroups, pages 679–746. Springer, 1997.
18. Chloé Rispal. *Automates sur les ordres linéaires: complémentation.* PhD thesis, University of Marne-la-Vallée, France, 2004.
19. Chloé Rispal and Olivier Carton. Complementation of rational sets on countable scattered linear orderings. In C. S. Calude, E. Calude, and M. J. Dinneen, editors, *DLT'2004*, volume 3340 of *Lect. Notes in Comput. Sci.*, pages 381–392, 2004.
20. Joseph G. Rosenstein. *Linear Orderings*. Academic Press, 1982.
21. Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Inform. Control*, 8:190–194, 1965.
22. Thomas Wilke. An Eilenberg theorem for ∞-languages. In *Automata, Languages and Programming: Proc. of 18th ICALP Conference*, pages 588–599. Springer, 1991.

On the Membership of Invertible Diagonal Matrices

Paul Bell and Igor Potapov*

Department of Computer Science,
University of Liverpool, Chadwick Building,
Peach St, Liverpool L69 7ZF, UK
{pbell,igor}@csc.liv.ac.uk

Abstract. In this paper we consider decidability questions that are related to the membership problem in matrix semigroups. In particular we consider the membership of a particular invertible diagonal matrix in a matrix semigroup and then a scalar matrix, which has a separate geometric interpretation. Both problems have been open for any dimensions and are shown to be undecidable in dimenesion 4 with integral matrices and in dimension 3 with rational matrices by a reduction of the Post Correspondence Problem (PCP). Although the idea of PCP reduction is standard for such problems, we suggest a new coding technique to cover the case of diagonal matrices.

1 Introduction

In this paper we consider decidability questions that are related to the membership problem in matrix semigroups. The membership problem for a semigroup with only one generator ("is a matrix B a power of a matrix A") was known as the "orbit problem" and was shown to be decidable (in polynomial time) by Kannan and Lipton in 1986 [6]. The most natural generalization of the the "orbit problem" is the membership problem for matrix semigroups, given by a list of generators.

Problem 1. **The membership problem.** Let S be a given finitely generated semigroup of $n \times n$ matrices from $\mathbb{Z}^{n\times n}$. Determine whether a matrix M belongs to S. In other words, determine whether there exists a sequence of matrices $M_1, M_2, \ldots, M_k$ in S such that $M_1 \cdot M_2 \cdots M_k = M$.

Paterson [9] showed that the problem is undecidable even for 3×3 integral matrices when he considered a special case of the membership problem for matrix semigroups - the *mortality problem* (determination of whether the zero matrix belongs to a matrix semigroup). It was show in [5] that the mortality problem is undecidable even for a case of two generators where the dimension of the matrices is at least 24.

* Work partially supported by The Nuffield Foundation grant NAL/00684/G

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 146–157, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The current research in this area is focused on long standing open problems in low dimensions such as freeness, membership or vector reachability prolems for 2×2 matrices and on the problems that are open in any dimension, like the membership problem of a unity matrix in a matrix semigroup [1]. A related problem, also open at the moment, asks whether the semigroup S contains a diagonal matrix [2]. In this context, we consider the following problem:

Problem 2. Given a multiplicative semigroup S generated by a finite set of $n \times n$ integer matrices and an invertible diagonal matrix M_{D}. Decide whether the semigroup S contains the matrix M_{D}.

In this paper we show that above problem is undecidable in dimension 4 and in case of rational matrices it is undecidable in dimension 3. As a corollary of the above fact we show that the membership of a scalar invertible matrix is also undecidable for a 3×3 rational matrix semigroup and 4×4 integral matrix semigroup. One of the interpretations for a scalar matrix is geometric scaling that alters the size of an object. So checking the membership of a scalar matrix gives us an answer to the following geometric problem:

Problem 3. **Scaling problem.** Given a finite set of linear transformations and a scalar $k \in \mathbb{N}$. Decide whether it is possible to have a combination of such transformations that enlarges the size of an object k times.

Both Problem 2 and Problem 3 are shown to be undecidable by a reduction of the Post Correspondence Problem (PCP) to the membership problem in an integral matrix semigroup. The idea of PCP reduction is quite common in algorithmic matrix problems [5, 9, 10], but in this paper we use a different technique. In particular, in the process of coding the words and indices, we design a set of matrices in such a way that only the correct order of matrix products leads to a particular matrix in a semigroup. In other words we design the semigroup generator set as a set of "tiles" that should be connected in the product with appropriate order, otherwise the product preserves some parts that cannot be reduced later.

2 Notations and Definitions

In what follows we use traditional denotations $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}^+$ and $\mathbb{Q}$ for the sets of naturals, integers, non-negative integers and rationals, respectively.

A *semigroup* is a pair $(S, \cdot)$, where S is a set and $\cdot$ is an associative binary operation on S. A semigroup $(S, \cdot)$ is generated by a set A of its elements iff every element of S is a finite product $a_{i_1} \cdot a_{i_2} \cdot \ldots \cdot a_{i_k}$ where $a_{i_j} \in A$. The set of $n \times n$ matrices over integers is denoted by $\mathbb{Z}^{n \times n}$. It is clear that the identity element for a semigroup $(\mathbb{Z}^{n \times n}, \cdot)$ is the identity matrix that we denote by E_n (or E). Minor $M_{i_1,\ldots,i_k}^{j_1,\ldots,j_l}$ of a matrix M is just the matrix formed from the selected rows $i_1, \ldots, i_k$ and columns $j_1, \ldots, j_l$

$$\mathbf{M} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{pmatrix} \qquad \mathbf{M}_{i1..1k}^{j1..jl} = \begin{pmatrix} a_{i_1,j_1} & \cdots & a_{i_1,j_l} \\ \vdots & \ddots & \vdots \\ a_{i_k,j_1} & \cdots & a_{i_k,j_l} \end{pmatrix}.$$

We denote an empty word by ϵ. The concatenation of two strings w and v is a string obtained by appending the symbols of v to the right end of w, that is, if $w = a_1a_2 \dots a_n$ and $v = b_1b_2 \dots b_n$ then the concatenation of w and v, denoted by $w \cdot v$ or wv, is $a_1a_2 \dots a_nb_1b_2 \dots b_n$. We denote a word $\underbrace{a \cdot a \cdots a}_{k}$ by a^k. The reverse of a string is obtained by writing the symbols in reverse order; if w is a string as shown above, then its reverse w^{R} is $a_n \dots a_2a_1$. The inverse of a character a is written a^{-1} and is the unique character such that $a \cdot a^{-1}$ is equal to the identity element. For any word w, we define $\mathrm{suff}_n(w)$ to be the *suffix* of length n from the word w.

We also define a notation for use with words called an *inverse palindrome*. This is a word in which the second half is equal to the reverse and inverse of the first half of the word. For example if w is an inverse palindrome, then it can be written $w = z \cdot (z^R)^{-1}$ for some word z. It is clear that any inverse palindrome is equal to the identity element.

2.1 Two Mappings Between Words and Matrices

Now we introduce two mappings ψ and φ that give us an embedding from words to matrices. Let us consider the mapping ψ between $\{0,1\}^*$ and 2×2 matrices:

$$\psi : \epsilon \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = E \quad \psi : 0 \mapsto \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} = M_0 \quad \psi : 1 \mapsto \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} = M_1$$

$$\psi : w_1 \cdot \ldots \cdot w_r \mapsto M_{w_1} \times \ldots \times M_{w_r}.$$

It is a well known fact that the mapping ψ is an isomorphism between $\{0,1\}^*$ and elements of the matrix semigroup generated by 2×2 matrices M_0 and M_1. Since for every matrix with non-zero determinant there is only one unique inverse matrix, we can also define a similar mapping φ. It can be defined using inverse matrices of the semigroup generator. Mapping φ is also an isomorphism between $\{0,1\}^*$ and elements of the matrix semigroup generated by 2×2 matrices $\{M_0^{-1}, M_1^{-1}\}$:

$$\varphi : \epsilon \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = E \quad \varphi : 0 \mapsto \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} = M_0^{-1} \quad \varphi : 1 \mapsto \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix} = M_1^{-1}$$

$$\varphi : w_1 \cdot \ldots \cdot w_r \mapsto M_{w_1}^{-1} \times \ldots \times M_{w_r}^{-1}.$$

Note that the mappings from $w \in \{0,1\}^+$ to $w^{\mathrm{R}} \in \{0,1\}^+$ and from $\psi(u)$ to $\varphi(u^{\mathrm{R}})$ are bijective. Another very useful property of these mapping is that they can be used to define a free semigroup and group:

Proposition 1. *[4, 11] The semigroup or group generated by the pair of matrices $\psi(0)$, $\psi(1)$ is free.*

Moreover since the semigroup generated by $\{\psi(0), \psi(1)\}$ is free we can express an equality on words in terms of matrix equality:

Lemma 1. *[10] Given two words $u, v \in X^*$, $u = v$ iff $\varphi(u^{\mathrm{R}}) = (\psi(v))^{-1}$.*

From Lemma 1 and the fact that matrices $\psi(w)$ and $\varphi(w)$ have inverse matrices (elements) for any word $w \in \{0,1\}^*$, the following lemma holds:

Lemma 2. *[10] Given a binary alphabet X, a finite sequence of pairs of words in X^*:*

$$(u_1, v_1), \ldots, (u_k, v_k)$$

and a finite sequence of indexes $\{i_j\}$ with $\{i_j \in \{1..k\}\}$. The word $u = u_{i_1} \cdot \ldots \cdot u_{i_n}$ is equal to the word $v = v_{i_1} \cdot \ldots \cdot v_{i_n}$ if and only if

$$\varphi(u^{\mathrm{R}}) \times \psi(v) = E.$$

2.2 Reduced Words and Their Cyclic Permutation

Let $\Gamma = \{0, 1, 0^{-1}, 1^{-1}\}$ where $0 \equiv \varphi(0)$, $1 \equiv \varphi(1)$, $0^{-1} \equiv \psi(0)$ and $1^{-1} \equiv \psi(1)$. For any word, u, such that

$$u = y_1 \cdot y_2 \cdots y_n, \quad (y_i \in \Gamma)$$

we say that u is a reduced word if $y_i \neq y_{i+1}^{-1}$, $(1 \leq i < n)$, i.e u does not contain a subword of the type yy^{-1} for any character $y \in \Gamma$.

Let P be the semigroup generated by the matrices $\{\varphi(0), \varphi(1), \psi(0), \psi(1)\}$ shown above, using multiplication as the associative operator. We define an isomorphic mapping, ω, from any element of P to its reduced word in Γ:

$$\omega : X_{2,2} \mapsto \Gamma^* \qquad | \quad X \in P$$

We shall also define $|\omega(X)|$ to be the number of characters in $\omega(X)$ for any matrix X. It is clear that $\omega(E) = \varepsilon$ therefore $|\omega(E)| = 0$. Let $\omega_k(X)$ denote the kth symbol of $\omega(X)$ and $\omega_F(X)$ denote the final symbol of $\omega(X)$.

For any sequence $T = (t_1, t_2, \cdots, t_n)$ we define a k-cyclic permutation (or k-cyclic shift) to be the result of moving all elements to the right k times with elements overflowing from the right being inserted to the left. Thus if we shift this sequence k places to the right we get the sequence $\mathrm{suff}_k(T) \cdot T \cdot (\mathrm{suff}_k(T)^{-1})^R$ where $0 \leq k \leq n$.

3 PCP Encoding

In this section we show the idea of reducing the Post Correspondence Problem to the membership problem for an invertible diagonal matrix. Post's correspondence problem (in short, PCP) is formulated as follows: Given a finite alphabet X and a finite sequence of pairs of words in X^*: $(u_1, v_1), \ldots, (u_k, v_k)$. Is there a finite sequence of indexes $\{i_j\}$ with $\{i_j \in \{1..k\}\}$, such that $u_{i_1} \cdot \ldots \cdot u_{i_m} = v_{i_1} \cdot \ldots \cdot v_{i_m}$? PCP(n) denotes the same problem with a sequence of n pairs. Without loss of generality we assume that the alphabet X is binary.

Let us construct a generator of a matrix semigroup S. For an instance of the PCP with n pairs of words the generator contains $4n + 2$ different matrices.

1. For each pair (u_i, v_i), we will create four matrices:
   - Matrix of type 1: $U_{1,2}^{1,2} = \varphi(u^R), U_{3,4}^{3,4} = \varphi(0^i 1), U_5^5 = 1$
   - Matrix of type 2: $U_{1,2}^{1,2} = \varphi(u^R), U_{3,4}^{3,4} = \varphi(0^i 1), U_5^5 = 2$
   - Matrix of type 3: $V_{1,2}^{1,2} = \psi(v), V_{3,4}^{3,4} = \psi(0^i 1), V_5^5 = 1$
   - Matrix of type 4: $V_{1,2}^{1,2} = \psi(v), V_{3,4}^{3,4} = \psi(0^i 1), V_5^5 = 3$
2. A single matrix, M, where $M_{1,2}^{1,2} = E, M_{3,4}^{3,4} = \varphi(1), M_5^5 = 5$
3. A single matrix, N, where $N_{1,2}^{1,2} = E, N_{3,4}^{3,4} = \psi(1), N_5^5 = 7$

We assign zero to all matrix elements not defined above. Now we state the reduction lemma and give an example of such an encoding below.

Lemma 3. *An instance of PCP has a solution iff the corresponding semigroup S contains the matrix M_D*

$$M_D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 210 \end{pmatrix}.$$

Example 1. Given an instance of the PCP, $P = ((101, 1), (0, 01010))$. We will construct an example of how our coding will represent this problem in a semigroup and what a solution to the problem will look like.

We are given two separate 'tiles' and need to construct a solution to the PCP. We can see that P_1, P_2, P_1 is one such solution. We will have a semigroup generator G of $((4 * n) + 2) = 10$ matrices $G_1, G_2, \ldots, G_{10}$ where:

Matrix Number	Word part	Index Part	Factor part
Matrix 1 :	$G_1{}_{1,2}^{1,2} = \varphi(101)$	$G_1{}_{3,4}^{3,4} = \varphi(01)$	$G_1{}_{5,5}^{5,5} = 1$
Matrix 2 :	$G_2{}_{1,2}^{1,2} = \varphi(101)$	$G_2{}_{3,4}^{3,4} = \varphi(01)$	$G_2{}_{5,5}^{5,5} = 2$
Matrix 3 :	$G_3{}_{1,2}^{1,2} = \varphi(0)$	$G_3{}_{3,4}^{3,4} = \varphi(001)$	$G_3{}_{5,5}^{5,5} = 1$
Matrix 4 :	$G_4{}_{1,2}^{1,2} = \varphi(0)$	$G_4{}_{3,4}^{3,4} = \varphi(001)$	$G_4{}_{5,5}^{5,5} = 2$
Matrix 5 :	$G_5{}_{1,2}^{1,2} = \psi(1)$	$G_5{}_{3,4}^{3,4} = \psi(01)$	$G_5{}_{5,5}^{5,5} = 1$
Matrix 6 :	$G_6{}_{1,2}^{1,2} = \psi(1)$	$G_6{}_{3,4}^{3,4} = \psi(01)$	$G_6{}_{5,5}^{5,5} = 3$
Matrix 7 :	$G_7{}_{1,2}^{1,2} = \psi(01010)$	$G_7{}_{3,4}^{3,4} = \psi(001)$	$G_7{}_{5,5}^{5,5} = 1$
Matrix 8 :	$G_8{}_{1,2}^{1,2} = \psi(01010)$	$G_8{}_{3,4}^{3,4} = \psi(001)$	$G_8{}_{5,5}^{5,5} = 3$
Matrix 9(M) :	$G_9{}_{1,2}^{1,2} = E$	$G_9{}_{3,4}^{3,4} = \varphi(1)$	$G_9{}_{5,5}^{5,5} = 5$
Matrix 10(N) :	$G_{10}{}_{1,2}^{1,2} = E$	$G_{10}{}_{3,4}^{3,4} = \psi(1)$	$G_{10}{}_{5,5}^{5,5} = 7$

As stated before, a sequence from P giving a solution of PCP is 1,2,1 thus we can define the following matrix product $\Omega = G_9 \cdot G_2 \cdot G_3 \cdot G_1 \cdot G_{10} \cdot G_6 \cdot G_7 \cdot G_5$. We will now show this gives the requires form of a matrix.

Consider the word part of the matrix first, $W(\Omega) = E \cdot \varphi(101) \cdot \varphi(0) \cdot \varphi(101) \cdot E \cdot \psi(1) \cdot \psi(01010) \cdot \psi(1) = E$. Now consider the index part, $I(\Omega) = \varphi(1) \cdot \varphi(01) \cdot \varphi(001) \cdot \varphi(01) \cdot \psi(1) \cdot \psi(01) \cdot \psi(001) \cdot \psi(01) = E$. Finally we have the factorization part as a (integer) product, $I(\Omega) = 5 * 2 * 1 * 1 * 7 * 3 * 1 * 1 = 210$. This is indeed a matrix is of the required form and is a solution of the above PCP instance.

In the next section we prove Lemma 3 by showing the correctness of the presented reduction.

3.1 Correctness of the Reduction

Let S be a semigroup that is constructed by the above rules for an instance of the PCP problem. We start by showing the word equation coding in minor $M_{1,2}^{1,2}$. Given a sequence of pairs of words in a binary alphabet $A = \{0, 1\}$:

$$(u_1, v_1), \ldots, (u_n, v_n).$$

Let us construct the sequence of pairs of 2×2 matrices using two mappings φ and ψ: $(\varphi(u_1), \psi(v_1)), \ldots, (\varphi(u_n), \psi(v_n))$.

Instead of equation $u = v$ we would like to consider a concatenation of two words $u^R \cdot v$ that is a palindrome in the case where $u = v$. Now we show a matrix interpretation of this concatenation. We associate 2×2 matrix C with a word w of the form $u^R \cdot v$. Initially we can think that C is an identity matrix corresponding to an empty word. The extension of a word w by a new pair of words (u_r, v_r) (i.e. that gives us $w' = u_r^R \cdot w \cdot v_r$) corresponds to the following matrix multiplication

$$C_{w'} = C_{u_r^R \cdot w \cdot v_r} = \varphi(u_r^R) \times C_w \times \psi(v_r) \tag{1}$$

According to Lemma 2 $u = u_{i_1} \cdot \ldots \cdot u_{i_n} = v_{i_1} \cdot \ldots \cdot v_{i_n} = v$ for a finite sequence of indexes $\{i_j\}$ with $\{i_j \in \{1..k\}\}$ if and only if $\varphi(u^R) \times \psi(v)$ is equal to the identity matrix. So the question of the word equality can be reduced to the problem of finding a sequence of pairwise matrix multiplications that gives us the identity matrix. Note that not only an inverse palindrome but also all its cyclic permutations are equal to the identity element.

Lemma 4. *Any k-cyclic permutation of an inverse palindrome of length n is a concatenation of two distinct (i.e. non-overlapping) inverse palindromes when $1 \leq k < n$.*

Proof. We are given a word $w = w_1 \cdot w_2 \cdots w_n$ which is an inverse palindrome (i.e. it can be written as $w = z \cdot (z^R)^{-1}$).

For a k-cyclic shift of w we will get a word of the form:

$$w' = \text{suff}_k(w) \cdot w \cdot (\text{suff}_k(w)^R)^{-1} \qquad | \quad 1 \leq k < n$$

In an inverse palindrome, element w_1 is inverse to w_n and w_2 is inverse to w_{n-1} etc. we can see that any cyclic permutation simply changes the order of the multiplication from left to right (i.e. $w_1 \cdot w_n$ becomes $w_n \cdot w_1$). Each time we shift right, the first sub-word increases by size 2 whilst the sub-word on the right decreases by size 2. Thus any cyclic shift of an inverse palindrome gives either one or two inverse palindromes (depending whether $k = n$).

Now by the definition of an inverse palindrome, each opposite pair from the centre outwards is inverse to each other and thus in any such word all elements cancel to give $w = \epsilon$. For any k-cyclic shift, we get one or two sub words which are inverse palindromes. In terms of matrices, this means all such cyclic permutations produce the identity matrix.

Since we cannot control the order of a matrix product in the semigroup we cannot directly apply the idea of pairwise multiplication. So we show that it is possible to avoid the pairwise matrix multiplications problem by increasing the dimension from 2 to 5 using the idea of relative matrices for index encoding.

The idea is to design such associated "tiles" for the above matrices that they disallow any products that cannot be represented as pairwise multiplications. In particular, a sequence of "tiles" in an incorrect order preserves some parts that cannot be reduced later.

We show now that using two specially designed minors of 5 dimensional matrices, $M_{3,4}^{3,4}$ and M_5^5, we can guarantee such a property. It is easy to see that the minor M_5^5 controls the exact number of appearences of auxiliary matrices and the minimum number of main matrices to avoid an empty word solution. This is achieved by assigning unique prime values to some matrices and by employing the fundamental theorem of arithmetic regarding prime factorization.

We will now prove that the index coding will only result in the identity matrix in the case that the matrix multiplication is of the correct form. The initial conditions of the following lemma are satisfied by the prime factorization in the last diagonal element of each matrix.

Lemma 5. *Let S be a set containing matrices $M = \varphi(1)$, $N = \psi(1)$, $U_i = \varphi(0^i 1)$ and $V_i = \psi(0^i 1)$ where $1 \leq i \leq n$. Let P be a set of matrices where each member of P is the product of at least one U and V matrix and exactly one M and N matrix from set S. The identity matrix is a member of the set P iff it is a cyclic permutation of the following sequence:*

$$M \cdot U_{i_1} \cdot U_{i_2} \cdots U_{i_n} \cdot N \cdot V_{i_n} \cdots V_{i_2} \cdot V_{i_1}$$

Proof. $\Leftarrow$ From the definition of the lemma, we know there is 1 matrix of type M, 1 matrix of type N and at least 1 matrix of type U and V. Thus we will prove by induction that any multiplication of the above forms gives the identity element.
Let us prove the base case, when $n = 1$:

$$M \cdot U_{i_1} \cdot N \cdot V_{i_1} = \varphi(1) \cdot \varphi(0^{i_1} 1) \cdot \psi(1) \cdot \psi(0^{i_1} 1) = E$$

Let us consider a matrix multiplication of the form $U_{n+1} \cdot N \cdot V_{n+1}$:

$$\varphi(0^{n+1} 1) \cdot \psi(1) \cdot \psi(0^{n+1} 1) = \psi(1) = N \tag{1}$$

We now assume the inductive hypothesis that for any n:

$$M \cdot U_{i_1} \cdots U_{i_n} \cdot N \cdot V_{i_n} \cdots V_{i_1} = E$$

But since we showed in (1) that $U_{n+1} \cdot N \cdot V_{n+1} = N$, we can substitute this into the above expression to get the same result for $n+1$. Thus this product gives the identity matrix by the principle of induction.

We now prove that if the above form is equal to the identity matrix, all cyclic permutations are aswell.

We can clearly see that in terms of atomic matrices, the given form of matrix product is an inverse palindrome as defined in lemma 4. This can be seen by looking at an example:

$$M \cdot U_{i_1} \cdot U_{i_2} N \cdot V_{i_2} \cdot V_{i_1} = \varphi(10^{i_1}10^{i_2}1) \cdot \psi(10^{i_2}10^{i_1}1)$$

But as shown in lemma 4, any cyclic permutation of an inverse palindrome is equal to two smaller inverse palindromes (except for the trivial case where we cycle a multiple of the number of matrices and get the original word back). Further, we showed that inverse palindromes are clearly equal to the identity element. Thus any cyclic shift of the above sequence gives the identity matrix.
$\Rightarrow$ We now move to the reverse direction of the proof; proving that all identity elements must be a cyclical permutation of the following form:

$$M \cdot U_{i_1} \cdot U_{i_2} \cdots U_{i_n} \cdot N \cdot V_{i_n} \cdots V_{i_2} \cdot V_{i_1}$$

We have four different matrix "types". We shall show that these elements cannot produce the identity element in any way other than the above form.

We first consider only U and V matrices in a product. We define a sequence of matrices by $(Y_{i_1}, Y_{i_2}, \cdots, Y_{i_n})$ where $Y_i \in \{U_i, V_i\}$. For any k, $\omega_1(U_k) = 0$ and $\omega_F(U_k) = 1$. Similarly, $\omega_1(V_k) = 0^{-1}$ and $\omega_F(V_k) = 1^{-1}$. Therefore any multiplication of these matrices will not have any consecutive inverse pairs since $1 \cdot 0^{-1} \neq E$ and $1^{-1} \cdot 0 \neq E$. More formally,

$$\left| \omega \left(\prod_{k=1}^{n} Y_{i_k} \right) \right| = \sum_{k=1}^{n} |\omega(Y_{i_k})| \quad , Y_k \in \{U_k, V_k\} \tag{2}$$

We will now prove that if we have a matrix sequence containing any consecutive products $U_i \cdot V_j$ (similarly for $V_j \cdot U_i$), it will never be able to be fully cancelled using just one M and N matrix:

Let $X = U_i \cdot V_j$ where $1 \leq i, j < n$. Given that $X = \varphi(0^i1) \cdot \psi(0^j1)$ and $\omega(X) = 0^i1(0^{-1})^j1^{-1}$, we can see $\omega_1(X) = 0$. Since for all $Y \in S$, $\omega_F(Y) = 1$ or 1^{-1}, no matrix can be pre-multiplied to reduce the length of $\omega(X)$. Given that $\omega_F(X) = 1^{-1}$, we can *only* post multiply by M to reduce the length of $\omega(X)$ because $\omega_1(M) = 1$. Therefore $U_i \cdot V_j \cdot M = \varphi(0^i1) \cdot \psi(0^j)$ and $\omega(U_i \cdot V_j \cdot M) = 0^i1(0^{-1})^j$. Again, no matrix can be pre-multiplied, but we can post-multiply by some U_k since only $\omega_1(U) = 0$. We have three cases, where $(k < j)$, $(k = j)$ and $(k > j)$ giving matrices $\varphi(0^i1) \cdot \psi(0^{j-k}) \cdot \varphi(1)$, $\varphi(0^i1)$ and $\varphi(0^i1) \cdot \varphi(0^{k-j}1)$ respectively. Since ω_F of these three matrices equals 1 it can only cancel with an N matrix. In all cases, there are now either 0 or 0^{-1} symbols on the right of the product. It can be seen however that further multiplications by the remaining U, V matrices will not fully cancel any of these products.

A similar argument holds for $X = V_j \cdot U_i$. Therefore if any matrix sequence contains consecutive elements U_i, V_j or V_j, U_i, its product cannot equal the identity matrix.

We now consider a matrix M in a product. We have four cases to consider.

1. Given $\omega(U_i \cdot M) = 0^i 11$, we see that it cannot be reduced to zero size because only N cancels with the final symbol leaving U_i and (2) shows that using only U and V matrices never gives the identity matrix.
2. $M \cdot U_i$ is of the correct form as shown in the first part of the proof.
3. $V_i \cdot M$ is of the correct form as shown in the first part of the proof.
4. Given $\omega(M \cdot V_i) = 1(0^{-1})^i 1^{-1}$. We cannot post-multiply by any remaining matrix type to reduce the number of matrix elements. We can pre-multiply this product by N but this again leaves only U and V matrices. Pre-multiplying by V_j gives a product: $\omega(V_j \cdot M \cdot V_i) = (0^{-1})^{(j+i)} 1^{-1}$, equal to V_{j+i} which cannot reduce to zero length because the M matrix has been used and only M cancels with the last element.

Since each matrix type has an inverse, the same situation occurs with the N matrix. Therefore any matrix product containing $(U_i \cdot M)$, $(M \cdot V_i)$, $(V_i \cdot N)$ or $(N \cdot U_i)$ will never be able to result in the identity element.

Thus the sequence must of the following form to produce the identity matrix:

$$\cdots V_{i_1} \cdot M \cdot U_{j_1} \cdots U_{j_n} \cdot N \cdot V_{k_m} \cdots V_{k_1} \cdot M \cdot U_{l_1} \cdots U_{l_p} \cdot N \cdot V_{h_q} \cdots$$

This pattern can repeat indefinitely, but since we only have a single M and N matrix:

$$\begin{array}{l} V_{i_m} \cdots V_{i_1} \cdot M \cdot U_{j_1} \cdots U_{j_n} \cdot N \cdot V_{k_p} \cdots V_{k_1} \mid m, p \in \mathbb{Z}^+, n \in \mathbb{N} \\ U_{i_m} \cdots U_{i_1} \cdot N \cdot V_{j_1} \cdots V_{j_n} \cdot M \cdot U_{k_p} \cdots U_{k_1} \mid m, p \in \mathbb{Z}^+, n \in \mathbb{N} \end{array}$$

Since $\varphi(1)$ is only inverse to $\psi(1)$ and each U,V matrix sequence contains exactly one of these matrices, the number of U matrices must equal the number of V matrices, thus $m + p = n$.

For the first equation, let us define two sub-sequences of matrices $\alpha_1 = (V_{i_m}, \cdots, V_{i_1}, M, U_{j_1}, \cdots, U_{j_m})$ and $\alpha_2 = (U_{j_{(n-p+1)}}, \cdots, U_{j_n}, N, V_{k_p}, \cdots, V_{k_1})$.

Assume the first equation is equal to the identity element. Now assume the contrary that $\alpha_1 \neq M$. There is an equal number of U and V matrices and the U matrices follow the M matrix therefore the last matrix in the sequence must be a U matrix. ω_F of any U is always 1. If $\alpha_2 = N$ then it will cancel with this element but leave a non-identity element in α_1(since $\alpha_1 \neq M$). Thus $\alpha_2 \neq N$ and since it has an equal number of U and V matrices and the U matrices preceed all V matrices, the first matrix in the sequence must also be some U matrix. But as shown in (2) the product of U matrices only increases the size of the sequence. The reverse argument also holds if $\alpha_2 \neq N$. Thus the resulting matrix cannot be the identity element unless $\alpha_1 = M$ and $\alpha_2 = N$.

Define $\alpha_1[i]$ to be the i'th element of the sequence. Now we prove that $\alpha_1[(m+1)-k] \cdot \alpha_1[(m+1)+k] = E$ where $(0 \leq k \leq m)$. Let us assume by contradiction that their exists some k where $\alpha_1[(m+1)-k] \cdot \alpha_1[(m+1)+k] \neq E$, i.e. 2 opposite elements who are not inverse to each other. Therefore we have:

$$V_a \cdot M \cdot U_b = \psi(0^a 1) \cdot \varphi(1) \cdot \varphi(0^b 1) \quad \mid a \neq b$$

If $a > b$ then it will give a matrix $\psi(0^{a-b}) \cdot \varphi(1)$. Since M has been used and N is not in this sequence however, we only have U and V matrices which (2)

shows cannot reduce the length of $|\omega(\psi(0^{a-b}) \cdot \varphi(1))|$ (and $|M| = 1$). If $a < b$ then it gives a matrix $\varphi(0^{b-a}1)$ which is equal to $U_{(b-a)}$. Again, since M has been used, there are only U and V matrices left which cannot reduce to M giving a contradiction.

Similarly for α_2 we get opposite matrices produced which gives the same result that all opposite matrices from the central element (N) are inverse.

Thus in the first equation given above, it must be of the form:

$$V_{i_m} \cdots V_{i_1} \cdot M \cdot U_{i_1} \cdots U_{i_m} \cdots U_{i_n} \cdot N \cdot V_{i_n} \cdots V_{i_{n-m+1}}$$

We can clearly see that this is a cyclic permutation of the form given in the first part of the proof and the form of equation 2 is a cyclic permutation of equation 1, thus both must be equal to the identity matrix.

The proof of above lemma ends the proof of reduction from Lemma 3 since the PCP has a solution if and only if the semigroup S contains the matrix M_D. Thus the following Theorem holds:

Theorem 1. *Problem 2 is undecidable in dimension five.*

3.2 Reduction to Lower Dimensions

Now we can reduce the dimensions used and state some corollaries.

Corollary 1. *Problem 2 is undecidable in dimension* 4.

Proof. The element M_5^5, in our previous construction, is a scalar value and is commutative in all matrices since all other elements along row and column 5 are zero. Therefore we can multiply minor $M_{3,4}^{3,4}$ by M_5^5 and we will still preserve this value across the multiplication without changing the structure of multiplications of $M_{3,4}^{3,4}$.

Corollary 2. *Problem 3 is undecidable for linear transformations defined by a finite set of integral* 4×4 *matrices.*

Proof. In order to prove the undecidablity of Problem 3 we can show that the scalar matrix $M = 210 \cdot E_4$ is undecidable. We use the same idea as we did for Problem 2 with the only difference being that we extend the generator of the semigroup by the following matrix R:

$$\begin{pmatrix} 210 & 0 & 0 & 0 \\ 0 & 210 & 0 & 0 \\ 0 & 0 & 210 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is easy to see that the above matrix commutes with all other matrices in the semigroup, since the minor $M_{1,2,3}^{1,2,3}$ is a scalar matrix and M_4^4 is the identity element. On the other hand we cannot use more than one copy of matrix R since the determinant of any matrix from a semigroup that uses more than one copy

of R will be more than 210^4. So the matrix $M = 210 \cdot E_4$ is reachable if and only if the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 210 \end{pmatrix}$$

is reachable and does not use R, that in turn is undecidable.

In fact we can prove an even stronger claim that membership of any non-unimodular scalar matrix over rationals is undecidable in dimension four for rational matrix semigroups.

Corollary 3. *Given a semigroup S generated by a finite set of $n \times n$ matrices over rationals and a scalar $k \in \mathbb{Q}$ such that $k \neq 1$. It is undecidable to check whether the scalar matrix $k \cdot E$ belonds to S for any $n \geq 4$.*

Proof. We use Lemma 3 to show the undecidablity of the membership problem for a scalar matrix $k \cdot E_4$ by repeating the proof of Corollary 2 and introducing another matrix R that is now equal to

$$\begin{pmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & \frac{k}{210} \end{pmatrix}.$$

We now use the suggestion of an anonymous referee to use our technique with different bijections ψ and φ to get an undecidability result in dimension 3 with rational matrices.

Corollary 4. *The problem of determining if a matrix $210 \cdot E_3$ is a member of a multiplicative semigroup with rational matrices is undecidable in dimension 3*

Proof. We show this result by using a different mapping for ϕ and ψ which form a free semigroup. We change this mapping for the separate minors $M_{1,2}^{1,2}$ and $M_{2,3}^{2,3}$. The mapping is such that the central element M_2^2 is always equal to 1 for all but one of the generators. This allows us to merge the two smaller minors into a single 3*3 matrix.

$$\psi_{M_{1,2}^{1,2}}(0) = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \psi_{M_{1,2}^{1,2}}(1) = \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix} \varphi_{M_{1,2}^{1,2}}(0) = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{-1}{2} & 1 \end{pmatrix} \varphi_{M_{1,2}^{1,2}}(1) = \begin{pmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{pmatrix}$$

$$\psi_{M_{2,3}^{2,3}}(0) = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \psi_{M_{2,3}^{2,3}}(1) = \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix} \varphi_{M_{2,3}^{2,3}}(0) = \begin{pmatrix} 1 & \frac{-1}{2} \\ 0 & \frac{1}{2} \end{pmatrix} \varphi_{M_{2,3}^{2,3}}(1) = \begin{pmatrix} 1 & -1 \\ 0 & \frac{1}{2} \end{pmatrix}$$

All mappings from ϵ give E_2. We can see that any product of the $M_{1,2}^{1,2}$ mapping matrices will give a 1 in element M_2^2. Similarly, any product of the $M_{2,3}^{2,3}$ mapping matrices will give a 1 in element M_2^2. These matrices also form

a free group and we can therefore use them to embed the PCP problem within a 3*3 matrix as before. However, we must now multiply the whole matrix by the scalars 2,3,5,7 that were previously in M_5^5. This means we cannot use this method for an arbitrary scalar matrix but only specific ones (in this case we use the matrix $210 \cdot E$).

4 Conclusion and Some Remarks

In this paper we have proved that the membership problem of a particular invertible diagonal matrix is undecidable for a 4×4 integral matrix semigroup. Then as a corollary of this fact we have shown that the membership of a particular invertible scalar matrix is undecidable in dimension 4 for an integral matrix semigroup and in dimension 3 for rational matrix semigroup. Moreover we have shown that in dimension 4 the membership of any non-unimodular scalar matrix is undecidable for a rational matrix semigroup. The same problems for lower dimensions and the membership problem for an arbitrary diagonal matrix in a matrix semigroup are still open.

Acknowledgements

We would like to thank the anonymous referees for their helpful suggestions. Special thanks to the referee who proposed the idea of different bijections that helps to get the undecidability result for dimension 3.

References

1. J.Bestel and J.Karhumäki. Combinatorics on Words - A Tutorial, Bulletin of the EATCS, February 2003, 178 - 228
2. V.Blondel, J.Cassaigne, J.Karhumäki. Problem 10.3. Freeness of multiplicative matrix semigroups. Unsolved Problems in Mathematical Systems and Control Theory, Edited by V.Blondel and A.Megretski, 309-314
3. V. Blondel and J. Tsitsiklis. A survey of computational complexity results in systems and control. Automatica, 36, 2000, 1249-1274
4. R.Graham, D.Knuth, O.Patashnik. Concrete Mathematics. 2nd edition, Addison-Wesley, 1994
5. V.Halava, T.Harju. Mortality in Matrix Semigroups, American Mathematical Monthly, Vol 108 No. 7, (2001) 649-653
6. R.Kannan and R.Lipton. Polynomial-time algorithm for the orbit problem, Journal of the ACM (JACM), Volume 33 , Issue 4, 1986, 808 - 821
7. A.Lisitsa, I.Potapov. Membership and reachability problems for row-monomial transformations. MFCS, 2004, 623-634
8. C.Moore. Unpredictability and Undecidability in Dynamical Systems, *Physical Review Letters*, Vol.64, N. 20, 1990, 2354-2357
9. M.Paterson. Unsolvability in 3×3 matrices. Studies in Applied Mathematics, 49 (1970), 105-107
10. I.Potapov. From Post systems to the reachability problems for matrix semigroups and counter automata. 8th International Conference on Developments in Language Theory (DLT04), LNCS 3340, 345-356
11. http://planetmath.org/encyclopedia/FreeGroup.html

A Kleene Theorem for Languages of Words Indexed by Linear Orderings

Alexis Bès[1] and Olivier Carton[2]

[1] LACL, Université Paris XII-Val de Marne
bes@univ-paris12.fr
http://www.univ-paris12.fr/lacl/bes/
[2] LIAFA, Université Paris 7
Olivier.Carton@liafa.jussieu.fr
http://www.liafa.jussieu.fr/~carton/

Abstract. In a preceding paper, Bruyère and the second author introduced automata, as well as rational expressions, which allow to deal with words indexed by linear orderings. A Kleene-like theorem was proved for words indexed by countable scattered linear orderings. In this paper we extend this result to languages of words indexed by all linear orderings.

1 Introduction

One of the fundamental results in automata theory is Kleene's theorem [15] which asserts the equivalence between sets of words accepted by automata and set of words described by rational expressions. During the past fifty years Kleene's theorem has been extended to various notions of infinite words, as well as structures like trees, pictures, and traces.

In [3], Bruyère and Carton introduce automata and rational expressions for words on linear orderings. These notions unify naturally previously defined notions for finite words, left- and right-infinite words, bi-infinite words, and ordinal words. They also prove that a Kleene-like theorem holds when the orderings are restricted to countable scattered linear orderings; recall that a linear ordering is scattered if it does not contain any dense sub-ordering. This result extends Kleene's theorem for finite words [15], infinite words [5, 16], bi-infinite words [12, 17] and ordinal words [7, 9, 26]. Since [3], the study of automata on linear orderings was carried on in several papers, that address the emptiness problem and the containment problem for languages [8, 25], as well as the classification of rational languages with respect to the rational operations needed to describe them [4]. More recently, the second author and Rispal [20] proved that regular languages of words over countable scattered linear orderings are closed under complementation.

In this paper we come back to Kleene's theorem, and show that the assumption that the linear orderings are countable and scattered, is not necessary. When all linear orderings are considered instead of countable and scattered ones, no change has to be made to the notion of automata already introduced in [3].

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 158–167, 2005.
© Springer-Verlag Berlin Heidelberg 2005

However the set of operators for the rational expressions has to be extended in order to deal with words indexed by a dense linear ordering. To cope with this issue, we add a *shuffle* operator for languages, which is a variant of the classical shuffle operation on linear orderings. A similar (but not equivalent) notion of shuffle for languages was already considered in [10, 14, 23]. This operator allows to extend the definition of rational languages of words, and to prove a general Kleene-like theorem.

Words indexed by a countable linear ordering were first considered in [10], where they were introduced as frontiers of labeled binary trees. Some kind of rational expressions were studied in [10, 14, 23], which lead to a characterization of words which are frontiers of regular trees.

Other related works can be found in the area of specification and verification of real-time systems. Indeed, words indexed by $\mathbb{R}$ (or other linear orderings) appear as a simple and natural way to model the behavior of a finite state real-time system. For example, ordinal words (called Zeno words) were recently considered as modeling infinite sequences of actions which occur in a finite interval of time [2, 13]. While the intervals of time are finite, infinite sequences of actions can be concatenated. A Kleene's theorem already exists for standard timed automata (where infinite sequences of actions are supposed to generate divergent sequences of times) [1]. In [2], automata considered by Choueka and Wojciechowski are adapted to Zeno words. A kind of Kleene's theorem is proved, that is, the class of Zeno languages is the closure under an operation called refinement of the class of languages accepted by standard timed automata. More recently, Dima introduces a notion of real-time automata [11] that captures a class of timed languages which is closed under complementation and for which a Kleene's theorem is proved. Let us finally mention the paper [19] which introduces star-free expressions for words indexed by $\mathbb{R}$, and shows that star-free languages of words indexed by $\mathbb{R}$ coincide with languages definable in some first-order logic, extending McNaughton-Papert theorem.

The paper is organized as follows: we recall in Sect. 2 some useful definitions related to linear orderings. Sections 3 and 4 respectively introduce rational expressions and automata for words over linear orderings. Section 5 states the main theorem and provides a few examples.

2 Linear Orderings

In this section we recall useful definitions and results about linear orderings. A good reference on the subject is Rosenstein's book [21].

A *linear ordering* J is an ordering $<$ which is total, that is, for any $j \neq k$ in J, either $j < k$ or $k < j$ holds. Given a linear ordering J, we denote by $-J$ the *backwards* linear ordering obtained by reversing the ordering relation. For instance, $-\omega$ is the backwards linear ordering of ω which is used to index the so-called left-infinite words.

The sum of orderings is concatenation. let J and K_j for $j \in J$, be linear orderings. The linear ordering $\sum_{j \in J} K_j$ is obtained by juxtaposition of the or-

derings K_j with respect to J. More formally, the *sum* $\sum_{j\in J} K_j$ is the set L of all pairs (k, j) such that $k \in K_j$. The relation $(k_1, j_1) < (k_2, j_2)$ holds iff $j_1 < j_2$ or ($j_1 = j_2$ and $k_1 < k_2$ in K_{j_1}). The sum of two orderings K_1 and K_2 is denoted $K_1 + K_2$.

Two elements j and k of a linear ordering J are called *consecutive* if $j < k$ and if there is no element $i \in J$ such that $j < i < k$. An ordering is *dense* if it contains no pair of consecutive elements. More generally, a subset $K \subset J$ is *dense* in J if for any $j, j' \in J$ such that $j < j'$, there is $k \in K$ such that $j < k < j'$.

The notion of a cut is needed to define a path in an automaton. A *cut* of a linear ordering J is a pair (K, L) of intervals such that $J = K \cup L$ and such that for any $k \in K$ and $l \in L$, $k < l$. The set of all cuts of the ordering J is denoted by $\hat{J}$. This set $\hat{J}$ can be linearly ordered by the relation defined by $c_1 < c_2$ iff $K_1 \subsetneq K_2$ for any cuts $c_1 = (K_1, L_1)$ and $c_2 = (K_2, L_2)$. This linear ordering can be extended to $J \cup \hat{J}$ by setting $j < c_1$ whenever $j \in K_1$ for any $j \in J$.

The consecutive elements of $\hat{J}$ deserve some attention. For any element j of J, define two cuts c_j^- and c_j^+ by $c_j^-(K, \{j\} \cup L)$ and $c_j^+ = (K \cup \{j\}, L)$ where $K = \{k \mid k < j\}$ and $L = \{k \mid j < k\}$. It can be easily checked that the pairs of consecutive elements of $\hat{J}$ are the pairs of the form (c_j^-, c_j^+).

An ordering J is *complete* if for any cut (K, L) such that $K \neq \varnothing$ and $L \neq \varnothing$, either K has a greatest element or L has a least element.

3 Words and Rational Expressions

Given a finite alphabet A, a *word* $(a_j)_{j\in J}$ is a function from J to A which maps any element j of J to a letter a_j of A. We say that J is the *length* $|x|$ of the word x. For instance, the *empty word* ε is indexed by the empty linear ordering $J = \varnothing$. Usual finite words are the words indexed by finite orderings $J = \{1, 2, \ldots, n\}$, $n \geq 0$. A word of length $J\omega$ is usually called an ω-word or an infinite word. A word of length $\zeta = -\omega + \omega$ is a sequence $\ldots a_{-2}a_{-1}a_0a_1a_2 \ldots$ of letters which is usually called a bi-infinite word.

The sum operation on linear orderings leads to a notion of product of words as follows. Let J and K_j for $j \in J$, be linear orderings. Let $x_j = (a_{k,j})_{k\in K_j}$ be a word of length K_j, for any $j \in J$. The *product* $\prod_{j\in J} x_j$ is the word z of length $L \sum_{j\in J} K_j$ equal to $(a_{k,j})_{(k,j)\in L}$. For instance, the word $a^\zeta = b^{-\omega}a^\omega$ of length ζ is the product of the two words $b^{-\omega}$ and a^ω of length $-\omega$ and ω respectively.

We now recall the notion of rational set of words on linear orderings as defined in [3]. The rational operations include of course the usual Kleene operations for finite words which are the union $+$, the concatenation $\cdot$ and the star operation $*$. They also include the omega iteration ω usually used to construct ω-words and the ordinal iteration $\sharp$ introduced by Wojciechowski [26] for ordinal words. Four new operations are also needed: the backwards omega iteration $-\omega$, the backwards ordinal iteration $-\sharp$, a binary operation denoted $\diamond$ which is a kind of iteration for all orderings, and finally a shuffle operation which allows to deal with dense linear orderings.

Let us recall the already known rational operations. We respectively denote by $\mathcal{N}$, $\mathcal{O}$ and $\mathcal{L}$ the classes of finite orderings, the class of all ordinals and the class of all linear orderings. For an ordering J, we denote by $\hat{J}^*$ the set $\hat{J} \setminus \{(\varnothing, J), (J, \varnothing)\}$ where $(\varnothing, J)$ and $(J, \varnothing)\}$ are the first and last cut. Given two sets X and Y of words, define

$$\begin{aligned}
X + Y &= \{z \mid z \in X \cup Y\},\\
X \cdot Y &= \{x \cdot y \mid x \in X, y \in Y\},\\
X^* &= \{\textstyle\prod_{j\in\{1,\dots,n\}} x_j \mid n \in \mathcal{N}, x_j \in X\},\\
X^\omega &= \{\textstyle\prod_{j\in\omega} x_j \mid x_j \in X\},\\
X^{-\omega} &= \{\textstyle\prod_{j\in-\omega} x_j \mid x_j \in X\},\\
X^\sharp &= \{\textstyle\prod_{j\in\alpha} x_j \mid \alpha \in \mathcal{O}, x_j \in X\},\\
X^{-\sharp} &= \{\textstyle\prod_{j\in-\alpha} x_j \mid \alpha \in \mathcal{O}, x_j \in X\},\\
X \diamond Y &= \{\textstyle\prod_{j\in J\cup\hat{J}^*} z_j \mid J \in \mathcal{L}, z_j \in X \text{ if } j \in J \text{ and } z_j \in Y \text{ if } j \in \hat{J}^*\}.
\end{aligned}$$

We use the notation $X^\diamond$ as an abbreviation for $(X \diamond \varepsilon) + \varepsilon$.

We now define the new shuffle operation which is needed to deal with dense orderings.

Definition 1. *Let A be a finite alphabet, $n \geq 1$, and $L_1, \dots, L_n \subseteq A^\diamond$. We define the shuffle of $L_1, \dots, L_n$, and denote by $\mathrm{sh}(L_1, \dots, L_n)$ the set of words $w \in A^\diamond$ that can be written as $w = \prod_{j\in J} w_j$ where*

- *J is a complete linear ordering without first and last element;*
- *there exists a partition $(J_1, \dots, J_n)$ of J such that all J_i's are dense in J, and for every $j \in J$, if $j \in J_k$ then $w_j \in L_k$.*

Let us remark that our definition of shuffle slightly differs from others, e.g. from [14, 23], because in Definition 1 we assume that J is a *complete* dense ordering.

Only countable orderings are considered in [14, 23]. Recall that $\mathbb{Q}$ is the unique countable and dense ordering without first and last element. Their definition of the shuffle operation is based on a partition $(J_1, \dots, J_n)$ of $\mathbb{Q}$ into dense subsets $J_1, \dots, J_n$. Then points of each J_i are substituted by words from L_i as we do. Our definition is not a straightforward generalization of this shuffle because $\mathbb{Q}$ is of course not complete. Actually the assumption that J is complete yields a more general notion of shuffle. The completion of $\mathbb{Q}$ yields the ordering $\mathbb{R}$. If each point of $\mathbb{R} \setminus \mathbb{Q}$ is substituted by the empty word, one obtains a shuffle in the sense of [14, 23]. This shows that our rational expression $\mathrm{sh}(L_1, \dots, L_n, \varepsilon)$ corresponds to the shuffle of languages $L_1, L_2, \dots, L_n$ in the sense of [14, 23].

An abstract *rational expression* is a well-formed term of the free algebra over $\{\varnothing\} \cup A$ with the symbols denoting the rational operations as function symbols. Each rational expression denotes a set of words which is inductively defined by the above definitions of the rational operations. A set of words is *rational* if it can be denoted by a rational expression. As usual, the dot denoting concatenation is omitted in rational expressions.

Example 1. Consider the word $w = (w_r)_{r\in\mathbb{R}}$ of length $\mathbb{R}$ over the alphabet $A = \{a, b\}$, defined by $w_r = a$ if $r \in \mathbb{Q}$, and $w_r = b$ otherwise. Then it is not difficult to check that $w \in \mathrm{sh}(a, b)$. Consider now the word $w' = (w'_q)_{q\in\mathbb{Q}}$ of length $\mathbb{Q}$ over the alphabet A, defined by $w'_q = a$ if $q \in \{m/2^n \mid m \in \mathbb{Z}, n \in \mathbb{N}\}$, and $w'_q = b$ otherwise. It can be checked that $w' \in \mathrm{sh}(a, b, \varepsilon)$ but that $w' \notin \mathrm{sh}(a, b)$ because $\mathbb{Q}$ is not complete.

Example 2. The rational expression $a^*(\varepsilon + \mathrm{sh}(a^*, \varepsilon))a^*$ denotes the set of words (over the unary alphabet $\{a\}$) whose length is an ordering containing no infinite sequence of consecutive elements. It is clear that the length of any word denoted by this expression cannot contain an infinite sequence of consecutive elements. Conversely, let J be such an ordering. Define the equivalence relation $\sim$ on J by $x \sim y$ iff there are finitely many elements between x and y. The classes of $\sim$ are then finite intervals. Furthermore the ordering of these intervals must be a dense ordering with possibly a first and a last element. This completes the converse.

Example 3. The rational expression $(\varepsilon + \mathrm{sh}(a)) \diamond a$ denotes the set of words (over the unary alphabet $\{a\}$) whose length is a complete ordering. The shuffle operator is defined using complete orderings and the ordering $\hat{J}$ is always complete. It follows from these two facts that the length of any word denoted by this expression is complete. Conversely, let J be a complete orderings. Define the equivalence relation $\sim$ on J by $x \sim y$ iff there is an open dense interval containing both x and y. Each class of $\sim$ is either a singleton or an open dense interval. Let K be the ordering of the singleton classes and let L_0 be the ordering of the dense classes. Let L_1 be the ordering of pairs of consecutive elements in K and let L be $L_0 \cup L_1$ equipped with the natural ordering. It can be shown that $K\hat{L}$. This gives the expression $(\varepsilon + \mathrm{sh}(a)) \diamond a$ where ε is due to L_1, $\mathrm{sh}(a)$ to L_0 and a to K. This completes the converse.

4 Automata

In this section, we recall the definition given in [3] for automata accepting words on linear orderings. As already noted in [3], this definition is actually suitable for all linear orderings.

Automata accepting words on linear orderings are classical finite automata equipped with limit transitions. They are defined as $\mathcal{A}(Q, A, E, I, F)$, where Q denotes the finite set of states, A is a finite alphabet, and I, F denote the set of initial and final states, respectively. The set E consists in three types of transitions: the usual *successor* transitions in $Q \times A \times Q$, the *left limit* transitions which belong to $2^Q \times Q$ and the *right limit* transitions which belong to $Q \times 2^Q$. A left (respectively right) limit transition $(P, q) \in 2^Q \times Q$ (respectively, $(q, P) \in Q \times 2^Q$) will usually be denoted by $P \to q$ (respectively $q \to P$).

We sometimes write that an automaton $\mathcal{A}$ has transitions $P_1, \ldots, P_m \to q_1, \ldots, q_n$ when $\mathcal{A}$ has all left limit transitions $P_i \to q_j$ for $1 \le i \le m$ and $1 \le j \le n$. Analogously we shall use the notation $q_1, \ldots, q_n \to P_1, \ldots, P_m$ for right limit transitions.

A word $x = (a_j)_{j \in J}$ of length J is accepted by $\mathcal{A}$ if it is the label of a successful path. A *path* γ is a sequence of states $\gamma = (q_c)_{c \in \hat{J}}$ of length $\hat{J}$ verifying the following conditions. For two consecutive states in γ, there must be a successor transition labeled by the letter in between. For a state $q \in \gamma$ which has no predecessor on γ, there must be a left limit transition $P \to q$ where P is the limit set of γ on the left of q. Right limit transitions are used similarly when q has no successor on γ.

Observe that the ordering $\hat{J}$ always has a first element and a last element, namely the cuts $c_{\min} = (\varnothing, J)$ and $c_{\max}(J, \varnothing)$. For any cut $c \in \hat{J}$, define the sets $\lim_{c^-} \gamma$ and $\lim_{c^+} \gamma$ as follows:

$$\lim_{c^-} \gamma = \{q \in Q \mid \forall c' < c \ \ \exists k \quad c' < k < c \text{ and } q = q_k\},$$
$$\lim_{c^+} \gamma = \{q \in Q \mid \forall c < c' \ \ \exists k \quad c < k < c' \text{ and } q = q_k\}.$$

A sequence $\gamma = (q_c)_{c \in \hat{J}}$ of states is an accepting path for the word $x = (a_j)_{j \in J}$ if the following conditions are fulfilled. For any pair (c_j^-, c_j^+) of consecutive cuts of J, the automaton must have the successor transition $q_{c_j^-} \xrightarrow{a_j} q_{c_j^+}$. For any cut $c \neq c_{\min}$ which has no predecessor in $\hat{J}$, $\lim_{c^-} \gamma \ \to \ q_c$ must be a left limit transition. For any cut $c \neq c_{\max}$ in $\hat{J}$ which has no successor, $q_c \to \lim_{c^+} \gamma$ must be a right limit transition. A path is *successful* if its first state $q_{c_{\min}}$ is initial and its last state $q_{c_{\max}}$ is final.

Example 4. Let $A = \{a, b\}$. The automata $\mathcal{A}$ pictured in Fig. 1 has two successor transitions, three left limit transitions and three right limit transitions. State 0 is the only initial state, and state 5 is the only final state.

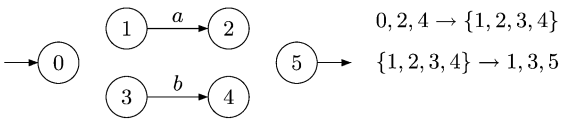


Fig. 1. Automaton accepting $\mathrm{sh}(a, b)$

Let us show that this automata accepts words in $\mathrm{sh}(a, b)$. Consider indeed a word $w = (w_j)_{j \in J}$, and assume first that w is accepted by $\mathcal{A}$. Let $\gamma = (q_c)_{c \in \hat{J}}$ be a successful path labeled by w. The ordering J must be dense since there are no consecutive transitions in $\mathcal{A}$. It must also be complete since there is no state with incoming left limit transitions and leaving right limit transitions. Occurrences of both a and b must be dense in J since all limit transitions involve the four states $\{1, 2, 3, 4\}$. Finally, J cannot have a first or a last element. Indeed, the only transition leaving state 0 is a limit one, and similarly for the only transition entering state 5.

Conversely, let $w = (w_j)_{j \in J}$ be a word indexed by a complete ordering J such that occurrences of both a and b are dense in J. Since J is complete, any

cut of J (apart from $c_{\min}$ and $c_{\max}$) are either preceded or followed by a letter. Then the sequence $\gamma = (q_c)_{c \in \hat{J}}$ defined as follows is a successful path labeled by w.

- $q_{c_{\min}} = 0$,
- $q_{c_{\max}} = 5$,
- $q_c = 1$ if c is followed by an a and $q_c = 2$ if c is preceded by an a,
- $q_c = 3$ if c is followed by a b and $q_c = 4$ if c is preceded by a b.

The construction of an automaton accepting $\mathrm{sh}(L_1, \ldots, L_n)$ from automata accepting $L_1, \ldots, L_n$ is a straightforward generalization of the automaton for $\mathrm{sh}(a, b)$ which is pictured in Fig. 1.

5 Rational Expressions vs. Automata

In this section we state the main theorem of the paper, i.e. that automata and rational expressions define the same languages of words over linear orderings.

Theorem 1. *A set of words over linear orderings is rational iff it is recognizable.*

This result was proved in [3] for the restricted case of countable scattered linear orderings. Many arguments still hold in the general case. As in [3], the *only if* part of the proof relies upon an induction on the rational expression. The only modification with respect to the proof of [3] is that we have to show that the shuffle of rational languages is rational. For the *if* part of the proof, we use as in [3] Yamada's classical technique, i.e. an induction on the set of states visited by a successful path of the automaton. A key ingredient is the use of successive condensations of linear orderings.

We illustrate the theorem with a few examples, over the alphabet $A = \{a, b\}$.

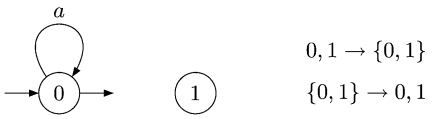


Fig. 2. Automaton accepting $a^*(\varepsilon + \mathrm{sh}(a^*, \varepsilon))a^*$

Example 5. The automaton pictured in Fig. 2 accepts the set $a^*(\varepsilon + \mathrm{sh}(a^*, \varepsilon))a^*$ of words already considered in Example 2.

Example 6. The automaton pictured in Fig. 3 accepts words over $\{a, b\}$ whose length is a complete ordering. Since all limit transitions enter state 0 and leave state 2, the length of an accepted word must be complete. Conversely, let x be a word of length J where J is a complete ordering. Define the path γ which maps any cut (K, L) of $\hat{J}$ to 0 if K has no greatest element, to 2 if L has no least element and to 1 otherwise. It is pure routine to check that this defines an accepting path for x.

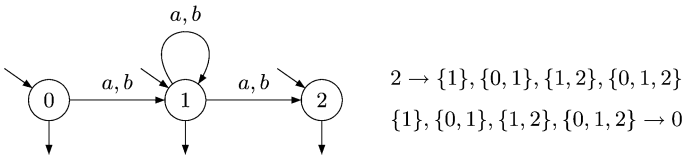


Fig. 3. Automaton accepting words with a complete length

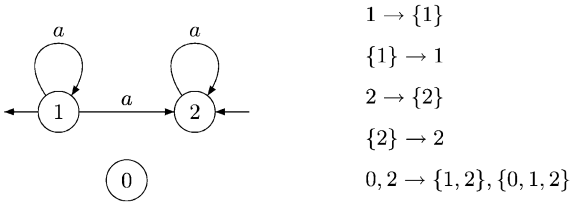


Fig. 4. Automaton accepting words with a non scattered length

Example 7. The automaton pictured in Fig. 4 accepts words over $\{a\}$ whose length is a non scattered ordering. Let $w = (w_j)_{j \in J}$ be a word labeling a successful path γ in $\mathcal{A}$. Let K be the set of positions j such that w_j is read by the transition $1 \xrightarrow{a} 2$ in γ. It can be checked that K is a dense subordering of J. Therefore, the ordering J is not scattered.

For the converse, recall that each linear ordering J can be written $J = \sum_{k \in K} J_k$ where each ordering J_k is scattered and the ordering K is either the one-element ordering if J is scattered or a dense ordering [21, chap. 4]. From this decomposition, a word w whose length is not scattered is equal to a product $\prod_{k \in K} w_k$ where K is a dense ordering. Then a path γ_k from state 1 to state 2 labeled by w_k can be constructed as follows. Let J_k be the length of w_k and let z_k be an arbitrarily chosen element of J_k. Any cut of J_k before z_k is mapped to state 1 and any cut after z_k is mapped to state 2. A path labeled by w is finally constructed as follows. Any cut inside some w_k is mapped to the corresponding state in γ_k and any remaining gap is mapped to state 0.

6 Conclusion and Open Questions

We considered rational expressions and automata for words indexed by linear orderings, and prove that these two formalisms capture the same languages.

A natural question is whether the class of recognizable languages is closed under complementation. It has been proved in [20] that the answer is positive when one considers only words indexed by countable and scattered orderings (the proof relies upon semigroup theory). However the answer is negative in the general case: one can prove that the set of words indexed by a non scattered ordering is recognizable, while its complement is not.

The connections between automata over linear orderings and logic would be interesting to explore. In his seminal paper [5], Büchi proved that recognizable languages of finite words coincide with languages definable in the weak monadic second order theory of $(\omega, <)$, which allowed him to prove decidability of this theory. In [6] he proved that a similar equivalence holds between recognizable languages of infinite words of length ω and languages definable in the monadic second order theory of $(\omega, <)$. The result was then extended to languages of words indexed by a countable ordinal [7]. What can be said about monadic second order theories for linear orderings beyond ordinals? Using the automata technique, Rabin proved in [18] decidability of the monadic second order theory of the binary tree, from which he deduces decidability of the monadic second order theory of $\mathbb{Q}$, which in turn implies decidability of the monadic second order theory of countable linear orderings. On the other hand, Shelah [22] improved model-theoretical techniques [24] that allow him to reprove almost all known decidability results about monadic second order theories, as well as new decidability results for the case of linear orderings. On the other hand he proved that the monadic second order theory of the real line is undecidable. Shelah's decidability method is model-theoretical, and up to now no corresponding automata techniques are known. This led Thomas to ask [24] whether there is an appropriate notion of automata for words indexed by linear orderings beyond the ordinals. As mentioned in [3], this question was an important motivation for the introduction of automata considered in the present paper. It would be interesting to provide a logical characterization of recognizable languages of words over linear orderings.

References

1. E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 160–171, 1997.
2. B. Bérard and C. Picaronny. Accepting Zeno words without making time stand still. In *Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lect. Notes in Comput. Sci.*, pages 149–158, 1997.
3. V. Bruyère and O. Carton. Automata on linear orderings. In J. Sgall, A. Pultr, and P. Kolman, editors, *MFCS'2001*, volume 2136 of *Lect. Notes in Comput. Sci.*, pages 236–247, 2001.
4. V. Bruyère and O. Carton. Hierarchy among automata on linear orderings. In R. Baeza-Yate, U. Montanari, and N. Santoro, editors, *Foundation of Information technology in the era of network and mobile computing*, pages 107–118. Kluwer Academic Publishers, 2002.
5. J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik und grundl. Math.*, 6:66–92, 1960.
6. J. R. Büchi. On a decision method in the restricted second-order arithmetic. In *Proc. Int. Congress Logic, Methodology and Philosophy of science, Berkeley 1960*, pages 1–11. Stanford University Press, 1962.
7. J. R. Büchi. Transfinite automata recursions and weak second order theory of ordinals. In *Proc. Int. Congress Logic, Methodology, and Philosophy of Science, Jerusalem 1964*, pages 2–23. North Holland, 1965.

8. O. Carton. Accessibility in automata on scattered linear orderings. In K.Diks and W.Rytter, editors, *MFCS'2002*, volume 2420 of *Lect. Notes in Comput. Sci.*, pages 155–164, 2002.
9. Y. Choueka. Finite automata, definable sets, and regular expressions over ω^n-tapes. *J. Comput. System Sci.*, 17(1):81–97, 1978.
10. B. Courcelle. Frontiers of infinite trees. *RAIRO Theoretical informatics*, 12(4):319–337, 1978.
11. C. Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6(1):3–24, 2001.
12. D. Girault-Beauquier. Bilimites de langages reconnaissables. *Theoret. Comput. Sci.*, 33(2–3):335–342, 1984.
13. M. R. Hansen, P. K. Pandya, and Z. Chaochen. Finite divergence. *Theoret. Comput. Sci.*, 138(1):113–139, 1995.
14. S. Heilbrunner. An algorithm for the solution of fixed-point equations for infinite words. *RAIRO Theoretical informatics*, 14(2):131–141, 1980.
15. S. C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon, editor, *Automata studies*, pages 3–41. Princeton university Press, Princeton, 1956.
16. D. E. Muller. Infinite sequences and finite machines. In *Switching Circuit Theory and Logical Design: Proc. Fourth Annual Symp.*, pages 3–16. I.E.E.E., New York, 1963.
17. M. Nivat and D. Perrin. Ensembles reconnaissables de mots bi-infinis. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 47–59, 1982.
18. M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
19. A. Rabinovich. Star free expressions over the reals. *Theoret. Comput. Sci.*, 233(1–2):233–245, 2000.
20. C. Rispal and O. Carton. Complementation of rational sets on countable scattered linear orderings. In C. S. Calude, E. Calude, and M. J. Dinneen, editors, *DLT'2004*, volume 3340 of *Lect. Notes in Comput. Sci.*, pages 381–392, 2004.
21. J. G. Rosenstein. *Linear orderings*. Academic Press, New York, 1982.
22. S. Shelah. The monadic theory of order. *Annals of Mathematics*, 102:379–419, 1975.
23. W. Thomas. On frontiers of regular sets. *RAIRO Theoretical informatics*, 20:371–381, 1986.
24. W. Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science, A Selection of Essays in Honor of A. Ehrenfeucht*, number 1261 in Lect. Notes in Comput. Sci., pages 118–143. Springer-Verlag, 1997.
25. O. Carton V. Bruyère and G. Sénizergues. Tree automata and automata on linear orderings. In T. Harju and J. Karhumäki, editors, *WORDS'2003*, pages 222–231. Turku Center for Computer Science, 2003.
26. J. Wojciechowski. Finite automata on transfinite sequences and regular expressions. *Fundamenta informaticæ*, 8(3-4):379–396, 1985.

Revolving-Input Finite Automata

Henning Bordihn[1], Markus Holzer[2], and Martin Kutrib[3]

[1] Institut für Informatik, Universität Potsdam,
August-Bebel-Straße 89, D-14482 Potsdam, Germany
henning@cs.uni-potsdam.de
[2] Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching bei München, Germany
holzer@informatik.tu-muenchen.de
[3] Institut für Informatik, Universität Giessen,
Arndtstraße 2, D-35392 Giessen, Germany
kutrib@informatik.uni-giessen.de

Abstract. We introduce and investigate revolving-input finite automata, which are nondeterministic finite automata with the additional ability to shift the remaining part of the input. We consider three different modes of shifting, namely revolving to the left, revolving to the right, and circular interchanging. We show that the latter operation does not increase the computational power of finite automata, even if the number of revolving operations is unbounded. The same result is obtained for the former two operations in case of an arbitrary but constant number of applications allowed. An unbounded number of these operations leads to language families that are properly contained in the family of context-sensitive languages, are incomparable with the family of context-free languages, and are strictly more powerful than regular languages. Moreover, we show that right revolving can be simulated by left revolving, when considering the mirror image of the input.

1 Introduction

Finite automata are probably best known for capturing the family of regular languages. These machines have been intensively studied and moreover, have been extended in various ways. Examples are pushdown automata [2, 3] or variants of stack automata [5, 7]. The results obtained for these classes of machines hold for a large variety of classes of automata, when appropriately abstracted. This led to the rich theory of abstract families of automata (AFA), which is the equivalent of the theory of abstract families of languages (AFL); for the general treatment of machines and languages we refer, e.g. to [4].

We introduce finite automata with the ability to shift the unread input circularly. From a more general point of view, these machines are finite automata equipped with an additional operation on the input. Typical formal language theoretical operations are, for instance, reversal or shift operations on words. Recently, automata with reversals have been investigated in several papers. These papers led to the devices of flip-pushdown automata [11], the "flip-pushdown

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 168–179, 2005.
© Springer-Verlag Berlin Heidelberg 2005

input-reversal" theorem [8], and input-reversal automata [1]. It is worth mentioning that these devices induce a hierarchy of languages based on the number of operations allowed during the computation. Loosely speaking, it was shown that $k+1$ pushdown flips are better than k for both deterministic and nondeterministic flip-pushdown automata [8]. A similar statement has been proven in the context of input-reversal automata [1]. Moreover, input-reversal automata have been shown to be deeply linked to controlled linear context-free languages [6], leading to an alternative characterization of Khabbaz hierarchy of languages [9, 10]. But what about the aforementioned shift operation on the input? Are shift operations stronger than reversal operations? If not, which are sets of operations that do lead to certain stronger language families? Which are sets of operations that are complete in this sense? In the following we distinguish three different modes of shifting: revolving to the left, revolving to the right, and circular interchanging (cf. Fig. 1).

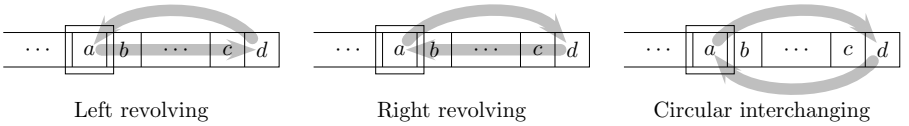


Fig. 1. Circular input operations

Obviously, if the number of operations applied is zero, the family of regular languages is characterized. We show that this remains true for circular interchanging, even if the number of revolving operations is unbounded, and that it remains true for revolving to the left or right as long as the number of shift operations is arbitrarily constant. In case the number of shift operations is not bounded by a constant, revolving finite automata induce language families which contain all regular languages, are incomparable (with respect to set inclusion) with context-free languages, and are properly contained in the family of context-sensitive languages. To be more precise, we prove that left- and right-revolving finite automata are somehow asymmetric, since we can show that every linear context-free language can be accepted by a left-revolving finite automaton, but, for example, the linear context-free language $\{ a^n b^n \mid n \geq 0 \}$ cannot be accepted by any right-revolving finite automaton. Moreover, right revolving can be simulated by left revolving, when considering the mirror image of the input. On the other hand, by pumping arguments we obtain that the family of languages accepted by left-revolving finite automata and the family of languages accepted by right-revolving finite automata are incomparable. The used pumping arguments are somehow more involved compared to ordinary pumping arguments on finite automata, since revolving automata may shift certain symbols around, thus seeing a symbol more than once during a computation. As a byproduct, it is shown that neither the family of left-revolving finite automata languages nor the family of right-revolving finite automata languages is closed under intersection with regular sets. This is quite surprising, since we are dealing with language families defined by automata.

The paper is organized as follows: The next section contains preliminaries and basics on revolving finite automata. Then Section 3 deals with the computational power of revolving finite automata in general. After that left- and right-revolving finite automata are compared in more detail. Finally, we summarize our results and present some open questions in Section 5.

2 Preliminaries

We denote the powerset of a set S by 2^S. The empty word is denoted by λ, the reversal of a word w by w^R, and for the length of w we write $|w|$. For the number of occurrences of a symbol a in w we use the notation $|w|_a$. Set inclusion and strict set inclusion are denoted by $\subseteq$ and $\subset$, respectively.

In the following we consider finite automata with the ability to shift the unread input circularly. We may start with a uniform definition.

Definition 1. *A* (nondeterministic) extended finite automaton *is a* 6*-tuple* $A = (Q, \Sigma, \delta, \Delta, q_0, F)$*, where* Q *is a finite set of states,* Σ *is the input alphabet,* δ *is a mapping from* $Q \times (\Sigma \cup \{\lambda\})$ *to* 2^Q *called the transition function,* Δ *is a mapping from* $Q \times \Sigma$ *to* 2^Q*,* $q_0 \in Q$ *is the initial state, and* $F \subseteq Q$ *is the set of accepting states. Furthermore,* A *is said to be* λ-free*, if* δ *obeys* $\delta : Q \times \Sigma \to 2^Q$*.*

The different modes are formally distinguished by different interpretations of the mapping Δ. To this end, we consider *configurations* of extended finite automata to be tuples (q, w), where $q \in Q$ is the current state and $w \in \Sigma^*$ is the yet to be consumed part of the input, where the leftmost letter of w currently scanned. If a is in $\Sigma \cup \{\lambda\}$ and w in Σ^*, then we write $(q, aw) \vdash_A (p, w)$, if p is in $\delta(q, a)$, for "ordinary" finite automata transitions.

A revolving operation is performed by applying the mapping Δ (cf. Fig. 1). For $a, b \in \Sigma$ and $w \in \Sigma^*$,

1. a *left-revolving transition* is defined by $(q, a) \vdash_A (p, a)$ and $(q, awb) \vdash_A (p, baw)$, for p in $\Delta(q, a)$,
2. a *right-revolving transition* is defined by $(q, aw) \vdash_A (p, wa)$, for p in $\Delta(q, a)$,
3. and a *circular-interchanging transition* is defined by $(q, a) \vdash_A (p, a)$ and $(q, awb) \vdash_A (p, bwa)$, for p in $\Delta(q, a)$.

An extended finite automaton $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ with left-revolving, right-revolving, or circular-interchanging transitions is called a *left-revolving finite automaton (left-RFA)*, *right-revolving finite automaton (right-RFA)*, or *circular-interchanging finite automaton (CIFA)*, respectively.

As a natural generalization of left-RFA and right-RFA, we also consider bi-revolving finite automata, where both left- and right-revolving transitions are possible. Formally, a 7-tuple $A = (Q, \Sigma, \delta, \Delta_\ell, \Delta_r, q_0, F)$ is a *bi-revolving finite automaton (bi-RFA),* where $A = (Q, \Sigma, \delta, \Delta_\ell, q_0, F)$ is a left-RFA and $A = (Q, \Sigma, \delta, \Delta_r, q_0, F)$ is a right-RFA. Whenever we refer to an automaton as *revolving finite automaton* it is either left-, right-, bi-revolving or circular-interchanging.

For any revolving finite automaton, whenever there is a choice between an ordinary or another transition, the automaton nondeterministically chooses the next move. As usual, the reflexive transitive closure of $\vdash_A$ is denoted by $\vdash_A^*$. The subscript A will be dropped from $\vdash_A$ and $\vdash_A^*$ whenever the meaning remains clear. Let k be a non-negative integer. We define the language *accepted with at most k non-ordinary steps* to be $T_k(A) = \{\, w \in \Sigma^* \mid (q_0, w) \vdash_A^* (q, \lambda)$ with at most k non-ordinary steps and $q \in F \,\}$. If the number of non-ordinary steps is not bounded, the language accepted is analogously defined as above and denoted by $T(A)$. In order to clarify our notation we give an example. In what follows, when specifying an automaton we will list only those transitions which do not map to the empty set.

Example 2. Let $A = (\{q_0, q_a, q_b\}, \{a, b\}, \delta, \Delta, q_0, \{q_0\})$ be a right-revolving finite automaton, where

1. $\delta(q_0, a) = \{q_b\}$
2. $\delta(q_0, b) = \{q_a\}$
3. $\delta(q_a, a) = \{q_0\}$
4. $\delta(q_b, b) = \{q_0\}$
5. $\Delta(q_a, b) = \{q_a\}$
6. $\Delta(q_b, a) = \{q_b\}$

Automaton A accepts the context-free language $L = \{\, w \in \{a, b\}^* \mid |w|_a = |w|_b \,\}$: The transitions (1) and (2) allow A to store the currently read input letter in the finite control in order to search for a corresponding mate letter. Whenever a corresponding mate is found A uses either transitions (3) or (4) to return to the initial state. Being in a search state, all non-mate letters are shifted through to the end of the input word. This is done with the transitions (5) and (6). Thus, the input satisfies the property $|w|_a = |w|_b$, when the automaton reaches the accepting state. It is easy to see that the above given automaton, when defined as a left-revolving automaton, accepts the same language.

Example 3. Let $A = (\{q_0, q_a, q_b, q'_a, q'_b\}, \{a, b\}, \delta, \Delta, q_0, \{q_0\})$ be a left-revolving finite automaton, where

1. $\delta(q_0, a) = \{q_a\}$
2. $\delta(q_0, b) = \{q_b\}$
3. $\delta(q'_a, a) = \{q_0\}$
4. $\delta(q'_b, b) = \{q_0\}$
5. $\Delta(q_a, a) = \{q'_a\}$
6. $\Delta(q_a, b) = \{q'_a\}$
7. $\Delta(q_b, a) = \{q'_b\}$
8. $\Delta(q_b, b) = \{q'_b\}$

Automaton A accepts the linear context-free language $L = \{\, ww^R \mid w \in \{a, b\}^* \,\}$: The transitions (1) and (2) allow A to store the currently read input letter in the finite control in order to search for a corresponding mate letter, which must be at the end of the input word. Then with transitions (5) through (8) the letter at the end of the input is revolved to the left, and with transitions (3) and (4) it is verified. Then the search process is started all over again.

The next theorem shows that λ-moves do not increase the computational power of revolving finite automata. The proof is a modification of the standard proof for finite automata. So, in the sequel we may consider λ-free automata for convenience.

Theorem 4. *Let k be a non-negative integer. For any revolving finite automaton A one can construct a λ-free revolving finite automaton B, such that $T_k(A) = T_k(B)$. The statements remain true if an unbounded number of revolving steps is allowed.* □

3 Revolving Finite Automata

This section is devoted to some basic results on revolving finite automata that reveal, roughly, the positions of the accepted language families in the Chomsky hierarchy. Our first results concern the bottom of the hierarchy. The following theorem shows that providing finite automata with an unbounded number of circular-interchanging operations does not increase their computational capacity.

Theorem 5. *A language L is accepted by a circular-interchanging finite automaton if and only if L is regular.*

Proof. The implication from left to right is obvious. The converse implication basically follows from the following observation. Whenever a circular-interchanging finite automaton performs a circular-interchanging operation it interchanges the currently read and the last letter in the input. So, once the last letter is known, a simulating automaton can remember the current last letter in its finite control in order to behave correctly. Moreover, initially, the last input letter can be guessed and, finally, the guess can be verified. Formally, let $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ be a λ-free circular-interchanging finite automaton. A language-equivalent nondeterministic finite automaton that may perform λ-moves $B = (Q', \Sigma, \delta', q_0', F')$ is constructed as follows.

Let $\hat{Q}$ and $\tilde{Q}$ be disjoint copies of Q, then $Q' = \Big((Q \cup \hat{Q}) \times \Sigma \times \Sigma\Big) \cup \{q_0'\} \cup \tilde{Q}$ and $F' = \{\, \tilde{q} \in \tilde{Q} \mid q \in F \,\}$. The transition function δ' is specified as follows:

1. Define $\delta'(q_0', \lambda) = \{\, (q_0, c, c) \mid c \in \Sigma \,\}$.
2. For all $q \in Q$, $a, c \in \Sigma$, $b \in \Sigma$, let $\delta'((q, a, c), b) \supseteq \{\, (p, a, c) \mid p \in \delta(q, b) \,\}$.
3. For all $q \in Q$, $a, c \in \Sigma$, let $\delta'((q, a, c), c) \supseteq \{\, \tilde{p} \mid p \in \delta(q, c) \,\}$.
4. For all $q \in Q$, $a, b, c \in \Sigma$, let $\delta'((q, a, c), b) \supseteq \{\, (\hat{p}, a, c) \mid p \in \Delta(q, b) \,\}$.
5. For all $\hat{q} \in \hat{Q}$, $a, b, c \in \Sigma$, let $\delta'((\hat{q}, a, c), b) \supseteq \{\, (p, a, c) \mid p \in \Delta(q, a) \,\}$.
6. For all $\hat{q} \in \hat{Q}$, $a, b, c \in \Sigma$, let $\delta'((\hat{q}, a, c), b) \supseteq \{\, (p, b, c) \mid p \in \delta(q, a) \,\}$.

No further transitions than listed above are contained in δ'. In the states of the form (q, a, c), q represents the current state of A, the second component a stores the input letter which would become leftmost after a circular-interchanging operation was applied, whereas the third component c serves for memorizing the initially guessed last input letter. Transition (1) implements the initial guess of the last input letter. Subsequently, ordinary moves of A are simulated by transitions of type (2). The last move in any accepting computation has to be an ordinary one. Transitions of type (3) switch to states from $\tilde{Q}$ if the currently read input letter matches the initially guessed last input letter. Since subsequently the computation is blocked and due to the construction of F', they implement the verification of the guess. Transitions (4) to (6) are for the simulation of circular-interchanging operations and the subsequent ordinary moves. After a simulated circular-interchanging operation the currently read input letter has to be remembered as the new last letter, whereas the formerly remembered last input letter becomes the letter for the next transition. □

So, circular-interchanging finite automata are yet another characterization of the regular languages. In general, the situation is completely different for left-, right-, and bi-revolving finite automata. But in the specific case where the number of operations is bounded by an arbitrary constant, again, the computational power is not increased.

Theorem 6. *Let k be a non-negative integer. A language L is accepted by a revolving finite automaton A with at most k revolving steps, i.e., $T_k(A) = L$, if and only if L is regular.*

Proof. For circular-interchanging finite automata the statement holds by Theorem 5. Since left- and right-revolving automata are particular types of revolving automata, it is sufficient to provide the proof for bi-revolving automata.

The implication from right to left is immediate. Conversely, we argue as follows: Let $A = (Q, \Sigma, \delta, \Delta_\ell, \Delta_r, q_0, F)$ be a bi-revolving finite automaton. By induction on k we show that the language $T_k(A)$ is regular. If $k = 0$ the statement is obviously true. Now consider a word w in $T_{k+1}(A)$. Then the first revolving step on w is a left- or right-revolving move. In the former case we find a decomposition $w = ua$ or $w = uavb$ with $u, v \in \Sigma^*$ and $a, b \in \Sigma$ such that $(q_0, w) = (q_0, ua) \vdash_A^* (q, a) \vdash_A (p, a) \vdash_A^* (q_f, \lambda)$, where $(q_0, ua) \vdash_A^* (q, a)$ is a computation without any revolving move, $p \in \Delta_\ell(q, a)$, $(p, a) \vdash_A^* (q_f, \lambda)$ is a computation with exactly k revolving moves, and $q_f \in F$, or $(q_0, w) = (q_0, uavb) \vdash_A^* (q, avb) \vdash_A (p, bav) \vdash_A^* (q_f, \lambda)$, where $(q_0, uavb) \vdash_A^* (q, avb)$ is a computation without any revolving move, $p \in \Delta_\ell(q, a)$, $(p, bav) \vdash_A^* (q_f, \lambda)$ is a computation with exactly k revolving moves, and $q_f \in F$. In the latter case, the decomposition of w reads as $w = uav$ with $u, v \in \Sigma^*$ and $a \in \Sigma$ such that $(q_0, w) = (q_0, uav) \vdash_A^* (q, av) \vdash_A (p, va) \vdash_A^* (q_f, \lambda)$, where $(q_0, uav) \vdash_A^* (q, a)$ is a computation without any revolving move, $p \in \Delta_r(q, a)$, $(p, va) \vdash_A^* (q_f, \lambda)$ is a computation with exactly k revolving moves, and $q_f \in F$. Thus, the language $T_{k+1}(A)$ can be expressed as the union of the following languages

$$L_{1,1} = \bigcup_{q \in Q} \bigcup_{a \in \Sigma} \bigcup_{p \in \Delta_\ell(q,a)} T_0(A_{q_0,q}) \cdot \{\, a \in \Sigma \mid a \in T_k(A_{p,q_f}) \text{ with } q_f \in F \,\}$$

$$L_{1,2} = \bigcup_{q \in Q} \bigcup_{a,b \in \Sigma} \bigcup_{p \in \Delta_\ell(q,a)} T_0(A_{q_0,q}) \cdot a \cdot \{\, v \in \Sigma^* \mid bav \in T_k(A_{p,q_f}) \text{ with } q_f \in F \,\} \cdot b$$

and

$$L_2 = \bigcup_{q \in Q} \bigcup_{a \in \Sigma} \bigcup_{p \in \Delta_r(q,a)} T_0(A_{q_0,q}) \cdot a \cdot \{\, v \in \Sigma^* \mid va \in T_k(A_{p,q_f}) \text{ with } q_f \in F \,\},$$

where $A_{p,q} = (Q, \Sigma, \delta, \Delta_\ell, \Delta_r, p, \{q\})$ is the bi-revolving finite automaton defined from A. Then by induction hypothesis, all sets involved are regular. Since regular sets are closed under concatenation and left- and right-quotient, it follows that language $T_{k+1}(A)$ is regular, too. □

Whenever the number of revolving moves is not restricted to be constant, then we find the following upper bound.

Theorem 7. *Every language accepted by a revolving finite automaton is context sensitive and belongs to NP.* □

Obviously, *unary* languages accepted by revolving finite automata are regular since a left- or right-revolving move does not change the remaining part of the input. Therefore, it can be omitted. More formally the statement reads as follows:

Theorem 8. *A unary language L is accepted by a revolving finite automaton if and only if L is regular.* □

An immediate consequence is that the inclusion in the context-sensitive languages and in NP is proper. But there is also a non-unary, somehow easy, language not accepted by any revolving finite automaton.

Theorem 9. *There is a deterministic, two-linear context-free language, which cannot be accepted by any revolving finite automaton.*

Proof. Consider the language $L = \{\, b^m a^m b a^n b^n \mid m, n \geq 1 \,\}$. Assume that there is a bi-revolving finite automaton $A = (Q, \Sigma, \delta, \Delta_\ell, \Delta_r, q_0, F)$ accepting L. Let $|Q| = n$ and consider the first n transitions of an accepting computation of A on input $b^{n+1}a^{n+1}ba^{2n}b^{2n}$. Without loss of generality, we may assume that A is λ-free and that the computation contains no (useless) loops. The first n transitions may be ordinary reading transitions, or left- or right-revolving transitions in any ordering. In any case, only symbols b are affected. By the pigeonhole principle, at least one state of Q, say p, is repeated, say in steps i and j, where $i < j \leq n$. Assume that until step i there occur k_1 reading transitions, ℓ_1 left-revolving and r_1 right-revolving transitions. Set $c_1 = \ell_1 - r_1$ to be the difference of the number of b's that are revolved from the suffix to the prefix and *vice versa*. Accordingly, from step i to step j let k_2 be the number of occurring reading transitions, ℓ_2 be the number of left-revolving and r_2 be the number of right-revolving transitions. Set $c_2 = \ell_2 - r_2$. Clearly, c_1 and c_2 can be negative numbers. Observe, that $k_1 + r_1 + l_1 + k_2 + r_2 + l_2 \leq n$. Thus, we have a computation

$$\begin{aligned}(q_0, b^{n+1}a^{n+1}ba^{2n}b^{2n}) &\vdash_A^* (p, b^{n+1+c_1-k_1}a^{n+1}ba^{2n}b^{2n-c_1}) \vdash_A^* \\ &(p, b^{n+1+c_1-k_1+c_2-k_2}a^{n+1}ba^{2n}b^{2n-c_1-c_2}) \vdash_A^* (q_f, \lambda)\end{aligned}$$

where $q_f \in F$. We conclude $c_2 = 0$. Otherwise, the computation

$$\begin{aligned}(q_0, b^{n+1+c_2-k_2}a^{n+1}ba^{2n}b^{2n-c_2}) &\vdash_A^* \\ &(p, b^{n+1+c_2-k_2+c_1-k_1}a^{n+1}ba^{2n}b^{2n-c_2-c_1}) \vdash_A^* (q_f, \lambda)\end{aligned}$$

accepts a word not in L. For similar reasons we conclude $c_2 - k_2 = 0$ which in turn implies $k_2 = 0$. Together this is a contradiction to the assumption, since it follows either $i = j$ or automaton A loops. □

4 Left- *Versus* Right-Revolving Finite Automata

In this section we consider left- and right-revolving finite automata with an unbounded number of revolving moves in more detail. Trivially, every regular language can be accepted by a left- or right-revolving finite automaton and moreover, the inclusion is known to be strict by Example 2. We summarize this statement in the following theorem.

Theorem 10. *(1) Let L be a regular language. Then L is accepted by a left- or right-revolving finite automaton A, i.e., $T(A) = L$. (2) There is a language accepted by a left- or right-revolving finite automaton, which cannot be accepted by any ordinary finite automaton.* □

The next theorem shows that for left-revolving finite automata we can improve the lower bound showing that every linear context-free language can be accepted.

Theorem 11. *Let L be a linear context-free language. Then L is accepted by a left-revolving finite automaton A, i.e., $T(A) = L$.*

Proof. Let $G = (N, T, P, S)$ be a linear context-free grammar in normal-form, i.e., every production is either of the form $A \to a$, $A \to aB$, or $A \to Ba$, for $A, B \in N$, $B \neq S$ and $a \in T$; additionally, $S \to \lambda$ is admitted, if $\lambda \in L(G)$. We construct a left-revolving finite automaton $A = (Q, T, \delta, \Delta, S, F)$, where $Q = N \cup \{ A' \mid A \in N \} \cup \{\Lambda\}$ the union being disjoint, $F = \{\Lambda\}$, if $\lambda \notin L(G)$, and $F = \{\Lambda, S\}$, if $\lambda \in L(G)$, and δ and Δ are specified as follows: For every A in N and a in T let $\delta(A, a) = \{ B \in N \mid (A \to aB) \in P \} \cup \{ \Lambda \mid (A \to a) \in P \}$ and $\delta(A', a) = \{ B \in N \mid (A \to Ba) \in P \}$. Moreover, for every A in N and a in T define $\Delta(A, a) = \{A'\}$.

Except for the fact that S serves as initial state it is unreachable. Hence, $\lambda \in T(A)$ if and only if $\lambda \in L(G)$. By easy means one observes that $A \Rightarrow aB$ if and only if $(A, au) \vdash_A (B, u)$ and $A \Rightarrow Ba$ if and only if $(A, ua) \vdash_A (A', au) \vdash_A (B, u)$, for $A, B \in N$, $a \in T$, and $u \in T^*$. Similar statements are valid in case of termination. This immediately implies that every word generated by the linear context-free grammar G is accepted by the left-revolving automaton A and *vice versa*. Thus $L(G) = T(A)$. □

By Example 2, the above given inclusion is strict, since the non-linear-context-free language $L = \{ w \in \{a, b\}^* \mid |w|_a = |w|_b \}$ is accepted by a left-revolving finite automaton. Thus, we have shown the following corollary.

Corollary 12. *There is a language L accepted by a left-revolving finite automaton, which cannot be generated by any linear context-free grammar.* □

The following theorem shows that a left-revolving finite automaton can simulate a right-revolving finite automaton provided that the input is reversed. Later we will see that the converse relation is not true in general.

Theorem 13. *Let L be accepted by a right-revolving finite automaton A, i.e., $L = T(A)$. Then the reversal of L can be accepted by some left-revolving finite automaton B, i.e., $T(B) = L^R$, where $L^R = \{ w^R \mid w \in L \}$.*

Proof. Let $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ be a λ-free right-revolving finite automaton accepting the language L. We construct a left-revolving finite automaton accepting L^R as follows: Define $B = (Q', \Sigma, \delta', \Delta', q_0'', F')$ with $Q' = Q \cup \{ q' \mid q \in Q \} \cup \{q_0'', q_f\}$, the unions being disjoint, and δ' and Δ' are specified as follows:

1. For all $q \in Q$, $a \in \Sigma$, let $q_f \in \delta'(q, a)$, if $\delta(q, a) \cap F \neq \emptyset$.
2. For all $a \in \Sigma$, let $q_0 \in \Delta'(q_0'', a)$.
3. For all $q \in Q$, $a \in \Sigma$, let $p' \in \delta'(q, a)$, if $p \in \delta(q, a)$.
4. For all $q' \in Q'$, $a \in \Sigma$, let $q \in \Delta'(q', a)$.
5. For all $p \in Q$, $a \in \Sigma$, let $\Delta'(p, a) = \Delta(p, a)$.

Finally, let $F' = \{q_f\}$, if $\lambda \notin L$, and $F' = \{q_0'', q_f\}$, if $\lambda \in L$. This completes the description of the left-revolving finite automaton B. It remains to prove the correctness of the simulation.

Note that q_0'' cannot be reached in a positive number of moves. Therefore, due to the construction of F' the empty word is accepted by B if and only if it is accepted by A. For words of length at least one we argue as follows: Let $a, b \in \Sigma$ and $w \in \Sigma^*$. On input $v = (aw)^R$ the simulation starts with

$$(q_0'', v) = (q_0'', (aw)^R) = (q_0'', w^R a) \vdash_B (q_0, aw^R)$$

using the transitions from (2). Then in order to simulate an ordinary transition move of the form $(q, abw) \vdash_A (p, bw)$ automaton B uses transitions from (3) and (4) leading to $(q, a(bw)^R) = (q, aw^R b) \vdash_B (p', w^R b) \vdash_B (p, bw^R)$. Observe, that a transition $(q, a) \vdash_A (p, \lambda)$ is simulated by $(q, a) \vdash_B (q_f, \lambda)$ in case $p \in F$ or by $(q, a) \vdash_B (p', \lambda)$, which cannot lead to acceptance anymore. Here transitions from (1) are used in the former case, while transitions from (3) lead to a blocking computation. Finally, the right-revolving move $(q, abw) \vdash_A (p, bwa)$ is mimicked with $(q, a(bw)^R) = (q, aw^R b) \vdash_B (p, baw^R) = (p, b(wa)^R)$ using transitions from (5). This shows that every word accepted by A is accepted in its reversed form by B. Conversely, similar statements are applicable. The proof is quite similar. Thus, the left-revolving automaton B accepts the reversed language L^R. □

Next we show that there is a language accepted by a left-revolving finite automaton, which cannot be accepted by any right-revolving finite automaton.

Theorem 14. *There is a language L accepted by a left-revolving finite automaton, which cannot be accepted by any right-revolving finite automaton.*

Proof. Consider the linear context-free language $L = \{ a^n b^n \mid n \geq 0 \}$, which by Theorem 11 is acceptable by a left-revolving finite automaton. Assume that L is accepted by a λ-free right-revolving finite automaton $A = (Q, \{a, b\}, \delta, \Delta, q_0, F)$.

The proof is done in two steps: Let $n = |Q|$. First we show that the number of ordinary steps reading a sequence of a's between two consecutive revolving

moves is bounded by n. This is obvious, because otherwise one state is repeated at least once due to the pigeon hole principle. Thus, cutting this loop leads to a valid computation. Therefore, whenever the original word is accepted, also the new word induced by the cut loop is also accepted. Since after the cutting the number of a's is not equal to the number of b's on the input the automaton accepts a word not of the appropriate form. Therefore, in the forthcoming we may assume that the automaton A fulfills the above mentioned property.

Second, consider an accepting computation of the right-revolving automaton A on input $w = a^{n(n+1)+1}b^{n(n+1)+1}$. Because of the above mentioned fact, there are at least $n+1$ positions where a right-revolving move is started by reading a letter a. Because of the pigeon hole principle we find a state, say p, which appears at least twice. Thus, starting the computation in state q_0 with input w, the first appearance of state p is reached by i ordinary moves and j right-revolving moves (inter-winded), with $0 \leq j < n+1$. Hence we have

$$(q_0, w) = (q_0, a^{n(n+1)+1}b^{n(n+1)+1}) \vdash_A^* (p, a^{n(n+1)+1-i-j}b^{n(n+1)+1}a^j).$$

Then from the latter configuration state p is reached a second time by k ordinary moves and ℓ right-revolving moves (inter-winded) with $1 \leq \ell \leq (n+1) - j$. Therefore we find

$$(p, a^{n(n+1)+1-i-j}b^{n(n+1)+1}a^j) \vdash_A^* (p, a^{n(n+1)+1-i-j-k-\ell}b^{n(n+1)+1}a^j a^\ell).$$

Since we are considering an accepting configuration, there is a state $q_f \in F$ such that $(p, a^{n(n+1)+1-i-j-k-\ell}b^{n(n+1)+1}a^{j+\ell}) \vdash_A^* (q_f, \lambda)$. Observe, that $j+\ell \leq (n+1)$ and $i+j+k+\ell \leq n(n+1)$. Now we can fool the automaton A by constructing an accepting computation for the word

$$w' = a^{n(n+1)+1-k-\ell}b^{n(n+1)+1}a^\ell$$

by cutting out the above considered loop in the computation. For this word we have the accepting computation

$$\begin{aligned}(q_0, w') = (q_0, a^{n(n+1)+1-k-\ell}b^{n(n+1)+1}a^\ell) &\vdash_A^* \\ (p, a^{n(n+1)+1-i-j-k-\ell}b^{n(n+1)+1}a^\ell a^j) &= \\ (p, a^{n(n+1)+1-i-j-k-\ell}b^{n(n+1)+1}a^{j+\ell}) &\vdash_A^* (q_f, \lambda)\end{aligned}$$

of A. Since the constructed word w' is not a member of L we obtain a contradiction. Thus no right-revolving finite automaton can accept the considered linear context-free language. □

The language used in the above given proof shows that the converse of Theorem 13, i.e., replacing left-revolving by right-revolving and *vice versa*, is not true in general. Thus, we can state the following corollary.

Corollary 15. *(1) There is a language L accepted by a left-revolving finite automaton, such that the reversal of L, i.e the language L^R, cannot be accepted by any right-revolving finite automaton. (2) There is a language accepted by a bi-revolving finite automaton, which cannot be accepted by any right-revolving finite automaton.* □

In the remainder of this section we compare the computational powers of right- and left-revolving automata in more detail.

Lemma 16. *(1) The families of languages accepted by right-revolving finite automata and their reversals on the one hand and the family of linear context-free languages on the other hand are incomparable. (2) The families of languages accepted by right-revolving finite automata and their reversals, left-revolving finite automata, and bi-revolving finite automata on one hand and the families of deterministic context-free, two-linear context-free, and context-free languages on the other hand are incomparable.*

Proof. By a straightforward extension of Example 2 one can construct a right- or left-revolving finite automaton that accepts the non-context-free language $L = \{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \}$, satisfying $L = L^R$. Conversely, (1) by Theorem 14 neither the linear context-free language $L_1 = \{ a^n b^n \mid n \geq 0 \}$ nor its reversal is accepted by any right-revolving finite automaton. (2) By Theorem 9 there is a deterministic, two-linear context-free language which cannot be accepted by any revolving finite automaton. □

Without proof we state the following theorem.

Theorem 17. *There is a language accepted by a right-revolving finite automaton, which cannot be accepted by any left-revolving finite automaton.* □

Basically, the proof relies on the witness language

$$\{ w \mid w = uacv, u \in \{a,b\}^*, v \in \{a,b,c\}^*, |w|_a = |w|_b = |w|_c \}$$

and the fact that a left-revolving finite automaton initially must not read too many symbols of u in order to compare the numbers of different symbols, and thus cannot check whether subword u is followed by symbol a. The above given theorem immediately implies the following separation result.

Corollary 18. *(1) There is a language accepted by a bi-revolving finite automaton, which cannot be accepted by any left-revolving finite automaton. (2) The families of languages accepted by left- and right-revolving finite automata are incomparable.* □

Finally, the presented witness languages in Example 2 and in the proofs of Theorems 14 and 17 show even more, namely the non-closure under intersection with regular sets.

Corollary 19. *Neither (1) the family of languages accepted by left-revolving finite automata nor (2) the family of languages accepted by right-revolving finite automata are closed under intersection with regular sets.* □

5 Conclusions

We have studied the power of revolving finite automata, that are finite machines equipped with the additional ability to shift the unread part of the input. The proven inclusion relations are summarized in Figure 2 – where $\mathcal{L}$(bi-RFA) ($\mathcal{L}$(left-RFA), $\mathcal{L}$(right-RFA), $\mathcal{L}$(CIFA), respectively) denotes the

family of languages which are accepted by bi-revolving (left-revolving, right-revolving, circular-interchanging, respectively) finite automata and the family $\mathcal{L}^R$(right-RFA) is equal to $\{ L^R \mid L \in \mathcal{L}(\text{right-RFA}) \}$. Moreover, CSL, CFL, 2-LIN, LIN, and REG refer to the families of context-sensitive, context-free, two-linear context-free, linear context-free, and regular languages, respectively. All shown inclusions are strict and language families that are not linked by a path are pairwise incomparable. Nevertheless, several questions for revolving automata remain unanswered. We mention three of them: (1) How do deterministic and nondeterministic revolving automata language families relate to each other? (2) What is the relationship between these language families and other well-known formal language classes? (3) What is the computational power of revolving pushdown or stack automata?

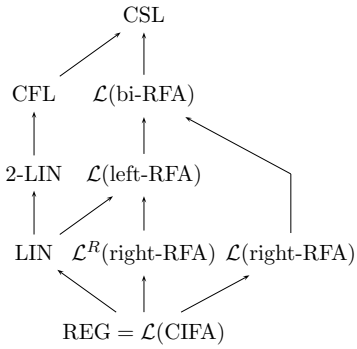


Fig. 2. Inclusion structure

References

1. H. Bordihn, M. Holzer, and M. Kutrib. *Input reversals and iterated pushdown automata – a new characterization of Khabbaz geometric hierarchy of languages*. Developments in Language Theory (DLT 2004), LNCS 3340, 2004, pp. 102–113.
2. N. Chomsky. *Handbook of Mathematic Psychology*, volume 2, chapter Formal Properties of Grammars, pp. 323–418, Wiley & Sons, New York, 1962.
3. R. J. Evey. *The Theory and Applications of Pushdown Store Machines*. Ph.D thesis, Harvard University, Massachusetts, 1963.
4. S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. North-Holland, Amsterdam, 1975.
5. S. Ginsburg, S. A. Greibach, and M. A. Harrison. *One-way stack automata*. Journal of the ACM 14 (1967), 389–418.
6. S. Ginsburg and E. H. Spanier. *Control sets on grammars*. Mathematical Systems Theory 2 (1968), 159–177.
7. S. A. Greibach. *An infinite hierarchy of context-free languages*. Journal of the ACM 16 (1969), 91–106.
8. M. Holzer and M. Kutrib. *Flip-pushdown automata: $k+1$ pushdown reversals are better than k*. International Colloqium on Automata, Languages and Programming (ICALP 2003), LNCS 2719, 2003, pp. 490–501.
9. N. A. Khabbaz. *Control sets and linear grammars*. Information and Control 25 (1974), 206–221.
10. N. A. Khabbaz. *A geometric hierarchy of languages*. Journal of Computer and System Sciences 8 (1974), 142–157.
11. P. Sarkar. *Pushdown automaton with the ability to flip its stack*. Report TR01-081, Electronic Colloquium on Computational Complexity (ECCC), 2001.

Some New Results on Palindromic Factors of Billiard Words

Jean-Pierre Borel[1,⋆] and Christophe Reutenauer[2]

[1] LACO, UMR CNRS 6090, 123 avenue Albert Thomas,
F-87060 Limoges Cedex, France
borel@unilim.fr
[2] UQÀM, Département de Mathématiques, case postale 8888,
succursale Centre-Ville, Montreal (Québec), H3C 3P8, Canada
christo@math.uqam.ca

Abstract. For heuristic reasons billiard words may have more palindromic factors than any other words. Many results are already known, concerning the palindromic factors and the palindromic prefixes of Sturmian words and billiard words on two letters. We give general results concerning multidimensional billiard words, which describe very different situations. In some cases, these words have arbitrary long palindromic prefix factors. In other cases, at the opposite, they have finitely many distinct palindromic factors.

Keywords: Words, languages, Sturmian, Billiard, palindromes.
AMS classification: 68R15.

1 Introduction

1.1 Palindromic Factors

We consider finite or infinite words on the finite alphabet $\mathcal{A} := \{a_1, a_2, \ldots, a_k\}$, with $k \geq 2$. In the following we use a geometrical approach of words, and k can be viewed as the *dimension* of the words. A finite word v is called a *palindromic word* when it is equal to its reversal. We denote by $\widetilde{v}$ the reversal of word v.

We can make two preliminary remarks:

1. The palindromic property corresponds to a symmetry property of the finite word. An infinite word on $\mathcal{A}$ corresponds to a trajectory, or a curve, in the k-dimensional space $\mathbb{R}_+^k$, so that infinite words with many palindromic factors may correspond to curves which are locally invariant by many central symmetries. The best candidates are lines, which correspond to the so-called *billiard words*.
2. Letters are trivial palindromic words of length 1. In dimension 2, any word of length bigger than 2 has a non trivial palindromic factor. In the other sense, in any higher dimension, there exist infinite words without any palindromic factors, e.g. $(a_1a_2a_3)^\infty$.

⋆ Research partially supported by Région Limousin

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 180–188, 2005.
© Springer-Verlag Berlin Heidelberg 2005

1.2 Billiard Words

There exist many different ways to define billiard words, which are special cases of general Sturmian words, especially in dimension 2. Here we choose the geometrical one. We consider a vector $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_k)$ with α_i positive. Let $\mathcal{D}$ be the half-line of origin O and parallel to $(\alpha_1, \alpha_2, \ldots, \alpha_k)$. We construct the *billiard word*, or *cutting sequence*, c_α as follow.

1.2.1 Billiard Words in Dimension $k = 2$

There are three different ways to define c_α, see Fig.1.a:

1. by looking at the horizontal and vertical segments on the grid $\mathcal{G}$, which is the set of vertical half-lines with integer x-coordinate and of horizontal half-lines with integer y-coordinate. We denote by a_1 the horizontal segement and by a_2 the vertical one. Then we encode the discrete path immediately under the half-line $\mathcal{D}$, and we obtain the *Christoffel word* $u_\alpha = a_1 c_\alpha$;
2. by moving from O to infinity on the half-line $\mathcal{D}$, we encode by a_1 a crossing point (black point) with a vertical line and by a_2 a crossing point with an horizontal line. This gives the infinite word c_α;
3. by looking at the centers (white points) of the unit squares crossed by $\mathcal{D}$. These centers are ordered, two consecutive centers correspond to joining squares, so that the vector joining these points is one of the vector of the canonical basis (e_1, e_2). We encode by a_i the vector e_i, and we obtain the infinite word c_α.

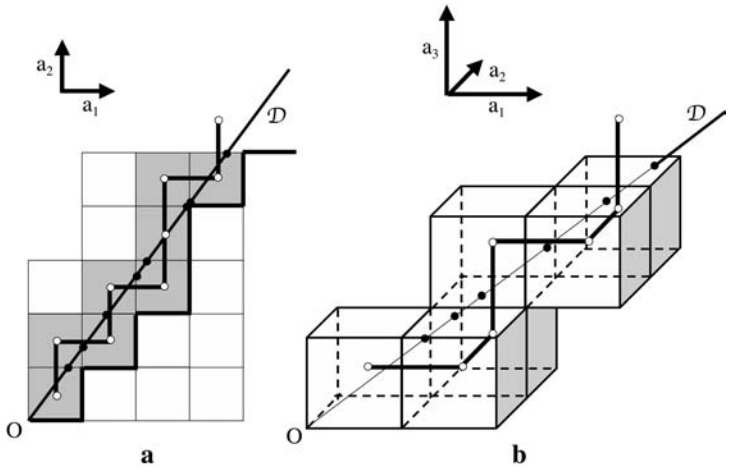


Fig. 1.

This works as soon as the half-line $\mathcal{D}$ has no integer point except for the origin, i.e., $\dfrac{\alpha_1}{\alpha_2}$ is an irrational number.

The same construction can be made using any half-lines $\mathcal{D}$, this gives general Sturmian words. These words have been intensively studied, see e.g. [1], [5],

[17], and are related to continued fraction expansions, Farey sequences, and the Stern-Brocot tree, [13] or [6]. The Christoffel words first appear in [7].

1.2.2 Billiard Words in Dimension $k \geq 3$

We consider some kinds of points in the k-dimensional space.

Definition 1.1. *Let $M = (x_1, x_2, \ldots, x_k) \in \mathbb{R}_+^k$ be a k-dimensional point with positive coordinates. Such a point is called:*

- *a 2 -integer point when at least two coordinates x_j are positive integers;*
- *a visible point whenever there are no 2-integer points N on the segment OM, except for the endpoints O and M;*
- *an integer point when all its coordinates are positive integers;*
- *a visible integer point when it is both visible and integer.*

We consider the *facets* of the unit k-cubes: a facet is a subset of the k-cube formed by all points having a fixed i-th coordinate. The methods 2. and 3. above can be generalized in any dimension. This works as soon as $\mathcal{D}$ crosses each facet in its interior, i.e., the half-line $\mathcal{D}$ has no 2-integer point except for O. This property corresponds to

$$\frac{\alpha_i}{\alpha_j} \notin \mathbb{Q}, 1 \leq i < j \leq k. \tag{1}$$

This condition already holds if we have:

$$\text{the } \alpha_i\text{'s are } \mathbb{Q}\text{-linearly independent} \tag{2}$$

and we say that $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is *totally irrational*, see [2]. These two conditions are the same only in dimension 2.

1.2.3 Finite Billiard Words

Let $M := (m_1, m_2, \ldots, m_k) \in \mathbb{N}^k$, where the m_i are pairwise coprime. The segment OM crosses several k-cubes and one defines, as before, a finite word c_M on the same alphabet, called the *(finite) billiard word* associated to M. One has:

$$\begin{cases} |c_M|_{a_i} = m_i - 1, 1 \leq i \leq k \\ |c_M| = \sum_{i=1}^{k} m_i - k \end{cases}$$

Note that, as usual, $|v|$ is the length of word v, and $|v|_a$ its a-degree. Observe that c_M is a palindrome.

1.2.4 Billiard Words with Intercept

The same construction can be made for any half-line $\mathcal{D}$ parallel to $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and starting from any point S as soon as $\mathcal{D}$ does not contain any 2-integer point. This condition is assumed in the following. By method 2. or 3., we construct the *billiard word with intercept* denoted by $c_{\alpha,S}$. So we have $c_\alpha = c_{\alpha,O}$, and by translation it suffices to consider the starting points S on the facets of the unit k-cube at the origin, i.e., with at least one zero coordinate s_j and the other ones between 0 and 1.

2 Factors of Billiard Words

2.1 Some Notations

For a given vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, we consider the subspace $\mathcal{D}^\perp$ of the k-dimensional space, and the orthogonal projection $\mathcal{P}$ of the open unit k-cube centered at the origin onto $\mathcal{D}^\perp$. We denote by b_j, $1 \leq j \leq k$, the orthogonal projection onto $\mathcal{D}^\perp$ of the vectors e_j of the canonical basis of $\mathbb{R}^k$. C denotes the orthogonal projection of the center of the first k-unit cube in the grid, i.e., $\vec{OC} = \frac{1}{2}\sum_{j=1}^{k} b_j$, and by $\tilde{S}$ the crossing point of $\mathcal{D}$ and $\mathcal{D}^\perp$.

Definition 2.1. *A finite sequence $H_0, H_1, H_2, \ldots, H_n$ of points in $\mathcal{P}$ is called a b-trajectory if for each $1 \leq i \leq n$, there exists some $j = j(i)$ such that $\vec{H_{i-1}H_i} = b_j$.*

2.2 b-Trajectories and Factors of Billiard Words

Such a b-trajectory can be characterized by its origin H_0 and its coding word $a_{j(1)}a_{j(2)}\ldots a_{j(n)}$ in $\mathcal{A}$. Let $\mathcal{F}_{\alpha_1,\alpha_2,\ldots,\alpha_k}$ be the language, i.e., the set of factors, of all the billiard words $c_{\alpha_1,\alpha_2,\ldots,\alpha_k,S}$.

Theorem 2.1. *1. For a given length n and almost all points H_0 in $\mathcal{P}$, there exists a unique b-trajectory of length n and starting from H_0.*
2. A finite word on $\mathcal{A}$ is in $\mathcal{F}_{\alpha_1,\alpha_2,\ldots,\alpha_k}$ if and only if it encodes some b-trajectory in $\mathcal{P}$.
3. In the totally irrational case (2) and for any S, a finite word on $\mathcal{A}$ is a factor of $c_{\alpha_1,\alpha_2,\ldots,\alpha_k,S}$ if and only if it encodes some b-trajectory in $\mathcal{H}$.

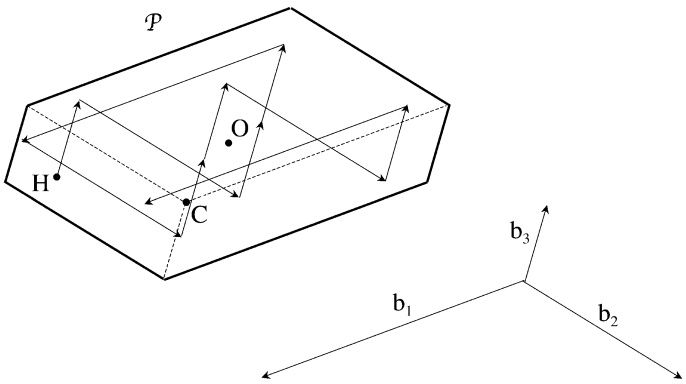


Fig. 2.

In the Figure 2 above, in dimension 3, $\mathcal{P}$ is an hexagon, and the b-trajectory of length 11 starting from H is represented.

The exceptional points in the first part of the theorem above can be characterized. They correspond to the ambiguous cases, i.e., to half-lines $\mathcal{D}$ parallel to

$(\alpha_1, \alpha_2, \ldots, \alpha_k)$ which contain some 2-integer point. By iterating the Part 1. of the theorem, there exists a unique infinite b-trajectory starting from a given non-exceptional point in $\mathcal{P}$. It can be easily proved that the orthogonal projections of the centers of the unit k-cubes crossed by $\mathcal{D}$ remain in the translated polyedron $\mathcal{P}_S := \mathcal{P} + \sum_{j=1}^{k} s_j b_j$ whose center is $\tilde{S}$. Thus the billiard word $c_{\alpha,S}$ encodes the infinite b-trajectory starting from the point $C - \tilde{S}$, i.e., $\sum_{j=1}^{k} (\frac{1}{2} - s_j) b_j$.

Moreover, in the totally irrational case (2), the set of the orthogonal projections of the centers of the unit k-cubes crossed by $\mathcal{D}$ is dense in $\mathcal{P}_S$. This proves the Part 3. of the theorem.

It is possible to give a characterization of those points H_0 which are starting points of a b-trajectory, for a given finite word v on $\mathcal{A}$.

Proposition 1 *For any finite word v on the alphabet $\mathcal{A}$, the set $\mathcal{P}_v$ of the starting points of the b-trajectories with coding word v is an open polyedral convex set of dimension $k-1$. The diameter of this set tends to zero as the length of v tends to infinity.*

For many v the set $\mathcal{P}_v$ is an empty set, the non-empty sets correspond to those words v of the language $\mathcal{F}_\alpha$. In [2] and [3], the study of the structure of these sets allows to obtain the complexity of the language of the billiard words in the totally irrational case, for 3-dimensional words and in any dimension respectively.

2.3 Application to Palindromic Factors

We say that a factor w of a finite word v is a *central factor* when we have $v = v_1 w v_2$ with $|v_1| = |v_2|$, and we call *center of* v the central factor of v of length 1 or 2, as $|v|$ is odd or even.

Proposition 2
- *For each letter a_j in $\mathcal{A}$, the set $\mathcal{P}_{a_j}$ contains the point $-\frac{1}{2} b_j$, thus it is non-empty.*
- *The only non-empty set $\mathcal{P}_{a_j a_j}$ corresponds to the letter a_{j_0} coding the unique b-trajectory of length 1 starting at the origin.*

Let $\mathcal{H}$ be the closure of the set of points in the infinite b-trajectory corresponding to the billiard word $c_{\alpha,S}$. With the total irrationality hypothesis (2), $\mathcal{H}$ is the closure of $\mathcal{P}$, and we obtain that any billiard word has infinitely many palindromic factors, and more precisely:

- for any even integer n, a unique palindromic factor of length n, whose center is the only pair of letters $a_{j_0} a_{j_0}$ which belongs to the language $\mathcal{F}_{\alpha_1, \alpha_2, \ldots, \alpha_k}$;
- for any odd integer n, and for each letter a_j in the alphabet $\mathcal{A}$, a unique palindromic factor of length n in which the letter a_j is in central position.

The general situation (1) is more complicate.

Theorem 2.2. – *The billiard word $c_{\alpha,S}$ contains arbitrary long palindromic factors of even length if and only if O is in $\mathcal{H}$.*
– *The billiard word $c_{\alpha,S}$ contains arbitrary long palindromic factors of odd length and of center a_j if and only if $\frac{1}{2}b_j$ is in $\mathcal{H}$.*

As an example in dimension $k = 3$, we take for α the vector $(2, \sqrt{5}, 1 + \sqrt{5})$ and consider the billiard word c_α starting at the origin. Then $\frac{1}{2}b_1$ is in $\mathcal{H}$, but O, $\frac{1}{2}b_2$ and $\frac{1}{2}b_3$ are not in $\mathcal{H}$. Moreover the billiard word does not contain any palindromic factor of even length: in this case, $j_0 = 3$, but $\mathcal{H}$ does not intersect $\mathcal{P}_{a_3a_3}$. So the word a_3a_3 is not a factor of this billiard word, the number of factors of length 2 is equal to 6 instead of 7 in the general case.

We prove that there always exist arbitrary long palindromic factors in any billiard word in dimension 3. This result is false in higher dimension, and it is possible to find a billiard word with 4 letters, with a finite number of palindromic factors.

3 Palindromic Prefix Factors of Billiard Words

In this section we consider only billiard words c_α starting from the origin.

3.1 Dimension 2

The question of palindromic prefix factors of billiard words has been studied in dimension 2, see for example [9], [10], [11], [12] and [16].

It can be easily shown that the palindromic prefix factors of billiard words are finite billiard words, and correspond to the continued fraction expansion (see [8] or [14]) of the slope $\rho := \frac{\alpha_2}{\alpha_1}$ of the half-line $\mathcal{D}$.

Theorem 3.1. *The palindromic prefixes of the infinite billiard word c_α are finite billiard words; for all $n > 0$ they are the prefixes of length $p_n + q_n - 2$, for all the main and intermediate convergents $\frac{p_n}{q_n}$ of the continued fraction expansion of the real number ρ.*

This result is stated in [4], [9], [10], in a slightly different formulation. A purely geometrical proof of this result can also be given, by using a geometrical approach of the theory of continued fraction expansions, made by H.J.S. Smith and Felix Klein, [15], at the end of the XIXth century, see for example [8].

3.2 Higher Dimensions

3.2.1 Main Result

Now the dimension k is greater than 2. In the general case, the billiards words have only finitely many palindrome prefix factors. Note that these palindromic prefixes are finite billiard words as in dimension 2.

Theorem 3.2. *With Hypothesis (1), in dimension $k \geq 3$, the set of vectors $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ such that $c_{\alpha_1,\alpha_2,\ldots,\alpha_k}$ has infinitely many palindromic prefixes if a negligible set, with respect to the Lebesgue measure on the k-dimensional unit sphere. However, this set is dense on the positive part of this unit sphere.*

As an example in dimension 3, if we choose $(\alpha_1, \alpha_2, \alpha_3) = (1, \frac{15+\sqrt{5}}{10}, \frac{1+\sqrt{5}}{2})$, the corresponding billiard word $c_{\alpha_1,\alpha_2,\alpha_3}$ is: $a_2a_3a_1a_2a_3a_2a_3a_1a_2a_3a_2a_1$ $a_3a_2a_3a_1a_2a_3a_2a_3a_1a_2a_3a_2a_1a_3a_2a_3a_2a_1a_3a_2a_1a_3a_2a_3a_2a_1a_3a_2a_2a_3\ldots$ and a_2 is the only one palindromic prefix factor of $c_{\alpha_1,\alpha_2,\alpha_3}$.

3.2.2 Integer Prefix Points

We use the notion of integer prefix point: the integer point M is called an *integer prefix point* of $\mathcal{D}$ when the triangle OHM does not contain any 2-integer points, except for O and M, where H is the orthogonal projection of M onto the line $\mathcal{D}$. These points correspond to the palindromic prefix factors of the billiard words.

Proposition 3 *Let $M = (m_1, m_2, \ldots, m_k)$ be any integer point and $\mathcal{D}$ any half-line parallel to vector $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ satisfying the irrationality condition (1). The following properties are the same:*

- *M is an integer prefix point of $\mathcal{D}$;*
- *c_M is a palindromic prefix factor of $c_{\alpha_1,\alpha_2,\ldots,\alpha_k}$.*

3.2.3 Up and Down Method

We can use the *projections* to study these points. For $1 \leq i < j \leq k$, the projections π_{ij} are defined by:

- on finite and infinite words, by $u_{ij} := \pi_{ij}(u)$ is the word on $\mathcal{A}_{ij} := \{a_i, a_j\}$ obtained by erasing in u any letters excepted a_i and a_j;
- on $I\!R^k_+$, by $\pi_{ij}(x_1, x_2, \ldots, x_k) := (x_i, x_j)$, which belongs to the plane Π_{ij};
- on the set of vectors $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ in $I\!R^k_+$, by $\pi_{ij}(\alpha_1, \alpha_2, \ldots, \alpha_k) := (\alpha_i, \alpha_j)$.

It is very simple to prove that M is an integer prefix point of $\mathcal{D}$ if and only if all its projections $M_{ij} := \pi_{ij}(M)$ are integer prefix points of $\mathcal{D}_{ij} := \pi_{ij}(\mathcal{D})$, thus correspond to some convergent of the ratio $\frac{\alpha_j}{\alpha_i}$. Hence the first coordinate m_1 of M is the denominator of some convergent for any irrational numbers $\frac{\alpha_j}{\alpha_1}$, $2 \leq j \leq k$. This is a *synchronization* property of convergents.

A classical theorem due to Lagrange, see [14], implies that M is an integer prefix point of $\mathcal{D}$ when M is very close to $\mathcal{D}$. Using this property, we can prove in some special cases that M is an integer prefix point of $\mathcal{D}$ when all the M_{1j}, $2 \leq j \leq k$, are integer prefix points of the half-lines $\mathcal{D}_{1j}$.

3.2.4 A Probabilistic Result on Continued Fraction Expansions

The following result shows that the synchronization property of convergents cannot appear infinitely many, for almost all vectors $(\alpha_1, \alpha_2, \ldots, \alpha_k)$.

Proposition 3.1. *For almost all positive real numbers α, the set of positive real numbers β, having an infinity of denominators of intermediate or main convergents in common with α, has Lebesgue measure* 0.

This proposition can be proved by using some classical probabilistic results on continued fraction expansion, see [18]. Then we get an upper bound of the probability that a given integer q is the denominator of some convergent, see the proposition below, and use the well-known Borel-Cantelli lemma.

Proposition 3.2. *Let q be a positive integer ≥ 2, and $0 < x < 1$. Then the probability P_q that q be a denominator of a main or intermediate convergent of x satisfies:*

$$P_q \leq \frac{2}{\sqrt{q}} + \frac{2}{\sqrt{q}-1}$$

3.2.5 The Exceptional Case

There are two different proofs for the existence of billiard words with infinitely many palindromic prefix factors. The first one consists of an iterative construction of the numerators and denominators of the convergents of the ratios $\frac{\alpha_j}{\alpha_1}$, $2 \leq j \leq k$, such that we can use the Lagrange theorem. The second one is a purely geometrical proof.

Both proofs give the density of these words, and the first one allows to choose billiard words respecting the total irrationality property (2).

References

1. J.-P. Allouche, J. Shallit, *Automatic sequences: Theory and Applications*, Cambridge University Press, Cambridge, 2003.
2. P. Arnoux, C. Mauduit, I. Shiokawa, J. I. Tamura, Complexity of sequences defined by billiard in the cube, *Bull. Soc. Math. France*, **122**, 1994, 1-12.
3. Y. Baryshnikov, Complexity of trajectories in rectangular billiards, *Comm. Math. Phys.*, **174**, 1995, 43-56.
4. J. Berstel, A. de Luca, Sturmian words, Lyndon words and trees, *Theoret. Comput. Sci.*, **178**, 1997, 171-203.
5. J. Berstel, P. Séébold, Sturmian words, in M. Lothaire, *Algebraic combinatorics on words*, Cambridge University Press, 2002.
6. J.-P. Borel, F. Laubie, Quelques mots sur la droite projective réelle, *J. Théorie des Nombres de Bordeaux*, **5**, 1993, 23-51.
7. E. B. Christoffel, Observatio arithmetica, *Annali di Matematica*, **6**, 1875, 148-152.
8. H. Davenport, *The Higher Arithmetic: An Introduction to the Theory of Numbers, 7th ed.*, Cambridge Univ. Press, 1999.
9. A. de Luca, Sturmian words: structure, combinatorics, and their arithmetics, *Theoret. Comput. Sci.*, **183**, 1997, 45-82.
10. A. de Luca, Combinatorics of standard sturmian words, *Structures in Logic and Computer Science, Lecture Notes Comput. Sci.*, **1261**, 1997, 249-267.
11. X. Droubay, Palindromes in the Fibonacci word, *Inf. Proc. Letters*, **55**, 1995, 217-221.

12. X. Droubay, G. Pirillo, Palindromes and Sturmian words, *Theoret. Comput. Sci.*, **223**, 1999, 73-85.
13. R.L. Graham, D.E. Knuth, O. Patashnik, *Concrete Mathematics*, Addison-Wesley, Reading, MA, 2nd ed., 1994.
14. G.H. Hardy, E.M. Wright, *An Introduction to the Theory of Numbers*, Oxford at the Clarendon Press, 5th ed., 1979.
15. F. Klein, *Ausgewählte Kapitel der Zahlentheorie*, Teubner, Liepzig, 1907.
16. G. Pirillo, A new characteristic property of the palindrome prefixes of a standard Sturmian word, *Sém. Lotharingien Combinatoire*, **43**, 1999, (electronic, see http://www.mat.univie.ac.at/∼slc/).
17. N. Pytheas Fogg, *Substitutions in Dynamics, Arithmetics and Combinatorics*, Lecture Notes in Math. **1794**, V. Berthé and al. ed., 2002.
18. A. M. Rockett, P. Szüsz, *Continued Fractions*, World Scientific Pub. Co, 1992.

A Note on a Result of Daurat and Nivat*

Srečko Brlek, Gilbert Labelle, and Annie Lacasse

Laboratoire de Combinatoire et d'Informatique Mathématique,
Université du Québec à Montréal,
CP 8888, Succ. Centre-ville, Montréal (QC) Canada H3C3P8
{brlek,gilbert,lacasse}@lacim.uqam.ca

Abstract. We consider paths in the square lattice and use a valuation called the *winding number* in order to exhibit some combinatorial properties on these paths. As a corollary, we obtain a characteristic property of self-avoiding closed paths, generalizing in this way a recent result of Daurat and Nivat (2003) on the boundary properties of polyominoes concerning salient and reentrant points.

Keywords: Lattice paths, discrete regions, winding number, polyominoes, salient and reentrant points

1 Introduction

Finding the route from a point A to a point B in a city is a problem easily solved, when an accurate local description of the geometric configuration of intersections is provided: indeed, everybody experienced the sometimes difficult task of getting to a point by following a list of direction changes at crossroads. This might be called the *taxi driver algorithm*. It is also well known that any differentiable continuous curve in the plane may be linearly approximated by a piecewise linear curve, which is used to solve all the basic computation needs in calculus. Here, we adopt this parametric or "dynamical" point of view, which consists to consider curves as a sequence of elementary steps, its local description, instead of using a closed formula or equation, which may be considered as a global description. Doing so we pursue the study initiated in a previous paper [2] where we explicited the benefits of applying a discrete version of the Green's theorem on discrete paths. In this paper we consider curves on square grids or lattices, identified with the discrete plane $\mathbb{Z} \times \mathbb{Z}$. A path in the square lattice is a polygonal path made of the elementary unit translations

$$a = (1,0), \overline{a} = (-1,0), b = (0,1), \overline{b} = (0,-1).$$

A finite path w is therefore a word on the alphabet $\Sigma = \{a, \overline{a}, b, \overline{b}\}$, also known as the Freeman chain code [5, 6] (see [1] for further reading). For instance the paths in Figure 1 are coded respectively by the words

$U = abbaaa\overline{b}\overline{a}bba\overline{b}\overline{b}\overline{a}\overline{a}\overline{a}bbbbaab\overline{b}\overline{a}b\overline{a}\overline{a}\overline{b}\overline{b}aaaaaba\overline{b}a\overline{b}\overline{a}\overline{b}\overline{a}\overline{b}\overline{a}bbaaab\overline{b}\overline{a}\overline{a}ba$,

$V = baab\overline{a}b\overline{a}\overline{b}abbabaaba\overline{b}a\overline{b}aaa\overline{b}a\overline{b}\overline{a}\overline{b}\overline{a}\overline{a}bbbba\overline{b}\overline{b}\overline{b}\overline{b}\overline{b}\overline{a}\overline{b}a\overline{b}\overline{a}\overline{b}a\overline{b}\overline{a}\overline{b}bbb\overline{a}\overline{b}aabb\overline{a}\overline{a}\overline{a}\overline{b}\overline{b}\overline{b}\overline{a}\overline{a}\overline{a}\overline{a}$.

* With the support of NSERC (Canada)

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 189–198, 2005.
© Springer-Verlag Berlin Heidelberg 2005

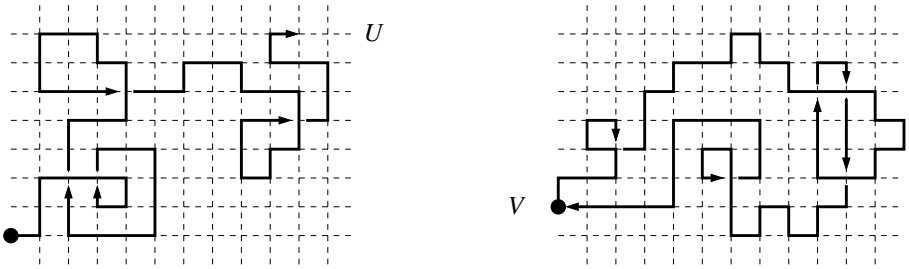


Fig. 1. An open path U and a closed path V with origin $\bullet$

Observe that crossings are quite ambiguous in the geometric representation. Fortunately, the codings provided by the words representation are not.

The paper is organized as follows. We introduce in Section 2 the *winding number* of a path in a square lattice, a valuation derived from the *Manhattan taxi driver algorithm*, and list some of its basic properties. Then we apply the results to self-avoiding paths. In the case of closed self-avoiding paths we obtain a generalization of the recent and nice result in discrete geometry obtained by Daurat and Nivat [4] that relates the S salient and R reentrant points of a polyomino:

$$S - R = 4.$$

Finally we discuss the extensions to other lattices: amazingly there is an analogue for hexagonal lattices stating that

$$S - R = 6,$$

and there is no other regular or semi-regular lattices for which the formula $S - R = \text{const}$ holds in general.

2 A Valuation on Square Lattice Paths

From now on, a path w of length $|w| = n$ is a function $w : [1..n] \longrightarrow \Sigma$ and is written $w = w_1 w_2 \cdots w_n$ where w_i is the i-th letter, $1 \leq i \leq n$. Also, the number of occurences of a given letter α in the word w is denoted $|w|_\alpha$. The set of n-length paths is denoted Σ^n, the set of all paths is Σ^*, and for later use $\Sigma^{\geq 2}$ is the set of paths of length at least 2. The *mirror image* $\widetilde{u}$ of $u = u_1 u_2 \cdots u_n \in \Sigma^n$ is the word $\widetilde{u} = u_n u_{n-1} \cdots u_1$. In what follows the alphabet is $\Sigma = \{a, \overline{a}, b, \overline{b}\}$. There is a usual length preserving morphism, the *complement* defined by the relations

$$\overline{\overline{a}} = a \ ; \ \overline{\overline{b}} = b,$$

which extends to words as follows. The complement of $u = u_1 u_2 \cdots u_n \in \Sigma^n$, is the word $\overline{u} = \overline{u_1}\ \overline{u_2}\ \overline{u_3} \cdots \overline{u_n}$.

2.1 The Manhattan Taxi Driver Algorithm

On a square grid, a path can be described by a sequence of left or right turns along with forward and backward steps. Using the alphabet Σ of units steps, we define the corresponding set of movements by

$$\begin{array}{ll} V_L = \{ab, b\overline{a}, \overline{a}\overline{b}, \overline{b}a\} & \text{is the set of left turns;} \\ V_R = \{ba, a\overline{b}, \overline{b}\overline{a}, \overline{a}b\} & \text{is the set of right turns;} \\ V_F = \{aa, \overline{a}\overline{a}, bb, \overline{b}\overline{b}\} & \text{is the set of forward steps;} \\ V_B = \{a\overline{a}, \overline{a}a, b\overline{b}, \overline{b}b\} & \text{is the set of backward steps.} \end{array}$$

These sets respectively correspond to the basic movements in the left (L), right (R), forward(F) and back (B) directions. Observe that

$$V_L = \widetilde{V_R}\ ;\ V_R = \overline{V_R}\ ;\ V_L = \overline{V_L}\ ;\ V_B = \overline{V_B}\ ;\ V_B = \widetilde{V_B}\ ;\ V_F = \widetilde{V_F}\ ;\ V_F = \overline{V_F}.$$

Note also that each path $w = w_1 w_2 \cdots w_n$ is completely determined, up to translation, by its initial step and a word on the alphabet $\Sigma_d = \{L, F, R, B\}$. Indeed let $g : \Sigma^2 \longrightarrow \Sigma_d$, be defined by

$$g(u) = \begin{cases} L & \text{if } u \in V_L, \\ F & \text{if } u \in V_F, \\ R & \text{if } u \in V_R, \\ B & \text{if } u \in V_B. \end{cases} \tag{1}$$

Then g is extended to a function $f : \Sigma^{\geq 2} \longrightarrow \Sigma \times \Sigma_d^*$ defined on arbitrary paths by

$$f(w) = \left(w_1, \prod_{i=1}^{n-1} g(w_i w_{i+1}) \right), \tag{2}$$

where n is the length of the word w and the product is the concatenation. Since we are only interested in the geometric properties of paths, we drop the starting step (which amounts to work with the equivalence class determined by the rotations). The following lemma is straightforward.

Lemma 1 *Let $w \in \Sigma^*$ be a closed path, then*

(i) *w is of even length: $w = w_1 w_2 \cdots w_{2n}$ for some integer $n \geq 1$;*
(ii) *$f(w)$ is of odd length.*

Proof. (i) Since w is closed, we have $|w|_a = |w|_{\overline{a}}$ and $|w|_b = |w|_{\overline{b}}$. And it follows that

$$|w| = |w|_a + |w|_{\overline{a}} + |w|_b + |w|_{\overline{b}} = 2|w|_a + 2|w|_b = 2(|w|_a + |w|_b).$$

(ii) clearly follows from (i), since $|f(w)| = |w| - 1$. □

We introduce now a weight function $p : \Sigma^2 \longrightarrow \Sigma_p = \{-1, 0, 1, 2\}$ on paths of length two by defining

$$p(u) = \begin{cases} -1 & \text{if } u \in V_R, \\ 0 & \text{if } u \in V_F, \\ 1 & \text{if } u \in V_L, \\ 2 & \text{if } u \in V_B. \end{cases} \tag{3}$$

This weight extends to a valuation $P : \Sigma^{\geq 1} \longrightarrow \mathbb{Z}$ defined on each path w such that $n \geq 1$ by setting

$$P(w) = \begin{cases} \sum_{i=1}^{n-1} p(w_i w_{i+1}) & \text{if } n > 1, \\ 0 & \text{if } n = 1. \end{cases} \tag{4}$$

The valuation is nothing but the *winding number*, that is to say the number of direction changes by $\frac{\pi}{2}$ units: a left turn corresponds to a positive rotation by $(+1)\frac{\pi}{2}$, a right turn to a negative rotation by $(-1)\frac{\pi}{2}$ and an element in V_B to a rotation by $(2)\frac{\pi}{2} = \pi$ (stepping in the opposite direction).

The following additivity formula follows directly from the definition of the valuation function P given by (4).

Property 1 [Additivity] *Let $u = u_1 u_2 \cdots u_k u_{k+1} \cdots u_n$ be a word in $\Sigma^{\geq 2}$. Then we have*

$$P(u) = P(u_1 \cdots u_k) + P(u_k u_{k+1}) + P(u_{k+1} \cdots u_n).$$

Examples. Consider the path U in Figure 1. We have $P(U) = -8$. Observe that the first and last letter are equal and that there are three clockwise cycles and 1 counterclockwise cycle. So that U can be factorized as

$$U = abba{\cdot}(aa\overline{b}\overline{a}bbaa\overline{bbb}\overline{aaa}bb){\cdot}bbaab{\cdot}(b\overline{a}b\overline{aa}\overline{bb}aaa){\cdot}aabaa\overline{b}aa\overline{b}{\cdot}(\overline{b}\overline{a}\overline{b}\overline{a}bbaa){\cdot}abb\overline{aa}ba.$$

In other words the final direction is the same but we have made 4 complete rotations: $-3 \times 4 + 1 \times 4 = -8$.

For the path V we have $P(V) = -3$. There are two clockwise and one counterclockwise cycles which cancel. The first letter of V is $v_1 = b$ and the last letter is $\overline{a}$.

More precisely, the valuation modulo 4 depends only on the first and last letter of the path. These observations lead to the following result.

Theorem 1. *Let $w = w_1 w_2 \cdots w_n \in \Sigma^{\geq 2}$, then*

$$P(w) \equiv p(w_1 w_n) \bmod 4.$$

Proof. The valuation modulo 4 indicates the relative position between the initial step and the current step. More precisely, we proceed by induction as follows: the result is straightforwardly true if $n \leq 3$, since

$$P(w_1 w_2 w_3) \equiv p(w_1 w_3) \bmod 4.$$

Suppose that it is true for $n-1 \geq 3$, then by definition,

$$P(w_1 w_2 \cdots w_n) = P(w_1 w_2 \cdots w_{n-1}) + P(w_{n-1} w_n).$$

Hence, by the induction hypothesis,

$$\begin{aligned} P(w_1 w_2 \cdots w_n) &\equiv p(w_1 w_{n-1}) + p(w_{n-1} w_n) \bmod 4 \\ &= P(w_1 w_{n-1} w_n) \\ &\equiv p(w_1 w_n) \bmod 4. \quad \square \end{aligned}$$

Since the winding number depends on the relative position of the first and last letter, the following properties follow immediately.

Proposition 1 *For any path $w = w_1 \cdots w_n$ we have*

(i) $P(w) \equiv 0 \bmod 4 \iff w_1 = w_n$;
(ii) $P(w \cdot w_1) \equiv P(\widetilde{w \cdot w_1}) \equiv 0 \bmod 4$.

Note that an analogous result is true for any piecewise differentiable path: this is nothing but the angle between the tangent vectors at each end point.

2.2 An Application to Self-avoiding Paths

By *self-avoiding path* we mean an oriented path having no backward steps and no distinct vertices v, v_1, v_2, v_3, v_4 such that $v_1 v v_2$ and $v_3 v v_4$ are triples made of consecutive vertices of the path (traversed in either direction, see Figure 2). In other words a path may not traverse itself.

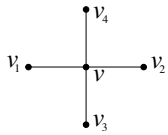


Fig. 2. The forbidden multiple points

Nevertheless multiple points may exist and some of them are displayed in Figure 3. Note that a self-avoiding closed path may not contain multiple points of type (a) or (d). Indeed, if a closed path w contains a multiple point of say type (a), then the point v_1 must be connected to the point v_2 by some self-avoiding path x, and point v_3 to point v_4 by some self-avoiding path y. Then x and y intersect, a contradiction.

In the case of self-avoiding paths, backward steps in $V_B = \{a\overline{a}, \overline{a}a, b\overline{b}, \overline{b}b\}$ are omitted and the valuation is the restriction of p to the set $\Sigma^2 \setminus V_B$, namely the function (identified by the same letter):

$$p : \Sigma^2 \setminus V_B \longrightarrow \{-1, 0, 1\} \quad \text{by} \quad p(u) = \begin{cases} -1 & \text{if } u \in V_R, \\ 0 & \text{if } u \in V_F, \\ 1 & \text{if } u \in V_L. \end{cases}$$

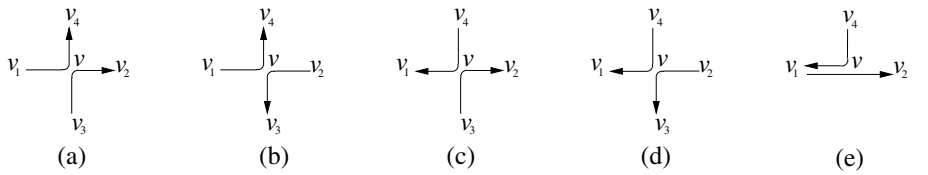


Fig. 3. Some allowed multiple points types

In this case the winding number is computed only with left and right turns and Proposition 1 may be specialized as follows.

Lemma 2 *Every nonempty open self-avoiding path* $w = w_1 w_2 \cdots w_n$ *such that*

(i) $w_1 = w_n$,
(ii) *the directed half-lines defined by* w_1 *and* w_n *do not intersect the path,*

satisfies

$$P(w) = P(\widetilde{w}) = 0. \tag{5}$$

Although a more general formulation holds, the condition (i) is enough for our need and (ii) is necessary to avoid the situations described in Figure 4.

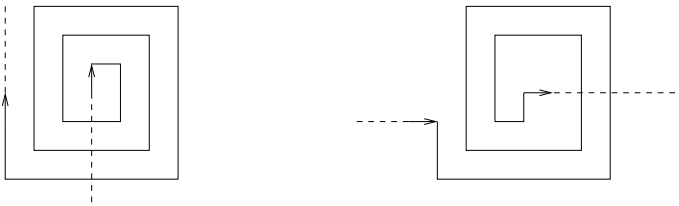


Fig. 4. Half-line intersecting the path

Proof. Condition (i) and the definition of P give

$$\begin{aligned} P(w) &= \sum_{i=1}^{n-1} p(w_i w_{i+1}) \\ &= \sum_{w_i w_{i+1} \in V_L} p(w_i w_{i+1}) + \sum_{w_i w_{i+1} \in V_F} p(w_i w_{i+1}) + \sum_{w_i w_{i+1} \in V_R} p(w_i w_{i+1}) \\ &= \sum_{w_i w_{i+1} \in V_L} p(w_i w_{i+1}) + \sum_{w_i w_{i+1} \in V_R} p(w_i w_{i+1}) \\ &\equiv 0 \bmod 4. \end{aligned} \tag{6}$$

By condition (ii), cummulative complete rotations are forbidden and the last sum is actually $P(w) = 0$. □

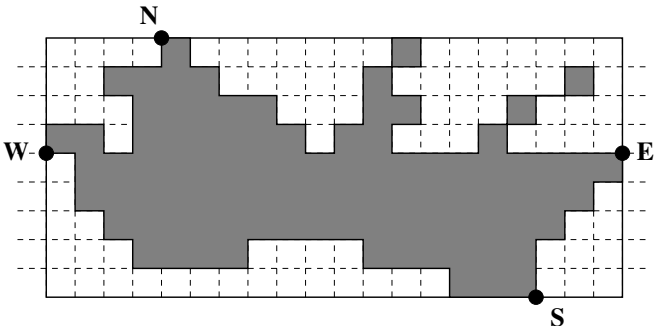


Fig. 5. A region and its four extremal points

Now, every self-avoiding closed path U is the boundary of some unique region. Let Q be the smallest rectangle containing U as shown in Figure 5.

The four *extremal* points are defined by the coordinates: **W** is the lowest intersection with the left side of Q, **N** the leftmost intersection with the top side, **E** the highest intersection with the right side, and **S** the rightmost intersection with the bottom side. Accordingly U may be rewritten (counterclockwise), starting from **W** as

$$U = (aX) \cdot (bY) \cdot (\overline{a}Z) \cdot (\overline{b}V), \tag{7}$$

where X, Y, Z, V may be either empty (not necessarily simultaneously), or $X = xa, Y = yb, Z = z\overline{a}, V = v\overline{b}$. Emptyness corresponds to the case where the extremal points coincide with the corners of the rectangle.

Consider the case where X, Y, Z, and V are non-empty. Then, using the additivity Property 1 and the Lemma 2 above we have

$$\begin{aligned} P(U) &= P(axa) + P(ab) + P(byb) + P(b\overline{a}) + P(\overline{a}z\overline{a}) + P(\overline{a}\overline{b}) + P(\overline{b}v\overline{b}) \\ &= 0 + 1 + 0 + 1 + 0 + 1 + 0 = 3, \end{aligned}$$

or equivalently

$$|f(U)|_L = |f(U)|_R + 3.$$

The case where some or all of the X, Y, Z, V are empty is left to the reader. We have thus proved the following result.

Corollary 1 *Let U be the contour of a polyomino such that $f(U)$ consists only of right and left turns, i.e, $f(U) \in \{LR\}^*$. Then we have*

$$|f(U)| = 2|f(U)|_L - 3 = 2|f(U)|_R + 3.$$

Proposition 2 *Every self-avoiding closed path U satisfies $P(U) = 3$.*

Taxi driver Proof. Making a run from W to W counterclockwise takes 3 left turns more than right ones. Indeed, each of the four open paths, WS, SE, EN and NW in the factorization of U above contains exactly the same number of right and left turns. □

A polyomino is a finite union of unit squares, defined up to a translation and side-connected (see Figure 6). Left (resp. right) turns in the counterclockwise traversal of the contour correspond to salient (resp. reentrant) points in the terminology of Daurat and Nivat [4]. The four extremal points in the decomposition above are salient.

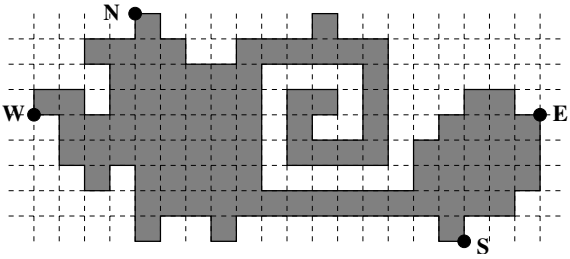


Fig. 6. A simply connected polyomino and its four extremal points

Since the closed path defining a polyomino is self-avoiding we obtain a constructive simple proof of the following result of Daurat and Nivat [4].

Corollary 2 *The S salient and R reentrant points in every polyomino are related by the formula*

$$S - R = 4. \tag{8}$$

Proof. Let U be the self-avoiding contour written counterclockwise. Starting at W and using equation 7, we have $P(Ua) = P(U) + P(\overline{b}a) = 3 + 1 = 4$, or equivalently

$$|f(Ua)|_L = |f(Ua)|_R + 4. \qquad \square$$

Note that the four extra salient points may be canonically identified as W, S, E and N.

3 Other Regular Lattices

It is possible to extend the results of Section 2 to arbitrary paths and also to other classes of lattices by suitably defining the weight function on the set Σ^2 of elementary steps. Amongst the possible lattices we have the regular hexagonal lattice (see Figure 7), for which the Daurat-Nivat formula turns out to be

$$S - R = 6.$$

by using similar arguments. Indeed a path is coded on a 6-letter alphabet and left or right turns are $2\pi/6$ rotations coded by the corresponding set of 2-letter words.

In this case the additivity property still holds and Theorem 1 is adapted in a straighforward manner. The specialization to self avoiding paths follows: the

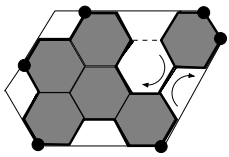


Fig. 7. Regular hexagonal lattice

rectangular bounding box of a polyomino (see Figure 5) is replaced by the smallest convex hexagon (see Figure 7) containing the closed path with six extremal points instead of four.

It is likely that there is no other regular lattice with such property since the formula $S - R =$ constant is false for closed self-avoiding paths in a triangular regular lattice. On the other hand, for the 8 semi regular lattices classified by Kepler (see Figure 8), that is the lattices made of at least two distincts regular polygons [7, 8] having the same cyclic pattern of polygon at each vertex, it can be checked that the Daurat-Nivat formula $S - R =$ constant is false. Due to the lack of space, the details will be carried out in a forthcoming paper.

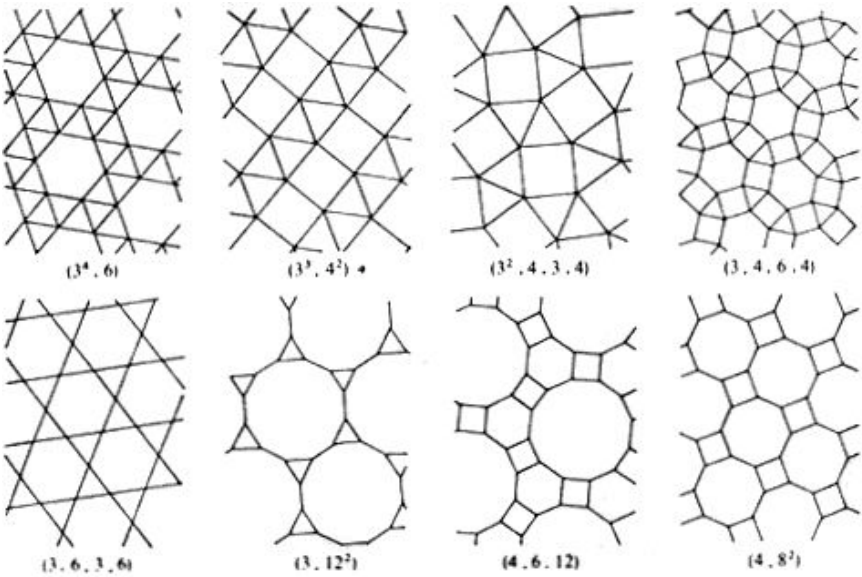


Fig. 8. The semi regular lattices [7]

4 Concluding Remarks

The valuation introduced in Section 2 is of geometrical nature. It measures the total variation of the angle between the first and final steps. In complex analysis the *contour winding number* or *index* of a contour γ relatively to a point z_0 is defined by

$$\text{Index}(\gamma, z_0) = \frac{1}{2\pi i} \oint_\gamma \frac{1}{z - z_0} dz,$$

and counts the number of times the contour γ turns around z_0. This close notion differs from ours, which is however defined even for open contours. Note that in the context of complex numbers our valuation P may be written as

$$P(w) = \frac{2 \sum_{k=1}^{n-1} \text{Arg}\left(\frac{w_{k+1}}{w_k}\right)}{\pi},$$

where $\text{Arg}(z)$ denotes the principal argument of the complex number $z \neq 0$ and satisfying $-\pi < \text{Arg}(z) \leq \pi$.

One may speculate about the existence of lattices where a Daurat-Nivat type formula holds, that is

$$mS - nR = l,$$

for some integers m, n, and l. For instance, one could consider, lattices defined by a finite set $\{v_1, v_2, \cdots, v_k\}$ of k linearly non zero vectors dependent over $\mathbb{N}$, in which case closed paths exist. A natural problem would be to find conditions on associated classes of generated simple closed paths that yield an analogue of the Daurat-Nivat formula.

Acknowledgements

Part of this work was done when the first author was visiting the LIRMM (Montpellier, France) as a CNRS research fellow in 2003.

References

1. Braquelaire, J.-P., Vialard, A.: Euclidean Paths: A New Representation of Boundary of Discrete Regions, Graphical Models and Image Processing 61(1) (1999) 16–43.
2. Brlek, S., Labelle, G., Lacasse, A.: Incremental Algorithms based on Discrete Green Theorem, in I. Nyström, G. Sanniti di Baja, S. Svensson (Eds), *Proc. 11-th International Conference on Discrete Geometry for Computer Imagery (DGCI'2003)* (Naples, Italy, November 19–21, 2003) Springer LNCS 2886 (2003) 277–287.
3. Clarke, A. L.: Isometrical polyominoes. J. Recreational Math. **13** (1980) 18–25.
4. Daurat, A., Nivat, M., Salient and Reentrant Points of Discrete Sets, *Proc. International Workshop on Combinatorial Image Analysis (IWCIA'03)* (Palermo, Italy, 14–16 May 2003).
5. Freeman, H. : On the Encoding of Arbitrary Geometric Configurations, IRE Trans. Electronic Computer **10** (1961) 260–268.
6. Freeman, H.: Boundary encoding and processing, in Picture Processing and Psychopictorics, B.S. Lipkin and A. Rosenfeld (Eds.) (Academic Press, New York, 1970) 241–266.
7. Grunbaum, B. and Shephard, G.C. : Tilings and Patterns. W. H. Freeman. New York, 1987.
8. Grunbaum, B. and Shephard, G.C : Tilings and Patterns: An Introduction. W. H. Freeman. New York, 1989.

Palindromes in Sturmian Words

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
I-80126 Napoli, Italy
aldo.deluca@unina.it, aledeluc@studenti.unina.it

Abstract. We study some structural and combinatorial properties of Sturmian palindromes, i.e., palindromic finite factors of Sturmian words. In particular, we give a formula which permits to compute in an exact way the number of Sturmian palindromes of any length. Moreover, an interesting characterization of Sturmian palindromes is obtained.

1 Introduction

Sturmian words have been widely studied for their theoretical importance and their applications to various fields of science. By definition, they are infinite words which are not eventually periodic and have minimal subword complexity. They also enjoy some remarkable characterizations of geometrical nature (*cutting sequences*, *mechanical words*). The reader is referred to [2] and to [1] for general surveys on this subject.

In recent years, some works have investigated Sturmian words by looking at their palindromic factors. A *palindrome* is a finite word which can be read indistinctively from left to right or from right to left.

Palindromes play an essential role in the structure of Sturmian words. In fact, an important theorem of X. Droubay and G. Pirillo [9] shows that an infinite word is Sturmian if and only if it has exactly one palindromic factor of length n for n even, and two for n odd. Moreover, A. de Luca and F. Mignosi [8] proved that the set of palindromic prefixes of all standard Sturmian words is equal to the set PER of *central words*, i.e., words having two periods p and q which are coprime, and length $p+q-2$. Central words are words over a two letter alphabet $\{a, b\}$ and satisfy remarkable structural properties. In particular, a central word w is such that wab and wba can be factorized as a product of two palindromes. Moreover, the set St of factors of all Sturmian words is equal to the set of factors of PER.

There exist, and even they are the majority, Sturmian palindromes which are not central. For instance, there are 14 Sturmian palindromes of length 7, whereas the number of central words of the same length is only 6 (see Table 2).

In this paper we are interested in the combinatorics and the structure of the language of Sturmian palindromes. A main result, proved in Section 4.1, is a simple formula which gives the number of Sturmian palindromes of any length.

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 199–208, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Moreover, some structural properties of Sturmian palindromes are proved in Section 4.2. A remarkable characterization of Sturmian palindromes is given. In particular, one obtains a new characterization of central words.

2 Preliminaries

Let $\mathcal{A}$ be a 2-letter alphabet $\{a, b\}$. As usual, we denote by $\mathcal{A}^*$ the *free monoid* generated by $\mathcal{A}$, that is the set of all words over $\mathcal{A}$ with the operation of concatenation. The identity element of $\mathcal{A}^*$ is the *empty word* ε. Let $w = a_1 a_2 \cdots a_n$ be a word, with $a_i \in \mathcal{A}$, $1 \leq i \leq n$. The integer n is called the *length* of w, and it is denoted by $|w|$. Conventionally, the length of the empty word is 0.

A word u is a *factor* of $w \in \mathcal{A}^*$ if $w = xuy$ for some words x, y. In the special case $x = \varepsilon$ (resp., $y = \varepsilon$), we call u a *prefix* (resp., *suffix*) of w. A factor u of w is *proper* if $u \neq w$. A factor u of w is *median* if $w = xuy$ with $|x| = |y|$. The set of factors of a word x is denoted by $\mathrm{Fact}\,(x)$. For any $X \subseteq \mathcal{A}^*$, one sets:

$$\mathrm{Fact}\,(X) = \bigcup_{x \in X} \mathrm{Fact}\,(x) \ .$$

Let $w = a_1 a_2 \cdots a_n$ be a word, with $a_i \in \mathcal{A}$, $1 \leq i \leq n$. A positive integer p is a *period* of w if for all i, j, $1 \leq i, j \leq n$, the following condition is satisfied: if $i \equiv j \pmod{p}$, then $a_i = a_j$. The minimal period of w will be denoted by π_w; it is natural to set $\pi_\varepsilon = 1$. For any $w \neq \varepsilon$, the unique positive integer k such that $w = z^k z'$, where $|z| = \pi_w$ and $|z'| < \pi_w$, will be called the *order* of w.

Given $w = a_1 a_2 \cdots a_n$ with all $a_i \in \mathcal{A}$, the *reversal* $w^\sim$ is the word $a_n \cdots a_1$. If $w = \varepsilon$, one sets $\varepsilon^\sim = \varepsilon$. A word $w \in \mathcal{A}^*$ is a *palindrome* if $w = w^\sim$. The set of all palindromes of $\mathcal{A}^*$ is denoted by PAL.

An *infinite word* x is just an infinite sequence of letters:

$$x = c_1 c_2 c_3 \cdots \text{ where } c_i \in \mathcal{A}, \text{ for all } i \geq 1 \ .$$

The product between a finite word and an infinite one is well defined. A (finite) factor of x is either the empty word or any sequence $u = c_i \cdots c_j$ with $i \leq j$, i.e., a finite block of consecutive letters in x. If $i = 1$, then u is a prefix of x. A *suffix* of x is an infinite word y such that $x = uy$ for some $u \in \mathcal{A}^*$. We shall denote by $\mathrm{Fact}\,(x)$ the set of finite factors of x.

An infinite word is *Sturmian* if for each $n \in \mathbb{N}$ it has $n+1$ factors of length n. An equivalent geometrical definition can be given in terms of *cutting sequences*. In fact, a Sturmian word can be defined by considering the sequence of cuts in a squared lattice $(\mathbb{N} \times \mathbb{N})$ made by a ray having a slope which is an irrational number α. A horizontal cut is denoted by the letter b, a vertical by a, and a cut with a corner by ab or ba. A Sturmian word represented by a ray starting from the origin is usually called *standard* or the *characteristic word* associated with the irrational α and it is often denoted by c_α.

Let x be a finite or infinite word over $\mathcal{A}$. A factor u of x is a *right special* factor of x if ua and ub are factors of x. As is well known [2], an infinite word x is Sturmian if and only if for any $n \geq 0$, there is only one right special factor of x of length n.

3 Sturmian Palindromes

Let St be the set of finite Sturmian words, i.e., factors of infinite Sturmian words over the alphabet $\mathcal{A} = \{a, b\}$. We recall that for any Sturmian word there exists a standard Sturmian word having the same set of factors (cf. [2]). In the sequel we shall be interested in the set $St \cap PAL$, whose elements will be called *Sturmian palindromes*.

As usual, we denote by PER the set of central words, that is, words having two coprime periods p and q and length $p + q - 2$. Conventionally, the empty word ε is central (in this case, $p = q = 1$). It is well known (see [2, 8]) that the set PER coincides with the set of palindromic prefixes of all standard Sturmian words, so that $PER \subseteq St \cap PAL$. However, the previous inclusion is strict since there exist non-central Sturmian palindromes, for instance *abba*. The set PER is particularly important because a finite word is Sturmian if and only if it is a factor of a central word, i.e., $St = \mathrm{Fact}\,(PER)$. We shall prove (Corollary 2) a similar characterization for Sturmian palindromes.

Theorem 1. *Every palindromic factor of a standard Sturmian word c_α is a median factor of a palindromic prefix of c_α.*

The result is attributed to A. de Luca [7] by J.-P. Borel and C. Reutenauer, who gave a geometrical proof in [3]. We shall see later a direct proof which does not use geometrical arguments.

Corollary 2. *A word is a Sturmian palindrome if and only if it is a median factor of some central word.*

Proof. Trivially, every median factor of a palindrome is itself a palindrome. Since $St = \mathrm{Fact}\,(PER)$, it follows that a median factor of an element of PER is a Sturmian palindrome.

Conversely, let u be in $St \cap PAL$. By definition, there exists an infinite (standard) Sturmian word s such that $u \in \mathrm{Fact}\,(s)$. By Theorem 1, u is a median factor of a palindromic prefix of s. Since palindromic prefixes of standard Sturmian words are exactly the elements of PER, the result follows. □

Our proof of Theorem 1, which follows a simple argument suggested by A. Carpi [4], is based on the following results (see [7]):

Proposition 3. *If $w \in \mathrm{Fact}\,(x)$, where x is an infinite Sturmian word, then the reversal $w^\sim$ is a factor of x too. Moreover, if x is standard, then w is a right special factor of x if and only if $w^\sim$ is a prefix of x.*

Corollary 4. *A palindromic factor of an infinite standard Sturmian word x is a right special factor of x if and only if it is a palindromic prefix of x.*

Proof (of Theorem 1). By contradiction, let $c_\alpha = \lambda u x$, where u is a palindrome that is not a median factor of any palindromic prefix of c_α, and $\lambda \in \mathcal{A}^*$ has minimal length for such condition. Since u cannot be a prefix of c_α, we have $|\lambda| \geq 1$. Thus we can assume, without loss of generality, $\lambda = \lambda' a$. Now let z be

the first letter of x, so that $x = zx'$. Suppose first $z = a$. The palindrome aua is not a median factor of a palindromic prefix of c_α, otherwise so would be u. But $c_\alpha = \lambda' auax'$ with $|\lambda'| < |\lambda|$, and this contradicts the minimality of $|\lambda|$. Therefore $z = b$, and then aub and $bua = (aub)^\sim$ are factors of c_α. This means in particular that u is a right special factor of c_α. Corollary 4 then implies that u is a prefix of c_α, a contradiction. □

4 Main Results

4.1 Enumeration of Sturmian Palindromes

In this section we shall give an explicit formula for the enumeration function of $St \cap PAL$. We start by recalling some basic facts (see [7, 8]):

Proposition 5. *Let w be a word. The following conditions are equivalent:*

1. *$w \in PER$,*
2. *awb and bwa are Sturmian,*
3. *awa, awb, bwa, and bwb are all Sturmian.*

Proposition 6. *If wa and wb are Sturmian words, then there exists a letter $x \in \mathcal{A}$ such that xwa and xwb are both Sturmian.*

We now prove two easy consequences (see also [7]):

Proposition 7. *Let $w \in \mathcal{A}^*$ be a palindrome. If wa and wb are Sturmian, then w is central.*

Proof. From the previous proposition, there exists a letter $x \in \mathcal{A}$ such that xwa and xwb are both Sturmian. Without loss of generality, we may suppose $x = a$, so that $awb \in St$. Therefore $(awb)^\sim = bwa$ is Sturmian too, thus by Proposition 5 w is central. □

Lemma 8. *Let w be a Sturmian palindrome. If w is not central, then there exists a unique letter $x \in \mathcal{A}$ such that xwx is Sturmian.*

Proof. If awa and bwb are both Sturmian, then w is central by Proposition 7, a contradiction. However, by Corollary 2, w is a (proper) median factor of some central word. □

Now let us introduce the function $g : \mathbb{N} \to \mathbb{N}$ defined for all $n \geq 0$ as

$$g(n) := \mathrm{card}(St \cap PAL \cap \mathcal{A}^n) \ .$$

For any $n \geq 0$, $g(n)$ counts the number of Sturmian palindromes of length n.

Theorem 9. *For any $n \geq 0$, the number $g(n)$ of Sturmian palindromes of length n is given by:*

$$1 + \sum_{i=0}^{\lceil n/2 \rceil - 1} \phi(n - 2i) \tag{1}$$

where ϕ is Euler's totient function. Equivalently, for any $n \geq 0$

$$g(2n) = 1 + \sum_{i=1}^{n} \phi(2i) \quad \textit{and} \quad g(2n+1) = 1 + \sum_{i=0}^{n} \phi(2i+1) \ .$$

Proof. Given $w \in St \cap PAL$, at least one of its "extensions" awa and bwb is Sturmian. Indeed, according to Lemma 8, if $w \notin PER$, then exactly one of these extensions is in St. If $w \in PER$, then from Proposition 5, both awa and bwb are Sturmian palindromes. Since the number of central words of length n is $\phi(n+2)$ (see [8]), one gets:

$$g(n+2) = g(n) + \phi(n+2)$$

and this implies the desired formula, because $g(0) = 1$ and $g(1) = 2$. □

We define a function $f : \mathbb{N} \to \mathbb{N}$ by setting, for $n \geq 0$:

$$f(2n) = 1 + \frac{n(n+1)}{2} \text{ and } f(2n+1) = 2 + n(n+1) \ .$$

It is easy to verify that $g(n) \leq f(n)$ for all $n \geq 0$. Moreover, for any $n \geq 0$ we set $h(n) = \text{card}(PER \cap \mathcal{A}^n) = \phi(n+2)$. In Table 1 we list the values of the functions g, f, and h for $0 \leq n \leq 17$.

Table 1. The functions g, f, and h

n	$g(n)$	$f(n)$	$h(n)$	n	$g(n)$	$f(n)$	$h(n)$
0	1	1	1	9	20	22	10
1	2	2	2	10	14	16	4
2	2	2	2	11	30	32	12
3	4	4	4	12	18	22	6
4	4	4	2	13	42	44	8
5	8	8	6	14	24	29	8
6	6	7	4	15	50	58	16
7	14	14	6	16	32	37	6
8	10	11	4	17	66	74	18

As an example, in Table 2 we report the set of all 14 Sturmian palindromes of length 7. The six central words in it are underlined.

4.2 Structural Properties

We have seen in Section 3 that a Sturmian palindrome is a median factor of a central word. In this section we shall give some further results concerning the structure of Sturmian palindromes.

Table 2. Sturmian palindromes of length 7 (central words are underlined)

$\underline{aaaaaaa}$	$\underline{bbbbbbb}$
$\underline{aaabaaa}$	$\underline{bbbabbb}$
$aababaa$	$bbababb$
$abaaaba$	$babbbab$
$\underline{abababa}$	$\underline{babahab}$
$abbabba$	$baabaab$
$abbbbba$	$baaaaab$

Proposition 10. *A palindrome $w \in \mathcal{A}^*$ with minimal period $\pi_w > 1$ can be uniquely represented as*

$$w = w_1xyw_2 = w_2yxw_1^{\sim}$$

with $x, y \in \mathcal{A}$, w_2 the longest proper palindromic suffix of w, and $|w_1xy| = \pi_w$. The word w is not central if and only if either $w_1 \notin PAL$ or $w = (w_1xx)^k w_1$ where $k \geq 1$ is the order of w.

Proof. As is well known (cf. [7]), a palindrome w has a period $p < |w|$ if and only if it has a palindromic suffix (prefix) of length $|w| - p$. Since the minimal period π_w of a palindrome is less than $|w|$ and $\pi_w > 1$, it follows that w can be uniquely factorized as $w = w_1xyw_2$ where w_2 is the longest proper palindromic suffix of w and $|w_1xy| = \pi_w$. Since w is a palindrome, we can write

$$w = w_1xyw_2 = w_2yxw_1^{\sim} \ .$$

From a classic result on central words (see [7]), when $\pi_w > 1$, w is central if and only if $w_1 \in PAL$ and $x \neq y$. Therefore, in the case $w_1 \in PAL$, w is not central if and only if $w = w_1xxw_2 = w_2xxw_1$. The word w has the two periods

$$\pi_w = |w_1xx| \ \text{ and } q = |w_2xx| \tag{2}$$

and length $\pi_w + q - 2$. Thus $w \notin PER$ if and only if $d = \gcd(\pi_w, q) > 1$. Since $|w| \geq \pi_w + q - d$, by Fine and Wilf's theorem (cf. [10]) w has the period $d \geq \pi_w$. This occurs if and only if $q = k\pi_w$ with $k \geq 1$. From (2) this condition is equivalent to the statement $w_2xx = (w_1xx)^k$, i.e., $w = (w_1xx)^k w_1$. □

Example 11. Let $w = aaabaaaaaabaaa \in St \cap PAL$, with $\pi_w = 7$. The word w can be factorized as $(aaaba)aa(aaabaaa)$, where $aaabaaa$ is the longest proper palindromic suffix of w, $|aaaba| = \pi_w - 2 = 5$. The prefix $aaaba$ is not a palindrome, thus w is not central.

Let $v = abaababababaaba \in St \cap PAL$. We factorize v as

$$v = (abaabab)ab(abaaba)$$

where $abaaba$ is the longest proper palindromic suffix of v. Also in this case $abaabab$ is not a palindrome, so that $w \notin PER$.

Let $u = abbabbabba \in St \cap PAL$. We factorize u as $(a)bb(abbabba)$, where $abbabba$ is the longest palindromic suffix of u. In this case, the prefix a is a palindrome, and $u = (abb)^3a$. Hence u is not central.

Lemma 12. *If $w = w_1xyw_2 = w_2yxw_1^{\sim}$, where w_2 is the longest proper palindromic suffix of w and $x, y \in \mathcal{A}$, then $w' = ywy$ has the minimal period $\pi_{w'} = \pi_w$.*

Proof. Since w is a factor of w', one has $\pi_{w'} \geq \pi_w$. The word yw_2y is a palindromic proper suffix of $w' = yw_1xyw_2y$, so that w' has the period $|yw_1x|$. Hence, $\pi_{w'} \leq |yw_1x| = |w_1xy| = \pi_w$. Thus $\pi_w = \pi_{w'}$. □

The next lemma is essentially a restatement of Lemma 2 in [5]. Note that its first part is an obvious consequence of Lemma 12.

Lemma 13. *Let $w = w_1xyw_2 = w_2yxw_1 \in PER$, with $|w_2| > |w_1|$, $\{x, y\} = \mathcal{A}$. The word $v = ywy$ has minimal period $\pi_v = \pi_w$, whereas $v' = xwx = xw_1xyw_2x$ has minimal period $\pi_{v'} = |w_2| + 2 = |w| - \pi_w + 2$.*

Let $w \in (St \cap PAL) \setminus PER$. We denote by u the (unique) shortest median extension of w in PER, and by v the longest central median factor of w. Note that also v is unique.

Theorem 14. *Let $w \in (St \cap PAL) \setminus PER$. With the preceding notation, one has $\pi_u = \pi_w$. Moreover, either $\pi_w = \pi_v$ or $\pi_w = |v| - \pi_v + 2$.*

Proof. We consider first the case that $\pi_v = 1$, so that $v = x^n$ with $x \in \mathcal{A}$. In such a case w has also the median palindromic factor $v_1 = yx^ny$, where $\{x, y\} = \mathcal{A}$ (recall that v is the longest central median factor of w). Moreover, $n = |v|$ is at least 2, otherwise v_1 would be equal to $yxy \in PER$. One has $\pi_{v_1} = |yx^n| = n + 1 = |v| - \pi_v + 2$. Now we define, for $2 \leq i \leq n$:

$$v_i = xv_{i-1}x = x^{i-1}yx^nyx^{i-1} = (x^{i-1}yx^{n-i+1})(x^{i-1}yx^{i-1}) \ . \tag{3}$$

The word $v_n = x^{n-1}yx^nyx^{n-1}$ is central, whereas for $i < n$ one has $v_i \notin PER$. From Lemma 8 it follows that the words v_i are the *only* Sturmian extensions of v_1 which are median factors of v_n. Since for $i < n$ one has $v_i \notin PER$, one derives that $w = v_k$ for some $1 \leq k < n$, and $u = v_n$. As shown in (3), by Lemma 12 all the v_i's have the same period, for $1 \leq i \leq n$. The result in this case follows: $\pi_w = \pi_u = |v| - \pi_v + 2$.

Now let us assume $\pi_v > 1$. In this case $v = w_1xyw_2 = w_2yxw_1$, with $w_1, w_2 \in PAL$, $x \neq y$. We suppose $|w_1| < |w_2|$, so that $\pi_v = |w_1| + 2$. From the definition of v, it follows that there exists a letter $z \in \mathcal{A}$ such that $v_1 = zvz$ is a median factor of w which is not central. By Lemma 13, we have $\pi_{v_1} = \pi_v$ if $z = y$, or else $\pi_{v_1} = |v| - \pi_v + 2$ if $z = x$.

Using Lemma 12, we shall now define a sequence of palindromes with the same minimal period as v_1. Let us first suppose that $z = y$, so that $v_1 = yw_1xyw_2y$. We set $v_2 = xv_1x = (xyw_1)(xyw_2yx)$. Moreover, if $w_1 = p_1p_2 \cdots p_k$ with $p_j \in \mathcal{A}$ for $1 \leq j \leq k$, we set $v_i = p_{k-i+3}v_{i-1}p_{k-i+3}$ for $i \geq 3$, so that

$$\begin{aligned} v_3 &= p_kv_2p_k = (p_kxyp_1 \cdots p_{k-1})(p_kxyw_2yxp_k) \ , \\ &\vdots \\ v_{k+2} &= p_1v_{k+1}p_1 = p_1 \cdots p_kxyw_1xyw_2yxp_k \cdots p_1 = w_1xyw_1xyw_2yxw_1^{\sim} \ . \end{aligned}$$

Since $w_1 = w_1^{\sim}$, the last equation can be written as

$$v_{k+2} = (w_1)xy(w_1xyw_2yxw_1) = (w_1xyw_2yxw_1)yx(w_1)$$

showing, by Proposition 10, that the word v_{k+2} is central, so that for any $i \leq \pi_v = k+2$, $v_i \in St \cap PAL$. Let $s \leq k+2$ be the minimal integer such that $v_s \in PER$. Since for $i < s$ one has $v_i \notin PER$, and from Lemma 8, one derives that $u = v_s$ and $w = v_r$ for some integer $r < s$. One has $\pi_w = \pi_{v_s} = \pi_u$, and in this case $\pi_w = \pi_v$.

The case $z = x$ is similarly dealt with, but interchanging the roles of w_1 and w_2. Thus one assumes $w_2 = q_1 \cdots q_k$, and defines v_i as $q_{k-i+3}v_{i-1}q_{k-i+3}$ for $i \geq 3$, starting from $v_2 = yv_1y = (yxw_2)(yxw_1xy)$ and ending with

$$v_{k+2} = w_2yxw_2yxw_1xyw_2 \in PER \ .$$

Therefore there exist integers r, s such that $1 \leq r < s \leq k+2 = |v| - \pi_v + 2$, $w = v_r$, and $u = v_s$, so that $\pi_w = \pi_u$ and $\pi_w = \pi_{v_1} = |v| - \pi_v + 2$. □

Example 15. Let $w = baaabaaab \in (St \cap PAL)$. Following the notations of Theorem 14, one has $v = aaabaaa$, $v_1 = w$, and $u = v_3 = aabaaabaaabaa$. Thus $\pi_w = \pi_u = \pi_v = 4$.

Let $w = babbbbab$. In this case we have $v = bbbb$, $w = v_2$, and $u = v_4 = bbbabbbbabbb$, so that $\pi_w = \pi_u = 5 = |v| + 1 = |v| - \pi_v + 2$.

For any word $w \in \mathcal{A}^*$, we denote by R_w the minimal nonnegative integer such that there is no right special factor of w of length R_w, and by K_w the length of the shortest unrepeated suffix of w. Conventionally, one assumes $R_\varepsilon = K_\varepsilon = 0$. The following theorem gives a further criterion, different from Proposition 10, to discriminate whether a (Sturmian) palindrome over $\mathcal{A}$ is central or not.

Theorem 16. *Let $w \in \mathcal{A}^*$ be a palindrome, with $\pi_w > 1$. Then w is central if and only if its prefix of length $\pi_w - 2$ is a right special factor of w.*

Proof. From Proposition 10, we can write

$$w = w_1xyw_2 = w_2yxw_1^{\sim} \tag{4}$$

where $x, y \in \mathcal{A}$, w_2 is the longest proper palindromic suffix of w, $|w_1| = \pi_w - 2$, and w is central if and only if $w_1 \in PAL$ and $x \neq y$. Therefore we have to prove that w_1 is a right special factor of w if and only if $w_1 = w_1^{\sim}$ and $x \neq y$.

Indeed, assume that these two latter conditions are satisfied. Since $w_1^{\sim} = w_1$ and w_2 is the longest proper palindromic suffix (and prefix) of w, one has that w_1 is a proper prefix and suffix of w_2. This implies, from (4), that w_1 is a right special factor of w.

Conversely, suppose w_1 is a right special factor of w. Let us first prove that $w_1 \in PAL$. By hypothesis, we have $\pi_w - 2 = |w_1| \leq R_w - 1$, that is $R_w \geq \pi_w - 1$. Since in general one has $\pi_w \geq R_w + 1$ (see [6, Corollary 5.3]), it follows $\pi_w = R_w + 1$. This implies $|w| = R_w + K_w$, again by [6, Corollary 5.3]. The suffix $w_1^{\sim}$

of w is repeated, because w_1 is a right special factor of w, which is a palindrome. This leads to

$$\pi_w - 2 = |w_1^{\sim}| \leq K_w - 1$$

and thus to $|w| = R_w + K_w \geq 2\pi_w - 2$. If $|w| = 2\pi_w - 2$, then $|w_1| = |w_2|$ so that one derives $w_1 = w_2 \in PAL$. If $|w| \geq 2\pi_w - 1$, then w has the prefix $w_1 x y w_1 x$, so that $y w_1 x \in \mathrm{Fact}\,(w)$. Let z be the letter such that $\mathcal{A} = \{x, z\}$. The word $w_1 z$ is a factor of w because w_1 is right special. Moreover, since $w_1 z$ is not a prefix, there exists a letter y' such that $y' w_1 z \in \mathrm{Fact}\,(w)$. One has $y \neq y'$, for otherwise $y w_1$ would be a right special factor of w of length $\pi_w - 1 = R_w$, which is a contradiction. As w is a palindrome, the words $x w_1^{\sim} y$ and $z w_1^{\sim} y'$ are factors of w too, so that $w_1^{\sim}$ is a right special factor of w. By [6, Proposition 4.7], this implies $w_1 = w_1^{\sim}$. Therefore we get $w_1 \in PAL$ again.

We shall now prove that $x \neq y$. By contradiction, suppose w has the factorization

$$w = (w_1 x x)^k w_1 \text{ , with } k \geq 1$$

as granted by Proposition 10. As before, assume $\mathcal{A} = \{x, z\}$. Since w_1 is a right special factor of w, one has $w_1 z \in \mathrm{Fact}\,(w)$. Thus we have either $w_1 z = x w_1$ or $w_1 z = v_2 x x v_1 z$, where $v_1 z$ is a prefix of w_1 and v_2 is a suffix of w_1. Since $|w_1| = |w_1 z| - 1$, we can write $w_1 = v_1 z \alpha v_2$, with $\alpha \in \mathcal{A}$. The first case is impossible since w_1 is a palindrome and $x \neq z$. In the latter case, one obtains:

$$v_1 z \alpha v_2 = w_1 = w_1^{\sim} = v_1^{\sim} x x v_2^{\sim}$$

which is absurd again, because $x \neq z$. □

Example 17. The word $w = baab$ is a Sturmian palindrome of minimal period $\pi_w = 3$. Its prefix of length 1 is not a right special factor, hence $w \notin PER$. The word $v = abababbababa$ is a Sturmian palindrome having minimal period 7, and its prefix $ababa$ of length 5 is not right special. Therefore $v \notin PER$. On the contrary, the word $u = aabaabaa$ has minimal period 3, and its prefix of length 1 is a right special factor, so that u is central.

Proposition 18. *A palindrome $w \in \mathcal{A}^*$ is Sturmian if and only if $\pi_w = R_w + 1$.*

Proof. The result is trivially true if $\pi_w = 1$. Since for any $w \in \mathcal{A}^*$ one has $\pi_w \geq R_w + 1$ (cf. [6]), in the case $\pi_w > 1$ the condition $\pi_w = R_w + 1$ is equivalent to the existence of a right special factor s of w of length $|s| = \pi_w - 2$.

Let us first prove that every Sturmian palindrome w such that $\pi_w \geq 2$ has such a factor. If w is central, the result follows directly from Theorem 16. Thus we suppose $w \notin PER$, and as in Theorem 14 we denote by v the central median factor of w of maximal length. If $\pi_v = 1$, then there exists a letter $x \in \mathcal{A}$ and an integer $n \geq 1$ such that $v = x^n$. From the maximality condition, one has $n > 1$. In this case, by Theorem 14 one derives $\pi_w = |v| + 1 = n + 1$ and $y x^n y \in \mathrm{Fact}\,(w)$, where $\{x, y\} = \mathcal{A}$; therefore x^{n-1} is the desired right special factor of w, of length $n - 1 = \pi_w - 2$. If $\pi_v > 1$, using Proposition 10 we write v as $v_1 x y v_2$, with $\pi_v = |v_1 x y|$. By Theorem 14, one has either $\pi_w = \pi_v$ or

$\pi_w = |v| - \pi_v + 2$. In the first case, the result is a consequence of Theorem 16. Indeed, the prefix v_1 of the central word v, whose length is $\pi_v - 2 = \pi_w - 2$, is a right special factor of v, and then of w. In the latter case, one derives that the word $xvx = xv_1xyv_2x = xv_2yxv_1x$ is a factor of w, so that v_2 is a right special factor of w, of length $|v| - \pi_v = \pi_w - 2$.

Conversely, let us assume that w has a right special factor s with $|s| = \pi_w - 2$. By Proposition 10, one can write

$$w = w_1xyw_2 = w_2yxw_1^{\sim}$$

with $x, y \in \mathcal{A}$, $|w_1xy| = \pi_w$. Moreover, let k be the order of w. By definition, k is the maximal integer such that $(w_1xy)^k$ is a prefix of w. This implies, by periodicity, that w is a prefix of $(w_1xy)^{k+1}$. By applying repeatedly Lemma 12, one obtains that the word $u = (w_1xy)^{k+1}w(yxw_1^{\sim})^{k+1}$ has minimal period π_w. Since $s \in \mathrm{Fact}\,((w_1xy)^{k+1})$, there exists a median factor t of u which starts with s and has w as a factor, i.e., $t = s\delta w\delta^{\sim}s^{\sim}$ for some word δ. We have $\pi_w \leq \pi_t \leq \pi_u = \pi_w$, therefore from Theorem 16 it follows that $t \in PER$, whence $w \in St$. □

Example 19. Let $w = abba \in St \cap PAL$. One has $\pi_w = 3 = R_w + 1$. The Sturmian word $u = ababaa$ is not a palindrome, but $\pi_u = 5 = R_u + 1$. However, the word $v = aabab \in St$ has $\pi_v = 5 > 3 = R_w + 1$.

The palindrome word $s = aabbaa$ is not Sturmian. One has $\pi_s = 4 > 3 = R_s + 1$.

References

1. Allouche, J.P., Shallit, J.: Chapters 9–10. In: Automatic Sequences. Cambridge University Press, Cambridge UK (2003)
2. Berstel, J., Séébold, P.: Sturmian words. In Lothaire, M., ed.: Algebraic Combinatorics on Words, Chapter 2. Cambridge University Press, Cambridge UK (2001)
3. Borel, J.P., Reutenauer, C.: Palindromic factors of billiard words. Theoretical Computer Science, to appear (2005)
4. Carpi, A.: Private communication (2005)
5. D'Alessandro, F.: A combinatorial problem on Trapezoidal words. Theoretical Computer Science **273** (2002) 11–33
6. de Luca, A.: On the combinatorics of finite words. Theoretical Computer Science **218** (1999) 13–39
7. de Luca, A.: Sturmian words: structure, combinatorics, and their arithmetics. Theoretical Computer Science **183** (1997) 45–82
8. de Luca, A., Mignosi, F.: Some combinatorial properties of Sturmian words. Theoretical Computer Science **136** (1994) 361–385
9. Droubay, X., Pirillo, G.: Palindromes and Sturmian words. Theoretical Computer Science **223** (1999) 73–85
10. Lothaire, M.: Combinatorics on Words. Addison-Wesley, Reading MA (1983)

Voronoi Cells of Beta-Integers

Avi Elkharrat[1] and Christiane Frougny[2]

[1] LPTMC, Université Paris 7, France
kharrat@ccr.jussieu.fr
[2] LIAFA, CNRS & Université Paris 7, and Université Paris 8, France
Christiane.Frougny@liafa.jussieu.fr

Abstract. In this paper are considered one-dimensional tilings arising from some Pisot numbers encountered in quasicrystallography as the quadratic Pisot units and the cubic Pisot unit associated with 7-fold symmetry, and also the Tribonacci number. We give characterizations of the Voronoi cells of such tilings, using word combinatorics and substitutions.

1 Introduction

Word combinatorics has been proved to be very useful in the solution of problems arising from the modelization of metallic alloys called *quasicrystals*. The first quasicrystal was discovered in 1984: it is a solid structure presenting a local symmetry of order 5, *i.e.* a local invariance under rotation of $\pi/5$, and it is linked to the golden mean and to the Fibonacci substitution. The Fibonacci substitution, given by

$$L \mapsto LS,\ S \mapsto L,$$

defines a quasiperiodic selfsimilar tiling of the positive real line, and is a historical model of a one-dimensional mathematical quasicrystal. The fixed point of the substitution is the infinite word

$$LSLLSLSLSL\cdots$$

Each letter L or S is considered as a tile. The vertices of the tiles are labelled by algebraic integers, the so-called β-integers, where β is equal to $\frac{1+\sqrt{5}}{2}$. The description and the properties of those β-integers use a base β number system.

A more general theory has been elaborated with Pisot numbers[1] for base, see [3, 6]. Note that so far, all the quasicrystals discovered by physicists present local symmetry of order 5, 8, 10, or 12, and are modelized using some quadratic Pisot units, namely $\frac{1+\sqrt{5}}{2}$, $1+\sqrt{2}$, and $2+\sqrt{3}$. More generally, a substitution can be associated with any Pisot number giving a selfsimilar quasiperiodic tiling of the positive real line [19].

[1] A *Pisot number* is an algebraic integer > 1 such that the other roots of its minimal polynomial have a modulus less than 1. The golden mean and the natural integers are Pisot numbers

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 209–223, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The construction of quasiperiodic point sets involves a method called *cut and projection* [12, 13]. The determination of the quasicrystal depends on a set called *window*. For instance, when β is the Tribonacci number, the window of the set of β-integers is the well known Rauzy fractal, see [11] for instance.

The purpose of this work is to give a combinatorial characterization of the geometry of tilings associated with sets of beta-integers. More precisely we show that local geometrical configurations of beta-integers, given by their *Voronoi cells*, are characterized by their beta-expansions. This allows to give a fine partition of the window associated with positive beta-integers according to the combinatorial properties of the underlying numeration system.

It is worthwile to mention that the fixed point u_β of the substitutions associated with the Pisot numbers β considered here enjoys the following properties. When β is a quadratic Pisot unit, u_β is a Sturmian sequence [6], that is to say, the number $\mathcal{C}(n)$ of factors of length n, is equal to $n+1$. When β is the Tribonacci number, u_β is an Arnoux-Rauzy sequence [1], of complexity $\mathcal{C}(n) = 2n+1$. When β is the cubic Pisot unit associated with 7-fold symmetry, u_β has complexity $\mathcal{C}(n) = 2n+1$, but is not an Arnoux-Rauzy sequence [7].

The paper is organized as follows: after some definitions, we give characterizations of the Voronoi cells of the tilings associated with quadratic Pisot units, with the Tribonacci number and, with the cubic Pisot unit associated with 7-fold symmetry. These results are given in terms of the properties of the beta-expansions as words, and by the belonging of the conjugates of beta-integers to some connected region, the window. For the Tribonacci number, our results allow to give a nice combinatorial interpretation of the domain exchange defined by Rauzy on the Rauzy fractal, see [11, 15, 16].

2 Preliminaries

2.1 Words

Let A be a finite set of symbols called the *alphabet*. We denote by A^* the set of finite *words* over A, and by ε the empty word. A *factor* of a word x is a word z such that $x = yzt$. If $y = \varepsilon$, z is said to be a *prefix* of x; if $t = \varepsilon$, the word z is a *suffix* of x. A prefix (or a suffix) z of y is *proper* if it is different of the entire word y. If v is a word, the concatenation of v k times is denoted by v^k, with the convention that if $k = 0$, v^k is the empty word ε.

A function $f : A^* \to B^*$ is a *morphism* if $f(xy) = f(x)f(y)$, for all $x, y \in A^*$. A morphism is a *substitution* if for each a in A, $f(a) \neq \varepsilon$.

The *radix order* for finite words over an ordered alphabet is defined by $x \leqslant y$ if $|x| < |y|$, or $|x| = |y|$ and their exist factorizations $x = uax'$ and $y = uby'$, for some word $u \in A^*$, $a, b \in A$ such that $a \leqslant b$, and $x', y' \in A^*$.

The set of infinite words over A is denoted by $A^{\mathbb{N}}$. It is the set of sequences of symbols of A indexed by non-negative integers. Denote by $v^\omega = vvvv\ldots$ the word obtained by the infinite concatenation of the word v. A word of the form uv^ω is called *eventually periodic* if $u \neq \varepsilon$, *periodic* otherwise.

The *lexicographic order* for infinite words over an ordered alphabet is defined by $x <_{\text{lex}} y$ if their exist factorizations $x = uax'$ and $y = uby'$, for some word $u \in A^*$, $a, b \in A$ such that $a < b$, and $x', y' \in A^{\mathbb{N}}$.

2.2 Beta-Expansions

For definitions and results on beta-expansions the reader may consult [10, Chapter 7]. Let $\beta > 1$ be a real number. A representation in base β, or a β-*representation*, of a real number $x > 0$ is an infinite sequence of integers $(x_i)_{i \leqslant N}$ such that $x = \sum_{i \leqslant N} x_i \beta^i$, for some N. A particular β-representation, called β-*expansion*, is computed by the "greedy algorithm" [17]. Denote by $\lfloor y \rfloor$ and by $\{y\}$ the integer part and the fractional part of the real number y, respectively. There exists $N \in \mathbb{Z}$ such that $\beta^N \leqslant x < \beta^{N+1}$. Let $x_N = \lfloor x/\beta^N \rfloor$, and let $r_N = \{x/\beta^N\}$. Then for $i < N$, $x_i = \lfloor \beta r_{i+1} \rfloor$, and $r_i = \{\beta r_{i+1}\}$. If $x < 1$, then $N < 0$ and we set $x_{-1} = \cdots = x_{N+1} = 0$. The β-expansion of x is denoted by

$$\langle x \rangle_\beta = x_N x_{N-1} \cdots x_1 x_0 \cdot x_{-1} x_{-2} \cdots ,$$

most significant digits first. The dot between x_0 and x_{-1} symbolizes the separation between positive and negative powers of the base. By abuse we refer to the word $x_N \cdots x_0$ as the β-*integer part*, and to the word $x_{-1}x_{-2}\cdots$ as the β-*fractional part* of x in base β. The digits x_i obtained by the greedy algorithm belong to the set $\mathbb{B} = \{0, 1, \ldots, \lfloor \beta \rfloor\}$, called the *canonical* alphabet associated with β, if β is not an integer. If β is an integer, then $\mathbb{B} = \{0, 1, \ldots, \beta - 1\}$, and the β-expansion is just the standard representation in base β. If a β-representation ends with infinitely many 0's it is said to be *finite* and the ending 0's are omitted.

A word (finite or infinite) is said to be *admissible* if it is the β-expansion of some number of $[0, 1[$. Let us introduce the so called *Rényi* β-*expansion of* 1, denoted by $d_\beta(1)$. It is computed as follows: let the β-*transform* of the unit interval be defined by $T_\beta(y) = \beta y \mod 1$. Then $d_\beta(1) = (t_i)_{i \geqslant 1}$, where $t_i = \lfloor \beta T_\beta^{i-1}(1) \rfloor$. Note that $d_\beta(1)$ belongs to $\mathbb{B}^{\mathbb{N}}$. A number β such that $d_\beta(1)$ is eventually periodic is called a *beta-number*, or a *Parry number*. When $d_\beta(1)$ is finite, β is said to be a *simple* Parry number. Set $d^*_\beta(1) = d_\beta(1)$ if $d_\beta(1)$ is eventually periodic, and $d^*_\beta(1) = (t_1 \cdots t_{m-1}(t_m - 1))^\omega$ if $d_\beta(1) = t_1 \cdots t_{m-1} t_m$ is finite. Let us recall the following result from [14]: An infinite sequence of non-negative integers $\xi = (\xi_i)_{i \geqslant 1}$ is admissible if and only if for every $p \geqslant 0$, $(\xi_{i+p})_{i \geqslant 1} <_{\text{lex}} d^*_\beta(1)$. We can now define the set of β-integers.

Definition 1 *The set of* β-integers *is the set of real numbers such that the* β-*expansion of their absolute value has a* β-*fractional part equal to* 0^ω

$$\mathbb{Z}_\beta = \{x \in \mathbb{R} \mid \langle |x| \rangle_\beta = x_N x_{N-1} \cdots x_1 x_0,\ N \geqslant 0\} . \tag{1}$$

Denote $\mathbb{Z}_\beta^+$ the set of non-negative β-integers. Note that $\mathbb{Z}_\beta = \mathbb{Z}_\beta^+ \cup (-\mathbb{Z}_\beta^+)$ and that $\beta\mathbb{Z}_\beta \subset \mathbb{Z}_\beta$. The set $\mathbb{Z}_\beta^+$ is ordered by the radix order on the (finite) β-expansions of its elements; its n-th element is denoted b_n.

An *algebraic integer* is a root of a monic polynomial with integer coefficients. A *Pisot number* is an algebraic integer greater than 1 such that the other roots of its minimal polynomial have a modulus smaller than 1. When the constant term of the minimal polynomial is equal to ± 1, β is said to be a *unit*. Recall that any Pisot number is a Parry number [2, 18].

2.3 Substitution Tilings

Let β be a Parry number. To such a number a substitution σ_β can be associated with β in a canonical way. Its fixed point u_β is written on an alphabet $\mathbb{A}$ of letters that are considered as tiles. This defines a tiling of the positive real line with a finite number of tiles. Each tile U is given a length $\ell(U)$, see [5, 19]. Each vertex of the positive real line is labelled by the length in tiles of the prefix of u_β ending in that vertex.

In the frame of this study, we restrict ourselves to a subclass of Parry numbers, namely the quadratic Pisot units and two examples of cubic Pisot units.

Quadratic Pisot Units. They are of two types.

Case 1. $\beta > 1$ is the root of the polynomial $X^2 - aX - 1$, $a \geqslant 1$. The canonical alphabet is $\mathbb{B} = \{0, 1, \ldots, a\}$, the β-expansion of 1 is finite and equal to $d_\beta(1) = a1$. Note that a word on $\mathbb{B}$ is a β-expansion if and only if it does not contain a factor $a1$. The substitution σ_β is defined on the alphabet $\mathbb{A} = \{L, S\}$ by

$$\sigma_\beta = \begin{cases} L \mapsto L^a S \\ S \mapsto L\,. \end{cases} \tag{2}$$

To each letter of $\mathbb{A}$ we associate a tile with the same name of length $\ell(L) = 1$, and $\ell(S) = T_\beta(1) = \beta - a = 1/\beta$.

Case 2. $\beta > 1$ is the root of the polynomial $X^2 - aX + 1$, $a \geqslant 3$. The canonical alphabet is $\mathbb{B} = \{0, 1, \ldots, a-1\}$, the β-expansion of 1 is eventually periodic and equal to $d_\beta(1) = (a-1)(a-2)^\omega$. The substitution σ_β is defined on the alphabet $\mathbb{A} = \{L, S\}$ by

$$\sigma_\beta = \begin{cases} L \mapsto L^{a-1} S \\ S \mapsto L^{a-2} S\,. \end{cases} \tag{3}$$

Here we have $\ell(L) = 1$, and $\ell(S) = T_\beta(1) = \beta - (a-1) = 1 - 1/\beta$.

Cubic Pisot Units. We consider two particular cases of cubic Pisot units, namely the roots of the polynomials

$$X^3 - X^2 - X - 1, \text{ Case 1}$$

$$X^3 - 2X^2 - X + 1, \text{ Case 2.}$$

The root $\beta > 1$ in Case 1 is the so-called Tribonacci number, see for instance [11]. The root β in Case 2 is a cyclotomic Pisot unit with a 7-fold symmetry, that

is to say, the ring $\mathbb{Z}[e^{i2\pi/7}]$, which is invariant by rotation of $2\pi/7$, satisfies $\mathbb{Z}[e^{i2\pi/7}] = \mathbb{Z}[\beta] + \mathbb{Z}[\beta]e^{i2\pi/7}$, see [3].

Case 1. The Tribonacci number. The canonical alphabet is $\mathbb{B} = \{0, 1\}$, the β-expansion of 1 is finite and equal to $d_\beta(1) = 111$. A word on $\mathbb{B}$ is a β-expansion if and only if it does not contain a factor 111. The substitution σ_β is defined on the alphabet $\mathbb{A} = \{L, M, S\}$ by

$$\sigma_\beta = \begin{cases} L \mapsto LM \\ M \mapsto LS \\ S \mapsto L\,. \end{cases} \tag{4}$$

We have $\ell(L) = 1$, $\ell(M) = T_\beta(1) = \beta - 1$, and $\ell(S) = T_\beta^2(1) = \beta^2 - \beta - 1$.

Case 2. Symmetry of order 7. The canonical alphabet is $\mathbb{B} = \{0, 1, 2\}$, the β-expansion of 1 is eventually periodic and equal to $d_\beta(1) = 2(01)^\omega$. The substitution σ_β is defined on the alphabet $\mathbb{A} = \{L, M, S\}$ by

$$\sigma_\beta = \begin{cases} L \mapsto LLS \\ S \mapsto M \\ M \mapsto LS \end{cases} \tag{5}$$

Here we have $\ell(L) = 1$, $\ell(S) = T_\beta(1) = \beta - 2$, and $\ell(M) = T_\beta^2(1) = \beta^2 - 2\beta$.

For all the above cases, the infinite word $u_\beta = \sigma_\beta^\infty(L)$ is the fixed point of the substitution σ_β. The interval $[0, \beta^j]$ is tiled by the tiling associated with the word $\sigma_\beta^j(L)$. Consequently the tiling associated with $\sigma_\beta^\infty(L)$ is a selfsimilar tiling of the positive real line and positive β-integers are the labels of the vertices of this tiling, see [3, 6]. The substitution σ_β acts on the tiles as the multiplication by β acts on β-integers.

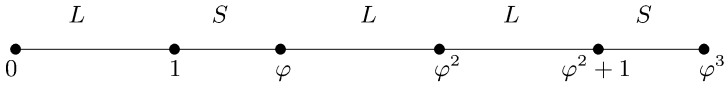
Example Let $\varphi = \frac{1+\sqrt{5}}{2}$. Then $d_\varphi(1) = 11$. The associated substitution is the Fibonacci substitution $L \mapsto LS$, $S \mapsto L$. We have $\ell(L) = 1$, and $\ell(S) = \varphi - 1$. The first non-negative φ-integers are

$$\begin{array}{ll} b_0 = 0 & \langle b_0 \rangle_\varphi = 0 \\ b_1 = 1 & \langle b_1 \rangle_\varphi = 1 \\ b_2 = \varphi & \langle b_2 \rangle_\varphi = 10 \\ b_3 = \varphi^2 & \langle b_3 \rangle_\varphi = 100 \\ b_4 = \varphi^2 + 1 & \langle b_4 \rangle_\varphi = 101 \\ b_5 = \varphi^3 & \langle b_5 \rangle_\varphi = 1000 \end{array}$$

The fixed point of the substitution is

$$u_\varphi = LSLLSLSLSL\cdots$$

Below is shown the beginning of the labelling of vertices of the Fibonacci tiling by φ-integers



2.4 Meyer Sets and Voronoi Cells

We recall here several definitions and results that can be found in [8, 9, 12, 13], see also [4] for a survey on these questions. Delaunay sets were introduced as a mathematical idealization of a solid-state structure. A set Λ in $\mathbb{R}^d$ is said to be *uniformly discrete* if there exists $r > 0$ such that every ball of radius r contains at most a point of Λ. A set Λ in $\mathbb{R}^d$ is said to be *relatively dense* if there exists $R > 0$ such that every ball of radius R contains at least a point of Λ. If both conditions are satisfied, Λ is said to be a *Delaunay set*.

Meyer introduced in [12, 13] the mathematical notion of *quasicrytals* as a generalization of ideal crystalline structures. They are now known as *Meyer sets*. A set $\Lambda \subset \mathbb{R}^d$ is said to be a *Meyer set* if it is a Delaunay set and if there exists a finite set F such that $\Lambda - \Lambda \subset \Lambda + F$. This is equivalent to $\Lambda - \Lambda$ being a Delaunay set [8]. The Meyer sets generalize the lattices of crystallography, that obey the relation $\Lambda - \Lambda \subset \Lambda$.

We now give the definition of Voronoi cells and of Voronoi tessellation.

Definition 2 *(i) Given a discrete set Λ in $\mathbb{R}^d$, the* Voronoi cell $\mathcal{V}(\lambda)$ *of $\lambda \in \Lambda$ is the closure of the set of all points in $\mathbb{R}^d$ closer to λ than to any other point of Λ*

$$\mathcal{V}(\lambda) = \{x \in \mathbb{R}^d \mid \delta(x - \lambda) \leqslant \delta(x - \lambda'),\ \lambda' \in \Lambda\}\,, \tag{6}$$

where δ is the Euclidean distance in $\mathbb{R}^d$.

(ii) The set of Voronoi cells of a discrete set Λ forms a tiling of $\mathbb{R}^d$ called the Voronoi tessellation *of $\mathbb{R}^d$ induced by Λ.*

Lagarias has proved in [9] that if Λ is a Meyer set, its Voronoi tessellation contains a finite number of tiles. It is proved in [3] that when β is a Pisot number, then the set $\mathbb{Z}_\beta$ of β-integers is a Meyer set.

There is a special class of Meyer sets, defined by Meyer [12, 13], called *model sets*, computed by the so called *cut and project algorithm* and in which arises the notion of *window*. In the frame of this article, we introduce the algebraic version of the cut and project algorithm in the particular cases we study.

Quadratic Pisot Units. Let $\beta > 1$ be a quadratic Pisot unit. Let now β' be the other root of the minimal polynomial associated with β, and let the *Galois conjugation automorphism* be the map $x = \sum_{0 \leqslant i \leqslant N} x_i \beta^i \mapsto x' = \sum_{0 \leqslant i \leqslant N} x_i \beta'^i$. We define the *window* of positive β-integers as the compact set Ω

$$\Omega = \overline{\{x' \mid x \in \mathbb{Z}_\beta^+\}} = \overline{(\mathbb{Z}_\beta^+)'}\,. \tag{7}$$

We know from [3] that a a number x of $\mathbb{Z}[\beta] \cap R^+$ is a positive β-integer if and only if its conjugate x' belongs to the window $\Omega = (-1, \beta)$ in Case 1, and $\Omega = (0, \beta)$ in Case 2.

Cubic Pisot Units. We have to consider our two cases separately.
Case 1. Let $\beta > 1$ be the Tribonacci number, and let α and α^c be its Galois conjugates (the symbol ^{c} denotes complex conjugation). The Galois conjugation automorphism is defined as $x = \sum_{0 \leqslant i \leqslant N} x_i \beta^i \mapsto x' = \sum_{0 \leqslant i \leqslant N} x_i \alpha^i$. Then the

window Ω of $\mathbb{Z}_\beta^+$ is a compact subset of $\mathbb{C}$ with a fractal boundary, see for instance Figure 3. This figure is called the Rauzy fractal [11, 16].
Case 2. Let $\beta > 1$ be the dominant root of the polynomial $X^3 - 2X^2 - X + 1$. The other roots of this polynomial are the real numbers $\alpha_1 = \beta^2 - 2\beta$ and $\alpha_2 = -\beta^2 + \beta + 2$. The Galois automorphism is $x = \sum_{0 \leqslant i \leqslant N} x_i \beta^i \mapsto x' = \sum_{0 \leqslant i \leqslant N} x_i (\alpha_1^i + \alpha_2^i e^{i4\pi/7})$.

The definition of the window Ω of positive β-integers is again given by Equation (7). Note that, unless for quadratic Pisot units, the determination of the window of positive β-integers is an open problem, see discussion in [6].

3 Beta-Integers Voronoi Cells

We shall now study the Voronoi tessellation of $\mathbb{Z}_\beta^+$, and characterize Voronoi cells of β-integers when β is a quadratic Pisot unit, the Tribonacci number, or a cubic Pisot unit associated with 7-fold symmetry.

When a β-integer is the common vertex of the generic tiles U and V, it is said to be an UV β-integer, and its Voronoi cell is consequently said to be an UV Voronoi cell. The window associated with positive UV β-integers is denoted by Ω_{UV}, and is given by

$$\Omega_{UV} = \overline{\{x' \mid x \in \mathbb{Z}_\beta^+,\, x \text{ is } UV\}}\,.$$

Since a negative β-integer b_{-n} is by definition equal to $-b_n$, by symmetry one obtains a tiling of the negative real line, and thus the beta-integer $b_0 = 0$ is always of type LL.

3.1 Quadratic Pisot Units

Recall that from the substitution σ_β we have only three possible tile-configurations, LL, LS and SL, since SS is excluded, so there are only three possible Voronoi cells. When a β-integer is SL or LS, the length of its Voronoi cell is $(1 + 1/\beta)/2$ in Case 1, and $(2 - 1/\beta)/2$ in Case 2. Figure 1 displays the case when the n^{th} β-integer is SL. When a β-integer is LL the length of its Voronoi cell is 1.

We will see in the following that it is possible to further differentiate Voronoi cells, from the analysis of the β-expansion of the β-integer they support.

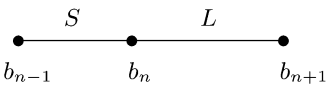


Fig. 1. Configuration where the n^{th} β-integer, b_n, is SL

Case 1. $\beta^2 = a\beta + 1$, $a \geqslant 1$

Proposition 1 *In each of the following assertions, (i), (ii) and (iii) are equivalent.*

1.1 (i) b_n is SL; (ii) $\langle b_n \rangle_\beta$ ends in the suffix $(0)^{2q+1}$, $q \in \mathbb{N}$ maximal; (iii) $b'_n \in \Omega_{SL} = (-1, 0)$.

1.2 (i) b_n is LL; (ii) for $n \geqslant 1$, $\langle b_n \rangle_\beta$ ends in either the suffix $(0)^{2q+2}$, $q \in \mathbb{N}$ maximal, or by an $h \in \{1, \ldots, a-1\}$; (iii) $b'_n \in \Omega_{LL} = [0, \beta - 1)$.

1.3 (i) b_n is LS; (ii) $\langle b_n \rangle_\beta$ ends in a; (iii) $b'_n \in \Omega_{LS} = (\beta - 1, \beta)$.

Proof We prove Proposition 1 in two steps. First we prove that $(i) \Leftrightarrow (ii)$ by recurrence. Recall that the set $\mathbb{Z}_\beta$ is symmetric with respect to the origin by definition, which makes the origin $b_0 = 0$ a LL β-integer. We have $b'_0 = 0$. The relations are clearly valid for small $n \geqslant 1$.

Suppose that b_n is an SL β-integer. In the fixed point of the substitution $u_\beta = \sigma_\beta^\infty(L)$, the factor SL can appear in two different configurations.

In the first configuration SL is a factor of $L^a SL^a S$. We thus have $b_n = \beta b_k$, where b_k is a LL β-integer,

$$\begin{array}{ccc} & b_n & \\ L^a S & | & L^a S \\ \uparrow & & \uparrow \\ & b_k & \\ L & | & L \end{array}$$

where the upright arrow symbolizes the action of the substitution σ_β on a given tile. By recurrence hypothesis, $\langle b_k \rangle_\beta$ ends either with an $h \in \{1, 2, \ldots, a-1\}$ and then $\langle b_n \rangle_\beta$ ends in $h0$, or $\langle b_k \rangle_\beta$ ends in $(0)^{2q+2}$, $q \in \mathbb{N}$ maximal and then $\langle b_n \rangle_\beta$ ends in $(0)^{2q+3}$, thus $\langle b_n \rangle_\beta$ ends with an odd number of 0's. Consequently, the β-expansions of $b_{n+1}, b_{n+2}, \ldots, b_{n+a-1}$, which are all LL β-integers, end in $1, 2, \ldots, (a-1)$ respectively, and the β-expansion of b_{n+a}, which is LS, ends in a.

In the second configuration SL is a factor of $L^a SLL^a S$ thus $b_n = \beta b_k$ where b_k is a LS β-integer. Since by recurrence hypothesis, $\langle b_k \rangle_\beta$ ends in a, then $\langle b_n \rangle_\beta$ ends with $a0$. Recall that $a1$ is not admissible for such a β. Thus the β-expansion of b_{n+1}, which is LL, ends in $j00$, where $j \neq 0$. Therefore, the β-expansion of $b_{n+2}, \ldots, b_{n+a}$, which are all LL, ends in $1, 2, \ldots, a-1$, respectively, and the β-expansion of b_{n+a+1}, which is LS, ends in a.

The cases where b_n is an LL or an LS β-integer have been already treated just above.

Let us now show that $(i) \Leftrightarrow (iii)$. Recall that a number x of $\mathbb{Z}[\beta] \cap R^+$ is a positive β-integer if and only if its conjugate x' belongs to $(-1, \beta)$. Let b_n be a SL β-integer. Then $b_{n-1} = b_n - \frac{1}{\beta}$ and $b_{n+1} = b_n + 1$ are β-integers, and $(b_{n-1})' = b'_n + \beta \in (-1, \beta)$ and $(b_{n+1})' = b'_n + 1 \in (-1, \beta)$. Therefore $b'_n \in (-1, 0)$.

Let b_n be a LL β-integer. Then $b_{n-1} = b_n - 1$ and $b_{n+1} = b_n + 1$ are β-integers, and $(b_{n-1})' = b'_n - 1 \in (-1, \beta)$ and $(b_{n+1})' = b'_n + 1 \in (-1, \beta)$. Since 0 is LL we have $b'_n \in [0, \beta - 1)$.

Finally, let b_n be a LS β-integer. Then $b_{n-1} = b_n - 1$ and $b_{n+1} = b_n + \frac{1}{\beta}$ are β-integers, and $(b_{n-1})' = b'_n - 1 \in (-1, \beta)$ and $(b_{n+1})' = b'_n - \beta \in (-1, \beta)$. Therefore $b'_n \in (\beta - 1, \beta)$. □

For LL β-integers we can refine the characterization. We give a partition of the window of LL β-integers as

$$\Omega_{LL} = \bigcup_{0 \leqslant h \leqslant a-1} \Omega_{LL}(h)\,,$$

where $\Omega_{LL}(h)$ is the window associated with positive LL β-integers such that their β-expansions end in $h \in \{0, \cdots, a-1\}$.

Proposition 2 *Let b_n be a LL β-integer, then $\langle b_n \rangle_\beta$ ends in*

2.1 $(0)^{2q+2}$*, q maximal in* $\mathbb{N}$*, if and only if* $b'_n \in \Omega_{LL}(0) = [0, \frac{1}{\beta})$,

2.2 an $h \in \{1, \ldots, a-1\}$ *if and only if* $b'_n \in \Omega_{LL}(h) = (\frac{1}{\beta} + h - 1, \frac{1}{\beta} + h)$.

Proof Let us first prove *2.1*. Let b_n be a LL β-integer, such that $\langle b_n \rangle_\beta$ ends in $(0)^{2q+2}$, $q \in \mathbb{N}$. By Proposition 1, *1.1*, the β-integer b_n/β is SL, and $(b_n/\beta)' = -\beta b'_n \in (-1, 0)$. Therefore $b_n \in (0, 1/\beta)$.

We prove *2.2* in two steps. Suppose that b_n is a LL β-integer such that $\langle b_n \rangle_\beta$ ends in 1. There are two cases for $b_{n-1} = b_n - 1$.

• b_{n-1} is a LL β-integer such that $\langle b_{n-1} \rangle_\beta$ ends in $(0)^{2q+2}$, $q \in \mathbb{N}$. Thus $b'_{n-1} \in [0, \frac{1}{\beta})$, and $b'_n \in [1, 1 + \frac{1}{\beta})$.

• b_{n-1} is a SL β-integer such that $\langle b_{n-1} \rangle_\beta$ ends with $h0$, $h \neq a$. The conjugate b'_n of b_n lies in the window computed as follows. Let b_s be a SL β-integer such that $\langle b_s \rangle_\beta$ ends with $a0$. Then b_s/β ends with a, and by Proposition 1, *1.3*, $(b_s/\beta)' \in (\beta - 1, \beta)$. Thus $b'_s \in (-1, -1 + 1/\beta)$. It is obvious now that $(b_{n-1})' \in (-1 + 1/\beta, 0)$, and $b'_n \in (\frac{1}{\beta}, 1)$.

Putting the two cases together we get that $b'_n \in (\frac{1}{\beta}, 1 + \frac{1}{\beta})$. The end of the proof for any h in $\{1, \ldots, a-1\}$ follows easily. □

On Figure 2 we display the window Ω of $\mathbb{Z}^+_\beta$, when β is a quadratic Pisot unit of Case 1, and its decomposition into subwindows which correspond to the windows of β-integers having specific Voronoi cells.

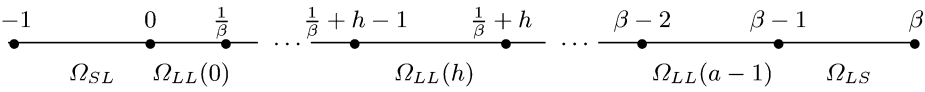


Fig. 2. Graphical representation of Proposition 1 *(iii)* and Proposition 2

Case 2. $\beta^2 = a\beta - 1$, $a \geqslant 3$

Let $\mathcal{M} = (a-1)(a-2)^*$ be the set of maximal words of each length in the radix order. Remark that any word in $\mathcal{M}$ is a prefix of $d_\beta(1) = (a-1)(a-2)^\omega$.

Proposition 3 *In each of the following assertions,* (i), (ii) *and* (iii) *are equivalent. For* $n \geqslant 1$

3.1 (i) b_n *is SL; (ii)* $\langle b_n \rangle_\beta$ *ends in* 0*; (iii)* $b'_n \in \Omega_{SL} = (0, 1)$.

3.2 (i) b_n *is LL; (ii)* $\langle b_n \rangle_\beta$ *ends in a word* $w \notin \mathcal{M} \cup 0$*; (iii)* $b'_n \in \Omega_{LL} = [1, \beta - 1)$.

3.3 (i) b_n *is LS; (ii)* $\langle b_n \rangle_\beta$ *ends in a word* $w \in \mathcal{M}$*; (iii)* $b'_n \in \Omega_{LS} = (\beta - 1, \beta)$.

Proof. We prove $(i) \Leftrightarrow (ii)$ by recurrence. Suppose that b_n is SL. The factor SL can appear in three different configurations in the fixed point of the substitution u_β.

In the first configuration SL is a factor of $L^{a-1}SL^{a-1}S$ where b_n is issued from the LL β-integer b_k, $b_n = \beta b_k$.

$$\begin{array}{ccc} & b_n & \\ L^{a-1}S & | & L^{a-1}S \\ \uparrow & & \uparrow \\ & b_k & \\ L & | & L \end{array}$$

By recurrence hypothesis $\langle b_k \rangle_\beta$ ends in $w \notin \mathcal{M} \cup 0$. Then $\langle b_n \rangle_\beta$ ends in $w0$. Consequently, the β-expansions of $b_{n+1}, b_{n+2}, \ldots, b_{n+a-2}$, which are all LL, end respectively by $w1, w2, \ldots, w(a-2)$, and the β-expansion of b_{n+a-1}, which is LS, ends in $(a-1)$, which belongs to $\mathcal{M}$.

In the second configuration, SL is a factor of $L^{a-1}SL^{a-2}S$, and $b_n = \beta b_k$ with b_k a LS β-integer.

$$\begin{array}{ccc} & b_n & \\ L^{a-1}S & | & L^{a-2}S \\ \uparrow & & \uparrow \\ & b_k & \\ L & | & S \end{array}$$

By recurrence hypothesis, $\langle b_k \rangle_\beta$ ends in $(a-1)(a-2)^q$, with $q \in \mathbb{N}$. Then $\langle b_n \rangle_\beta$ ends in $(a-1)(a-2)^q 0$. Thus, the β-expansion of $b_{n+1}, b_{n+2}, \ldots, b_{n+a-3}$, which are LL, ends in $1, 2, \ldots, (a-3)$, respectively, and the β-expansion of b_{n+a-2}, which is LS, ends in $(a-1)(a-2)^{q+1} \in \mathcal{M}$.

In the third configuration, SL is a factor of $L^{a-2}SL^{a-1}S$, $b_n = \beta b_k$ with b_k a SL β-integer.

$$\begin{array}{ccc} & b_n & \\ L^{a-2}S & | & L^{a-1}S \\ \uparrow & & \uparrow \\ & b_k & \\ S & | & L \end{array}$$

By recurrence hypothesis $\langle b_k \rangle_\beta$ ends in 0, and $\langle b_n \rangle_\beta$ ends in 00. Therefore, the β-expansion of $b_{n+1}, b_{n+2}, \ldots, b_{n+a-2}$, which are LL, ends in $01, 02, \ldots, 0(a-2)$, and the β-expansion of b_{n+a-1}, which is LS, ends in $0(a-1)$.

The cases where b_n is an LL or an LS β-integer have been already treated. The case SS cannot occur.

Now let us prove $(i) \Leftrightarrow (iii)$. Recall that a number x of $\mathbb{Z}[\beta] \cap R^+$ is a positive β-integer if and only if its conjugate x' belongs to $(0, \beta)$. Let b_n be a SL β-integer. Then $b_{n-1} = b_n - (1 - 1/\beta)$ and $b_{n+1} = b_n + 1$ are β-integers, and $(b_{n-1})' = b'_n - 1 + \beta \in (0, \beta)$ and $(b_{n+1})' = b'_n + 1 \in (0, \beta)$. Therefore $b'_n \in (0, 1)$.

Let b_n be a LL β-integer. Then $b_{n-1} = b_n - 1$ and $b_{n+1} = b_n + 1$ are β-integers, and $(b_{n-1})' = b'_n - 1 \in (0, \beta)$ and $(b_{n+1})' = b'_n + 1 \in (0, \beta)$. Therefore $b'_n \in (1, \beta - 1)$. Since $b_1 = 1$ is LL, $b'_n \in [1, \beta - 1)$.

Finally, let b_n be a LS β-integer. Then $b_{n-1} = b_n - 1$ and $b_{n+1} = b_n + (1 - 1/\beta)$ are β-integers, and $(b_{n-1})' = b'_n - 1 \in (0, \beta)$ and $(b_{n+1})' = b'_n + 1 - \beta \in (0, \beta)$. Therefore $b'_n \in (\beta - 1, \beta)$. □

We now precise the characterization for LL β-integers.

Proposition 4 *Let b_n be a LL β-integer, then $\langle b_n \rangle_\beta$ ends in*

4.1 an $h \in \{1, \ldots, a-3\}$ if and only if $b'_n \in \Omega_{LL}(h) = [h, h+1)$

4.2 $(a-2)$ not prefixed by an element of $\mathcal{M}$ if and only if $b'_n \in \Omega_{LL}(a-2) = (a-2, \beta - 1)$.

Proof. Let us first prove *4.1*. Let b_n be a LL β-integer such that $\langle b_n \rangle_\beta$ ends in 1. Then $b_n - 1 = b_{n-1}$ is SL, and $b'_n - 1 \in (0, 1)$. Then $b'_n \in (1, 2)$. Since $b_1 = 1$ is LL, $b'_n \in [1, 2)$. From the fact that $h' = h$ for $h \in \{1, \ldots, (a-3)\}$, we deduce *4.1*.

The proof of *4.2* is now straightforward. The only possibility for β-integers such that their β-expansion ends in $(a-2)$ not prefixed by an element of $\mathcal{M}$ is $\Omega_{LL}(a-2) = (a-2, \beta - 1)$. □

3.2 Some Cubic Pisot Units

Case 1. The Tribonacci number: $\beta^3 = \beta^2 + \beta + 1$

The substitution σ_β allows only the following configurations (respectively Voronoi cells): LM, LS, ML, SL and LL in u_β.

Proposition 5 *In each of the following assertions (i) and (ii) are equivalent. For $n \geqslant 1$*

5.1 (i) b_n is LM; (ii) $\langle b_n \rangle_\beta$ ends in 01, or $n = 1$ and $\langle b_1 \rangle_\beta = 1$.

5.2 (i) b_n is LS; (ii) $\langle b_n \rangle_\beta$ ends in 011, or $n = 3$ and $\langle b_3 \rangle_\beta = 11$.

5.3 (i) b_n is ML; (ii) $\langle b_n \rangle_\beta$ ends in $10(000)^q$, $q \in \mathbb{N}$.

5.4 (i) b_n is SL; (ii) $\langle b_n \rangle_\beta$ ends in $100(000)^q$, $q \in \mathbb{N}$.

5.5 (i) b_n is LL; (ii) $\langle b_n \rangle_\beta$ ends in $1000(000)^q$, $q \in \mathbb{N}$.

Proof. Suppose that b_n is ML. The word ML can be issued from the substitution of three configurations of letters. In the first configuration, ML is a factor of $LMLSLM$, and $b_n = \beta b_k$, where b_k is LM.

$$\begin{array}{ccccccc} & & b_n & & b_{n+1} & b_{n+2} & b_{n+3} \\ LM & | & LS & & | & LM & \\ \uparrow & & \uparrow & & & \uparrow & \\ & & b_k & & b_{k+1} & & \\ L & | & M & & | & L & \end{array}$$

Since by recurrence hypothesis, $\langle b_k \rangle_\beta$ ends in 01, $\langle b_n \rangle_\beta$ ends with 010. Consequently the β-expansions of b_{n+1}, b_{n+2} and b_{n+3}, which are respectively LS, SL and LM, end with 011, 100 and 101, respectively.

In the second configuration ML is a factor of $LMLLM$, and $b_n = \beta b_k$, where b_k is LS.

$$\begin{array}{ccccccc} & & b_n & & b_{n+1} & b_{n+2} & \\ LM & | & L & & | & LM & \\ \uparrow & & \uparrow & & & \uparrow & \\ & & b_k & & b_{k+1} & & \\ L & | & S & & | & L & \end{array}$$

By recurrence hypothesis, $\langle b_k \rangle_\beta$ ends in 011, then $\langle b_n \rangle_\beta$ ends with 0110. Thus, the β-expansions of b_{n+1}, b_{n+2} which are respectively LL and LM end with 1000 and 1001, respectively. Since M is always follwed by L, b_{n+3} is ML, and ends in 1010.

Finally, in the third configuration ML is a factor of $LMLM$, and $b_n = \beta b_k$, where b_k is LL.

$$\begin{array}{ccccc} & & b_n & b_{n+1} & \\ LM & | & LM & & \\ \uparrow & & \uparrow & & \\ & & b_k & & \\ L & | & L & & \end{array}$$

By recurrence hypothesis, $\langle b_k \rangle_\beta$ ends in $1000(000)^q$, $q \in \mathbb{N}$, then $\langle b_n \rangle_\beta$ ends with $10(000)^{q+1}$. Then the β-expansions of b_{n+1} and b_{n+2}, which are respectively LM and ML, end with $10(000)^q001$ and $10(000)^q010$, respectively.

Let now b_n be SL, which appears as a factor of $LSLM$. Then $b_n = \beta b_k$, where b_k is ML.

$$\begin{array}{ccccc} & & b_n & b_{n+1} & \\ LS & | & LM & & \\ \uparrow & & \uparrow & & \\ & & b_k & & \\ M & | & L & & \end{array}$$

Since by recurrence hypothesis $\langle b_k \rangle_\beta$ ends with $10(000)^q$, then $\langle b_n \rangle_\beta$ ends with $100(000)^q$, and consequently, the β-expansions of b_{n+1} and b_{n+2}, which are LM and ML, respectively, end in 001 and 010, respectively.

Eventually, let b_n be LL, which appears as a factor of LLM. Then $b_n = \beta b_k$, with b_k SL.

$$\begin{array}{ccc} & b_n\ b_{n+1} & \\ L & | & LM \\ \uparrow & & \uparrow \\ & b_k & \\ S & | & L \end{array}$$

By recurrence hypothesis $\langle b_k \rangle_\beta$ ends in $100(000)^q$, then $\langle b_n \rangle_\beta$ ends with $1000(000)^q$, and consequently, the β-expansions of b_{n+1} and b_{n+2}, which are LM and ML respectively, end in 001 and 010. Cases LM and LS are treated above. □

Let α be one of the complex roots of $X^3 - X^2 - X - 1$. On Figure 3 we display the Rauzy fractal, *i.e.* the set $\Omega = \overline{(\mathbb{Z}_\beta^+)'}$, which is the closure of the set $\{\sum_{0 \leqslant i \leqslant N} x_i \alpha^i \mid \text{there is no factor 111 in } x_N \cdots x_0,\ N \geqslant 0\}$, and its partition according to Voronoi cells of β-integers. Recall that, by definition, $b_0 = 0$ is LL. The origin 0, although it belongs to Ω_{LL}, lies at the intersection of Ω_{LL} with Ω_{ML} and Ω_{SL}.

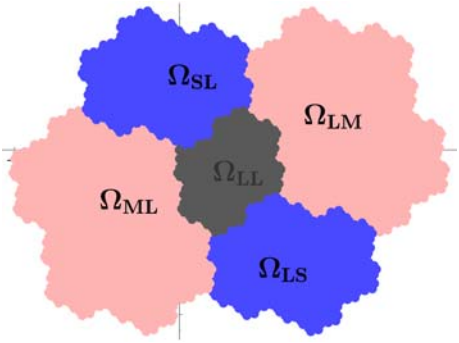


Fig. 3. The Rauzy fractal, obtained when β is the Tribonacci number

Usually the Rauzy fractal is divided into three basic tiles T_0, T_{01} and T_{011}, see [11, 16] or [15, Chapter 7]. Obviously our partition of the Rauzy fractal is a refinement of the classical division, with $T_0 = \Omega_{SL} \cup \Omega_{ML} \cup \Omega_{LL}$, $T_{01} = \Omega_{LM}$, and $T_{011} = \Omega_{LS}$.

Thus the domain exchange ρ defined on the Rauzy fractal (see Theorem 7.4.4 in [15]) is just the following:

$$T_0 = \Omega_{SL} \cup \Omega_{ML} \cup \Omega_{LL} \overset{\rho}{\mapsto} \Omega_{LS} \cup \Omega_{LM} \cup \Omega_{LL}$$
$$T_{01} = \Omega_{LM} \overset{\rho}{\mapsto} \Omega_{ML}$$
$$T_{011} = \Omega_{LS} \overset{\rho}{\mapsto} \Omega_{SL}.$$

From Proposition 5 one obtains the following result.

Proposition 6 *In the Rauzy fractal we have the following relations*

(i) $\Omega_{ML} = \Omega_{LM} + \alpha^{-1} + \alpha^{-2}$
(ii) $\Omega_{SL} = \Omega_{LS} + \alpha^{-1}$
(iii) $\Omega_{LL} = \alpha\Omega_{LS} + 1 = \alpha^2\Omega_{LM} + \alpha + 1$.

Proof (i). Let b'_n be in Ω_{ML}. Then the β-expansion of b_n is of the form $w10(000)^q$. Using that the word $10(000)^q$ has the same value in base β (or α) as the word $(011)^q01.11$, we obtain that b'_n belongs to $\Omega_{LM} + \alpha^{-1} + \alpha^{-2}$.

Conversely, let b'_p be in Ω_{LM}. Then the β-expansion of b_p is of the form $v01$. The word $v01.11$ has same value as $v10$. If $v10$ is already in normal form, it is an element of Ω_{ML}. If not, it is of the form $u(011)^k10$, with k maximal, and its normal form is $u10(000)^k$, thus the result follows.

(ii) The proof is similar.
(iii) It follows easily from (i) and (ii), and from the fact that $\Omega_{LL} = \alpha\Omega_{SL} = \alpha^2\Omega_{ML}$. □

Case 2. Symmetry of order 7: $\beta^3 = 2\beta^2 + \beta - 1$

The substitution σ_β allows only the following configurations (respectively Voronoi cells): LL, LS, SL, SM and ML.

Let $\mathcal{M}_1 = 2(01)^*$ and $\mathcal{M}_2 = 2(01)^*0$. The set $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$ is the set of maximal words of each length in the radix order, i.e., the set of prefixes of $d_\beta(1)$.

Proposition 7 *In each of the following assertions (i) and (ii) are equivalent. For* $n \geqslant 1$

6.1 (i) b_n *is* LL*; (ii)* $\langle b_n \rangle_\beta$ *ends in* $w1$ *where* $w \notin \mathcal{M}_2$.

6.2 (i) b_n *is* LS*; (ii)* $\langle b_n \rangle_\beta$ *ends in a word* $w \in \mathcal{M}_1$.

6.3 (i) b_n *is* SL*; (ii)* $\langle b_n \rangle_\beta$ *ends in* $w0$ *where* $w \notin \mathcal{M}_1$ *or by* $(0)^{2q+1}$, $q \in \mathbb{N}^*$.

6.4 (i) b_n *is* SM*; (ii)* $\langle b_n \rangle_\beta$ *ends in a word* $w \in \mathcal{M}_2$.

6.5 (i) b_n *is* ML*; (ii)* $\langle b_n \rangle_\beta$ *ends in* $(0)^{2q+2}$, $q \in \mathbb{N}$.

Proof. It is similar to the proof of Proposition 5. □

References

1. P. Arnoux et G. Rauzy, Représentation géométrique de suites de complexité $2n+1$. *Bull. Soc. Math. France* **119** (1991), 199–215.
2. A. Bertrand, Développements en base de Pisot et répartition modulo 1, *C. R. Acad. Sc. Paris*, Série A, t.285, (1977) 419–421.
3. Č. Burdik, Ch. Frougny, J. P. Gazeau, R. Krejcar, Beta-integers as natural counting systems for quasicrystal, *J. of Physics A: Math. Gen.* **31** (1998) 6449–6472.
4. A. Elkharrat, Etude et application des beta-réseaux aux structures apériodiques, Ph. D. Thesis, Université Paris 7, novembre 2004.
5. S. Fabre, Substitutions et β-systèmes de numération, *Theor. Comp. Sci.* **137** (1995) 219–236.

6. Ch. Frougny, J. P. Gazeau, R. Krejcar, Additive and multiplicative properties of point sets based on beta-integers, *Theor. Comp. Sci.* **303** (2003) 491-516.
7. Ch. Frougny, Z. Masáková, E. Pelantová, Complexity of infinite words associated with beta-expansions, *RAIRO-Inf. Theor. Appl.* **38** (2004) 163–185. Corrigendum, *RAIRO-Inf. Theor. Appl.* **38** (2004) 269–271.
8. J. C. Lagarias, Meyer's concept of quasicrystal and quasiregular sets, *Commun. Math. Phys.* **179** (1996) 365–376.
9. J. C. Lagarias, Geometric models for quasicrystals: I. Delone sets of finite type, *Discrete Comput. Geom.* **21** (1999) 161–191.
10. M. Lothaire, *Algebraic combinatorics on words*, Cambridge University Press (2002).
11. A. Messaoudi, Frontière du fractal de Rauzy et système de numération complexe, *Acta Arithmetica* **95** (2000) 195–224.
12. Y. Meyer, *Algebraic numbers and harmonic analysis*, North-Holland (1972).
13. Y. Meyer, Quasicrystals, Diophantine approximation and algebraic numbers, in *Beyond Quasicrystals*, F. Axel, D. Gratias (Eds), Les Editions de Physique, Springer (1995).
14. W. Parry, On the β-expansions of real numbers, *Acta Math. Acad. Sci. Hungar.* **11** (1960) 401–416.
15. N. Pytheas Fogg, *Substitutions in Dynamics, Arithmetics and Combinatorics*, Lecture Notes In Mathematics **1794**, Springer (2002).
16. G. Rauzy, Nombres algébriques et substitutions, *Bull. Soc. Math. France* **110** (1982) 147–178.
17. A. Rényi, Representations for real numbers and their ergodic properties, *Acta Math. Acad. Sci. Hung.* **8** (1957) 477–493.
18. K. Schmidt, On periodic expansions of Pisot numbers and Salem numbers, *Bull. London Math. Soc.* **12** (1980) 269–278.
19. W. Thurston, *Groups, tilings, and finite state automata.* AMS Colloquium Lecture Notes, Boulder, (1989).

Languages with Mismatches and an Application to Approximate Indexing*

Chiara Epifanio, Alessandra Gabriele, and Filippo Mignosi

Dipartimento di Matematica ed Applicazioni,
Università degli Studi di Palermo
Via Archirafi 34, 90123 Palermo, Italy
{epifanio,sandra,mignosi}@math.unipa.it

Abstract. In this paper we describe a factorial language, denoted by $L(S,k,r)$, that contains all words that occur in a string S up to k mismatches every r symbols. Then we give some combinatorial properties of a parameter, called *repetition index* and denoted by $R(S,k,r)$, defined as the smallest integer $h \geq 1$ such that all strings of this length occur at most in a unique position of the text S up to k mismatches every r symbols. We prove that $R(S,k,r)$ is a non-increasing function of r and a non-decreasing function of k and that the equation $r = R(S,k,r)$ admits a unique solution.
The repetition index plays an important role in the construction of an indexing data structure based on a *trie* that represents the set of all factors of $L(S,k,r)$ having length equal to $R(S,k,r)$. For each word $x \in L(S,k,r)$ this data structure allows us to find the list $occ(x)$ of all occurrences of the word x in a text S up to k mismatches every r symbols in time proportional to $|x| + |occ(x)|$.

Keywords: Combinatorics on words, formal languages, approximate string matching, indexing.

1 Introduction

In this paper we study some combinatorial properties of languages that represent words that occur in a text S with k mismatches every r symbols, k, r non negative integers. More precisely, given a text $S = a_1 a_2 \dots a_n$ and two non negative integers k and r, $k \leq r$, we say that a string u k_r-*occurs* with mismatches in S at position l, $1 \leq l \leq n$, if it occurs in S at position l with at most k mismatches in any window of size r. We call such a string u a k_r-*occurrence* in S at position l. The combinatorial study of the language, denoted by $L(S,k,r)$, of all the k_r-occurrences in a text S is important from several points of view. For instance it helps in the design and in the analysis of algorithms and data structures for approximate string matching, that have, in turn, plenty of applications such as the text searching in presence of typing and spelling errors and the finding of DNA subsequences after possible evolutionary mutations. Indeed, in this paper

* Partially supported by MIUR National Project PRIN "Linguaggi Formali e Automi: teoria ed applicazioni"

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 224–235, 2005.
© Springer-Verlag Berlin Heidelberg 2005

we also describe an index of k_r-occurrences in a text. Different definitions of approximate pattern matching can be found in [10], [11], [12].

The combinatorial study of the language $L(S,k,r)$ that we describe in this paper mainly focus on a parameter, called *repetition index* and denoted by $R(S,k,r)$, introduced for the first time in [4]. It is defined as the smallest integer $h \geq 1$ such that all strings of this length occur at most in a unique position of the text S up to k mismatches every r symbols.

Notions similar to the repetition index have been deeply studied. It belongs to folklore the fact that the size of the longest repeated factor in a string S is equal to the maximal string-depth (or height) of internal nodes in the suffix-tree of S. It represents a special case of the repetition index when the number k of mismatches is zero. In [1, 13] it is defined the *generalized height* $H(S,D)$ as the largest h for which there exist $1 \leq i < j \leq |S|-k+1$ such that $d(S(i,i+h-1), S(j,j+h-1)) \leq hD$. The real number D is the *proportion of mismatches*. In the same papers an average evaluation of this notion is given.

In [4] an evaluation of the repetition index is given. More precisely, if S is an infinite sequence generated by a memoryless source and S_n the sequences of S prefixes, n being the length of S_n, the repetition index $R(S_n,k,r)$ is logarithmically upper bounded by the size of S_n almost surely.

In this paper we prove several combinatorial properties of $R(S,k,r)$. It is a non-increasing function of the window size r, a non-decreasing function of the number k of errors and, under the hypothesis that k is fixed and $r \geq R(S,k,r)$, it gets constant. Moreover, there exists a unique solution of the equation $r = R(S,k,r)$.

As an application, we define an approximate indexing data structure in which the repetition index plays an important role.

Literature on approximate indexing involves many results. Among the most recent ones, in [2] Cole, Gottlieb and Lewenstein propose a data structure for Hamming distance that answer queries in time $O(\frac{(\log|S|)^k \cdot \log\log|S|}{k!} + |x| + |occ(x)|)$, where x is the word we are looking for, $occ(x)$ is the list of all its occurrences, k is the maximal number of allowed mismatches and $k \leq \log|S|$. In [7], Huynh, Hon, Lam and Sung describe two indexing data structures that take $O(|S| \cdot \log|S|)$ bits and $O(|S|)$ bits, respectively. In the special case $k = 1$, the query time is $O(|x| \cdot \log|S| + |occ(x)|)$ in the first data structure and $O(|x| \cdot \log^2|S| + |occ(x)| \cdot \log|S|)$, in the second one. In the more general case $k > 1$, the query time is $O(|x|^k \cdot \log|S| + |occ(x)|)$ in the first data structure and $O(|x|^k \cdot \log^2|S| + |occ(x)| \cdot \log|S|)$ in the second one. In [4] another indexing data structure is defined in the special case $r = R(S,k,r)$. For each word $x \in L(S,k,r)$ this data structure finds the list of all k_r-occurrences of x in S in time proportional to $|x| + |occ(x)|$, but under an additional hypothesis on the distribution of the repetition index. The average size of the data structure for k fixed is $O(|S| \cdot \log^{k+1}|S|)$.

In this paper we give an alternative method to build up an indexing data structure that is simpler than the one given in [4], has the same average size and has the same optimal query time without any additional hypothesis. More

precisely, we define a new indexing data structure based on a *trie* representing all factors of $L(S,k,r)$ of length equal to $R(S,k,r)$. For each word x this data structure allows us to find the list of all k_r-occurrences of x in S in time proportional to $|x| + |occ(x)|$ but, differently from the search in [4], without additional hypothesis.

The remainder of this paper is organized as follows. In the second section we give some basic definitions we will use in the following. In the third and fourth sections we define the language $L(S,k,r)$ of the k_r-occurrences of a text S and the repetition index $R(S,k,r)$ and we prove some combinatorial properties of them. Finally, last section is devoted to the construction of the *trie* based indexing data structure and to the algorithm for searching the k_r-occurrences of a word in a text represented through this data structure.

2 Basic Definitions

Let Σ be a finite set of symbols, usually called the *alphabet*. We denote by Σ^* the set of words over Σ and by ϵ the empty word. For a word w, we denote by $|w|$ the length of w.

A word $u \in \Sigma^*$ is a *factor* (or *substring*)(*prefix*, *suffix* resp.) of a word w if there exist words $x,\ y\ \in \Sigma^*$ such that $w = xuy$ ($w = uy$, $w = xu$ resp.). The factor (prefix, suffix, resp.) is *proper* if $xy \neq \epsilon$ ($y \neq \epsilon$, $x \neq \epsilon$, resp.). We denote the set of factors (prefixes, suffixes, resp.) of a language L by $Fact(L)$ (resp. $Pref(L)$, resp. $Suff(L)$).

We denote an occurrence of a factor in a string $w = a_1 a_2 \dots a_n$ at position i, $1 \leq i \leq n$, by $w(i,j) = a_i \dots a_j$, for some j such that $i \leq j \leq n$. The length of a substring $w(i,j)$ is the number of the letters that compose it, i.e. $j - i + 1$. Given a set I of words, we also denote by $\|I\|$ the sum of the lengths of all words in the set I.

A *factorial language* (cf. [8]) $L \subseteq \Sigma^*$ is a language that contains all factors of its words, i.e. $\forall\ u,\ v \in \Sigma^*,\ uv \in L \Rightarrow u,\ v \in L$. The complement language $L^c = \Sigma^* \setminus L$ is a (two-sided) ideal of Σ^*. Let us denote by $MF(L)$ the base of this ideal. We have that $L^c = \Sigma^* MF(L) \Sigma^*$. The set $MF(L)$ is called the set of *minimal forbidden words* for L. By definition, a word $v \in \Sigma^*$ is *forbidden* for the factorial language L if $v \notin L$, which is equivalent to say that v occurs in no word of L. In addition, v is minimal if it has no proper factor that is forbidden.

Let w be a finite word on the alphabet Σ. It is possible to define the set of minimal forbidden factors of a single word w as the set of minimal forbidden factors of the language $Fact(w)$ and we denote it simply by $MF(w)$. For example, if we consider the word $w = atgcta$ over the alphabet $\Sigma = \{a, c, g, t\}$, One has that $MF(w) = \{aa, ac, ag, ata, ca, cc, cg, ctg, ga, gg, gt, tat, tc, tt\}$.

In order to introduce the problem of *approximate string matching*, we need a notion of distance between words.

Let us consider the function $d : \Sigma^* \times \Sigma^* \to \mathbb{R}$ defined between two strings x and y on the alphabet Σ as the minimal cost of a sequence of *operations* that transform x into y (and ∞ if no such sequence exists). The cost of a sequence of operations is the sum of the costs of the individual operations. The operations

are a finite set of rules of the form (x, y), where x and y are *different* strings. Their cost is $\delta(x, y) = t$, t being a real number.

In most applications the possible operations are:

Insertion i.e. inserting the letter a, with cost $\delta(\epsilon, a)$;
Deletion i.e. deleting the letter a, with cost $\delta(a, \epsilon)$;
Substitution or Replacement i.e. substituting a by b for $a \neq b$, with cost $\delta(a, b)$;
Transposition i.e. swap the adjacent letters a and b for $a \neq b$, with cost $\delta(ab, ba)$.

There are different distances between words that are defined starting from above operations and that allow to transform a word x into another word y. The most commonly used distance functions are the following:

- *Levenshtein or edit distance* [Levenshtein, 1965]: it allows insertions, deletions and substitutions. In the simplified definition all the operations cost 1. This can be rephrased as "the minimal number of insertions, deletions and substitutions to make two strings equal". In the literature the search problem in many cases is called "string matching with k differences". It also holds $0 \leq d(x, y) \leq \max\{|x|, |y|\}$;
- *Hamming distance* [Sankoff and Kruskal, 1983]: it allows only substitutions, which cost 1 in the simplified definition. In the literature the search problem in many cases is called "string matching with k mismatches". It is finite whenever $|x| = |y|$. In this case it holds $0 \leq d(x, y) \leq |x|$.

In this paper we take in consideration the *Hamming distance* d in the simplified definition. In this case, given two strings x and y having the same length, $d(x, y)$ is the minimal number of character substitutions that transform x into y. For example, if we consider the strings $x, y \in \Sigma^*$, $x = acgactccga$ and $y = ccgacttcgt$, one has that the Hamming distance between the stings x and y is $d(x, y) = 3$.

3 The Language $L(S, k, r)$

In the field of approximate string matching, typical approaches for finding a string in a text consist in considering a percentage D of errors, or fixing the number k of them. In the classical notion of approximate string matching we introduce a new parameter r and allow at most k errors for any factor of length r of the text. We have the following definition.

Definition 1. *Let S be a string over the alphabet Σ, and let k, r be non negative integers such that $k \leq r$. A string u* occurs in S at position l up to k errors in a window of size r *or, simply,* k_r-occurs in S at position l, *if one of the following two conditions holds:*

- *if $|u| < r$ then $d(u, S(l, l + |u| - 1)) \leq k$;*
- *if $|u| \geq r$ then $\forall\ \ i, \ \ 1 \leq i \leq |u| - r + 1, \ \ d(u(i, i + r - 1), S(l + i - 1, l + i + r - 2)) \leq k$.*

A string u satisfying the above property is a k_r-occurrence *of S.*

From now on we will suppose that the text is non-empty, $r \geq 2$ and $0 \leq k \leq r$, otherwise the above definition would have no meaning.

Remark 1. If we wish to have no window at all, we can set the size of the window r to be equal to $|S|$. Indeed, when the size r of the window is equal to the size of the text S, then the problem of finding all k_r-occurrences of a string u in the text is equivalent to the k-mismatch problem, that consists in finding all occurrences of the string u in S with at most k errors (cf. [6]).

We denote by $L(S,k,r)$ the set of words u that k_r-occur in S at position l, for some l, $1 \leq l \leq |S| - |u| + 1$. Notice that $L(S,k,r)$ is a *factorial language*, i.e. if $u \in L(S,k,r)$ then each factor (or substring) of u belongs to $L(S,k,r)$. Clearly if u is a factor of S then $L(u,k,r) \subseteq L(S,k,r)$.

Example 1. Let $S = abaa$ be a string on the alphabet $\Sigma = \{a,b\}$. The set $L(S,1,2)$ of words that k_r-occur in S, when $k = 1$ and $r = 2$, is $L(S,1,2) = \{\ a,\ b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, aaaa, aaab, abaa, abab, abba, bbaa, bbab, bbba\}$.

Notice that words $aab, aaab, bbab, bbba$ occur with one error every $r = 2$ symbols, but with two errors in the whole word. Hence, they belong to $L(S,1,2)$, but not to $L(S,1,4)$.

Definition 2. *Let u be a string over the alphabet Σ. The* neighborhood *of u (with respect to k, r) is the set $V(u,k,r) = L(u,k,r) \cap \Sigma^{|u|}$.*

Remark 2. If a word u has length m then it is easy to see that the maximal distance between a word $v \in V(u,k,r)$ and u is bounded by $k\lceil \frac{m}{r} \rceil$. More precisely, if u has length $m = rq + p$, with $q = \lfloor \frac{m}{r} \rfloor$ and $p < r$, then the maximal distance between a word $v \in V(u,k,r)$ and u is $kq + \min\{k,p\}$. Therefore, a simple combinatorial argument shows that the number of elements in the set $V(u,k,r)$ is at most $\sum_{i=0}^{kq+\min\{k,p\}} \binom{m}{i} (|\Sigma| - 1)^i$.

The elements of $V(S,k,r)$ are the strings u such that $|u| = |S|$ and the k_r-occurrences of the factors of length r starting at the same position i ($1 \leq i \leq |S| - r + 1$) in S and u respectively have distance less than or equal to k, that is $\forall\ i,\ 1 \leq i \leq |S| - r + 1,\ d(S(i, i + r - 1), u(i, i + r - 1)) \leq k$.
The following equality holds between the language $L(S,k,r)$ and the set of all factors of $V(S,k,r)$.

Proposition 1. $L(S,k,r) = Fact(V(S,k,r))$.

Proof. The inclusion $Fact(V(S,k,r)) \subseteq L(S,k,r)$ follows from the definition of $V(S,k,r)$ and the factoriality of $L(S,k,r)$.

Conversely, if u belongs to $L(S,k,r)$ then u k_r-occurs in S at some position l, $1 \leq l \leq |S| - |u| + 1$, and then, by definition, u is such that one of the following two conditions holds. If $|u| < r$, $d(u, S(l, l + |u| - 1)) \leq k$; if $|u| \geq r$,

$\forall\ i,\ 1 \le i \le |u| - r + 1,\ d(u(i, i+r-1), S(l+i-1, l+i+r-2)) \le k$. In both cases let us factorize S in the following way: $S = h_1 S(l, l+|u|-1) h_2$ and hence $h_1 u h_2 \in V(S, k, r)$. □

Let I be a set of factors of the string S, i.e. $I \subseteq Fact(S)$. We denote by $I_{k,r}$ the union of the neighborhoods of the elements of I, i.e.: $I_{k,r} = \bigcup_{u \in I} V(u, k, r)$.

In next proposition we estimate the size $\|I_{k,r}\|$ of $I_{k,r}$ when I is the set of factors of S of length r.

Proposition 2. *Let $\mathcal{S}$ be an infinite sequence over a fixed alphabet Σ, S_n the prefix of $\mathcal{S}$ of length n, k the number of errors allowed in a window of size r and I the set of all factors of S_n having length r. If k is fixed, and $r \to \infty$ when $n \to \infty$, the size of the set $I_{k,r}$ is $\|I_{k,r}\| = O(n \cdot r^{k+1})$.*

Proof. By Remark 2 the number of elements in the set $V(u, k, r)$ when $|u| = r$ is at most $\sum_{i=0}^{k} \binom{r}{i} (|\Sigma|-1)^i$. Hence, the total length of all words in $V(u, k, r)$ is $O(r \cdot \sum_{i=0}^{k} \binom{r}{i} (\Sigma - 1)^i) = O(r^{k+1})$. Since the number of factors of S_n having size r is obviously bounded by n, the thesis follows.

4 The Repetition Index $R(S, k, r)$

In this section we introduce a parameter, called *repetition index* and denoted by $R(S, k, r)$, that plays an important role in the construction of an indexing data structure based on a trie that represents the set of all factors of a string S having length $R(S, k, r)$.

Definition 3. *The* repetition index *of S, denoted by $R(S, k, r)$, is the smallest integer h such that all strings of this length k_r-occur at most in a unique position of the text S, i.e.*

$$R(S, k, r) = \min\{h \ge 1 \quad | \ \forall\ i,\ j,\ \ 1 \le i, j \le |S| - h + 1,$$
$$V(S(i, i+h-1), k, r) \cap V(S(j, j+h-1), k, r) \ne \emptyset \Rightarrow i = j\}.$$

Remark 3. $R(S, k, r)$ is well defined because the integer $h = |S|$ is an element of the set above described.

Lemma 1. *If $\frac{k}{r} \ge \frac{1}{2}$ then $R(S, k, r) = |S|$.*

Proof. For any length $R < |S|$ pick two factors of S, $S(l_1, l_1 + R - 1)$ and $S(l_2, l_2 + R - 1)$ with $l_1 \ne l_2$. Define $v = a_1 \dots a_R$, a_i letters, such that in every window of v of size r, half of the letters are letters of $S(l_1, l_1 + R - 1)$ and the remaining are of $S(l_2, l_2 + R - 1)$. More formally, we let $a_i = S_{l_1+i-1}$ if $(i \pmod r) < \frac{r}{2}$ and $a_i S_{l_2+i-1}$ otherwise.

Under the hypothesis that $\frac{k}{r} \ge \frac{1}{2}$, it follows that the Hamming distance between v and $S(l_1, l_1+R-1)$ and between v and $S(l_2, l_2+R-1)$ is smaller than or equal to k in every window of size r. Therefore for any $R < |S|$, $v = a_1 \dots a_R$ k_r-occurs at least twice in S. Hence $R(S, k, r) = |S|$. □

Example 2. Let us consider the string $S = abcdefghijklmnoabzdezghzjkzmnz$. We want to determine the repetition index $R(S, k, r)$, with $k = 1$, by considering different sizes of the window in which errors are allowed.

- If $r = 2$, then $\frac{k}{r} = \frac{1}{2}$ and by Lemma 1 the repetition index $R(S, 1, 2) = |S| = 30$. A word w of length $R(S, 1, 2) - 1 = 29$ that appears in position 1 and in position 2 up to 1 error every 2 symbols is $wacceeggiikkmmoobbddzzhhjjz znn$.
- If $3 \leq r \leq 6$ then $R(S, 1, r) = \frac{|S|}{2} + 1 = 16$ and a word w of length $R(S, 1, r) - 1 = \frac{|S|}{2} = 15$ that appears twice up to 1 error every r symbols is $wabcdezghijkzmno$.
- If $r \geq 7$ then $R(S, 1, r) = 9$ and a word w of length $R(S, 1, r) - 1 = 8$ that appears twice up to 1 error every r symbols is $w = abzdefgh$.

The following evaluation of the repetition index, that we need in the following to evaluate the size of an indexing data structure, is given in [5].

Theorem 1. *Let $\mathcal{S}$ be an infinite sequence generated by a memoryless source, let S_n be the prefix of $\mathcal{S}$ of length n and p be the probability that the letters in two distinct positions are equals.*

1. *If k and r are fixed, almost surely*
   - $\limsup_{n\in\mathbb{N}} \frac{R(S_n,k,r)}{\log(n)} \leq \frac{2}{\mathcal{H}(D,p)}$, *when* $D = \frac{2k}{r} < 1 - p$,
   - $\liminf_{n\in\mathbb{N}} \frac{R(S_n,k,r)}{\log(n)} \geq \frac{2}{\mathcal{H}(D,p)}$, *when* $D = \frac{k}{r} < 1 - p$,

   where $\mathcal{H}(D, p) = (1-D)\log\frac{1-D}{p} + D\log\frac{D}{1-p}$ *is the classical entropy function.*
2. *If k is fixed and r is a function $r(n)$ of n such that* $\lim_{n\to\infty} r(n) = +\infty$ *almost surely* $\lim_{n\to\infty} \frac{R(S_n,k,r(n))}{\log n} = \frac{2}{\mathcal{H}(0,p)}$.

In [4] it is described an algorithm for finding the repetition index $R(S, k, r)$ of a text S up to mismatches having worst case running time proportional to the size $\|I_{k,r}\|$ of the set $I_{k,r}$ times $\log_2 R(S, k, r)$, where I is the set of all factors of S of length $2R(S, k, r)$, i.e. $O(\|I_{k,r}\| \cdot \log_2 R(S, k, r))$.

4.1 Some Combinatorial Properties of the Repetition Index

In this section we present some combinatorial properties of the repetition index.

First of all, we prove that the repetition index $R(S, k, r)$ is a non-increasing function of parameter r and a non-decreasing function of k.

Lemma 2. *If k and S are fixed and $r_1 \leq r_2$ then $R(S, k, r_1) \geq R(S, k, r_2)$.*

Proof. If a word v of length $|v|$ appears in two different positions in S up to k errors every r_2 symbols then it also appears in the same positions up to k errors every r_1 symbols. Since $R(S, k, r_2) - 1$ is the maximal length of such words then $R(S, k, r_1) - 1 \geq R(S, k, r_2) - 1$. □

Lemma 3. *If r and S are fixed and $k_1 \leq k_2$ then $R(S, k_1, r) \leq R(S, k_2, r)$.*

We omit the proof of previous statement because it is similar to the proof of Lemma 2 and it can be handled symmetrically.

The following lemma shows that under the hypothesis that k is fixed and $r \geq R(S,k,r)$, the repetition index gets constant.

Lemma 4. *If k and S are fixed, $r_1 \leq r_2$ and $r_1 \geq R(S,k,r_1)$ then $R(S,k,r_1) = R(S,k,r_2)$.*

Proof. Since by Lemma 2 $R(S,k,r_1) \geq R(S,k,r_2)$, one has that $r_2 \geq r_1 \geq R(S,k,r_1) \geq R(S,k,r_2)$.

By definition, any word v of length equal to $R(S,k,r_2)$ appears in S at most in a unique position up to k errors in a window of size r_2. Since $r_2 \geq R(S,k,r_2)$ every word v of length equal to $R(S,k,r_2)$ appears in S at most in a unique position up to k errors. As $r_1 \geq R(S,k,r_2)$ we can say that v appears at most in a unique position in S up to k errors every r_1 symbols. This fact implies that $R(S,k,r_2) \geq R(S,k,r_1)$ and the lemma is proved. □

Lemma 5. *If k and S are fixed and $r > R(S,k,r)$ then $r-1 \geq R(S,k,r-1)$.*

Proof. We claim that $R(S,k,r) = R(S,k,r-1)$. By Lemma 2, we have that $R(S,k,r) \leq R(S,k,r-1)$, as $r > r-1$. By definition, any word v of length equal to $R(S,k,r)$ appears in S at most in a unique position up to k errors in a window of size r. Since $r > R(S,k,r)$ every word v of length equal to $R(S,k,r)$ appears in S at most in a unique position up to k errors. As $r-1 \geq R(S,k,r)$ we can say that v appears at most in a unique position in S up to k errors every $r-1$ symbols. This fact implies that $R(S,k,r-1) \leq R(S,k,r)$, and the claim is proved. Since $r > R(S,k,r)$ then $r-1 \geq R(S,k,r)R(S,k,r-1)$ and the thesis follows. □

By using Lemma 2 and Lemma 5 we can prove the following theorem.

Theorem 2. *If k and S are fixed then there exists only one solution to the equation $r = R(S,k,r)$.*

Proof. Let us first prove that there exists at least one solution to the equation $r = R(S,k,r)$. The idea of the proof is to start with a large value of r, so that $r \geq R(S,k,r)$ and then to decrease r by one unit until the equality is reached.

Let us start with $r = |S|$. Since no repeated factor of S can have length greater than $|S|-1$, we have that $r \geq R(S,k,r)$. If $r = R(S,k,r)$ the statement is trivial. So we only have to consider the case when $r > R(S,k,r)$. If $r > R(S,k,r)$, by Lemma 5 we have that $r-1 \geq R(S,k,r-1)$. By iterating this argument, there will exist a value $h < r$ such that $r-hR(S,k,r-h)$, and the *existence* is proved.

The *uniqueness* follows by Lemma 2, since the repetition index $R(S,k,r)$ is a non-increasing monotonic function of the window size r. □

Proposition 3. *Let $\mathcal{S}$ be an infinite sequence over a fixed alphabet, S_n the prefix of $\mathcal{S}$ of length n, k the number of errors allowed in a window of size r and I the set of all factors of S_n having length r. If k is fixed, and r_n is a function of n that for any $n \geq 1$ satisfies the equations $r_n = R(S_n,k,r_n)$, we have that $r_n \to \infty$ when $n \to \infty$.*

Proof. If $n_1 > n_2$, since S_{n_2} is a prefix of S_{n_1}, then $r_{n_1} = R(S_{n_1}, k, r_{n_1}) \geq r_{n_2} = R(S_{n_2}, k, r_{n_2})$. Therefore the sequence r_n is monotone non decreasing. Suppose, by contradiction, that it does not converge to ∞, i.e., that there exists a constant $\hat{r}$ such that, for any n, $r_n \leq \hat{r}$. Since the alphabet is fixed, the number of words of length $\hat{r}$ is also finite. Let $\hat{N}$ be this number. If $n > \hat{N} + \hat{r}$ then the number of positions of S_n that represent an occurrence of a factor of S_n of length $\hat{r}$ is greater than $\hat{N}$. By the pigeon-hole principle at least two must be equal, contradicting the fact that $r_n = R(S_n, k, r_n) \leq \hat{r}$.

5 A Trie Based Approach for Approximate Indexing Data Structures

In this section, we give an algorithm for building an indexing data structure using a *trie* that represents the set of all possible strings having length $R(S, k, r)$ that k_r-occur in S.

First of all we find the repetition index $R(S, k, r)$ of the string S by using an algorithm described in [4] called Find-Repetition-Index, and then we keep the set I of all factors of S having length $R(S, k, r)$.

In the second step, given the set I of factors u of S of length $R(S, k, r)$, we build the set $I_{k,r}$ that is the union of neighborhoods of elements of I, i.e. $I_{k,r} = \bigcup_{u \in I} V(u, k, r)$. To do this we use a variation of the routine for generating the *frequentable neighbours* of factors of the text having a fixed length ([3, Section 7.6]) and its running time is proportional to $\|I_{k,r}\|$, where with the notation $\|I_{k,r}\|$ we denote the sum of the lengths of all strings in the set $I_{k,r}$.

The third step is to build the trie $\mathcal{T}(I, k, r)$ of the set $I_{k,r}$. Each leaf of this trie represents a k_r-occurrence of the string S having length $R(S, k, r)$. We add to any leaf of the trie $\mathcal{T}(I, k, r)$ an integer such that for any leaf i, the concatenation of the edge-labels on the path from the root to leaf i is the k_r-occurrence of S starting at position i. We note that, given a factor u of S, this number is the same for every word v of the neighborhood $V(u, k, r)$ of u.

For any leaf i we also add a position j of an array, denoted by Pos, such that $Pos[j] = i$ is the starting position of the k_r-occurrence of S represented by the j-th leaf in the lexicographic order, supposed a fixed order among the letters. To build array Pos and to add these positions on the leaves it is sufficient to make a linear time visit of the trie in the lexicographic order.

During this visit we can also associate to any internal node x of the trie $\mathcal{T}(S, k, r)$ two integers j_r and j_l such that $Pos[j_r] = r$ and $Pos[j_l] = l$, where r and l are the first and the last leaf in the lexicographic order of the subtrie of $\mathcal{T}(I, k, r)$ having root x.

Next proposition gives an evaluation of the size of the trie $\mathcal{T}(I, k, r)$.

Proposition 4. *Let $\mathcal{S}$ be an infinite sequence generated by a memoryless source, S_n the sequences of prefixes of $\mathcal{S}$ of length n and I the set of all factors of S_n having length $R(S_n, k, r)$.*

If k is fixed and $r = R(S_n, k, r)$, there exists a fixed constant C, that does not depend on the string $\mathcal{S}$, and an integer n_0 (that can depend on the string

$\mathcal{S}$) such that for every $n > n_0$ the size of the trie $\mathcal{T}(I,k,r)$ is upper bounded by $C \cdot n \cdot \log^{k+1}(n)$ almost surely.

Proof. The size of the trie $\mathcal{T}(I,k,r)$ is bounded by $\|I_{k,r}\|$, where with the notation $\|I_{k,r}\|$ we denote the sum of the lengths of all strings in the set $I_{k,r}$. Under the hypothesis that k is fixed and $r = R(S_n,k,r)$, by Proposition 3 and Proposition 2 the size of $I_{k,r}$ is $O(n \cdot R(S_n,k,r)^{k+1})$. Under the same hypothesis, from Theorem 1 we can deduce that almost surely $\lim_{n\to\infty} \frac{R(S_n,k,r)}{\log n} = \frac{2}{\mathcal{H}(0,p)}$, where p is the probability that the letters in two distinct positions are equals. Hence, almost surely $\forall\, \epsilon > 0,\ \exists\, n_0$ such that $\forall\, n > n_0 \left| \frac{R(S_n,k,r)}{\log(n)} - \frac{2}{\mathcal{H}(0,p)} \right| < \epsilon$. Then there exists a fixed constant C', that does not depend on $\mathcal{S}$ but only on the entropy $\mathcal{H}(0,p)$ of the source, such that almost surely $R(S_n,k,r(n)) \leq C' \log(n)$, and the thesis follows. □

5.1 Searching in the Indexing Data Structure

Once $\mathcal{T}(I,k,r)$ is equipped in this way, it can be used as indexing data structure for solving the problem of finding all k_r-occurrences of a string x in the text S.

By using a Read-Trie Function, that is a slight variation of Desc-Lente ([3, Section 5.1]), we "read", as long as possible, the string x in the trie $\mathcal{T}(I,k,r)$ and we output the last visited node q.

Now we can distinguish the following three cases.

i) $|x| = R(S,k,r)$. In this case the length of the string x is equal to the repetition index $R(S,k,r)$. Thus, the list of all k_r-occurrences of x has at most one element. If the end of x is reached, the output is exactly the number associated with the leaf i that represents the unique k_r-occurrence of x in S.

ii) $|x| > R(S,k,r)$. In this case the length of the string x is greater than the repetition index $R(S,k,r)$. Thus, the list of all k_r-occurrences of x has at most one element, that could be a *false positive* for the original problem (i.e. the pattern x has more than k errors in every window of size r). So, when we reach the leaf that represents the prefix of x of length $R(S,k,r)$ in the trie $\mathcal{T}(I,k,r)$, we obtain a unique position i. By using a Check-Errors Function we have only to check if x k_r-occurs in S in position i. In case of positive answer, that is if the output of the Check-Errors Function is *true*, the algorithm returns i. Check-Errors Function runs in time $O(|x|)$.

iii) $|x| < R(S,k,r)$. In this case the length of the string x is smaller than the repetition index. Therefore the list of all k_r-occurrences of the string x can have more than one element. Hence, if the end of x is reached, we consider the indexes j_r and j_s in the array Pos of the first and last leaf in the lexicographic order of the subtrie of $\mathcal{T}(I,k,r)$ having root x. Let us consider the array Pos from index j_r to index j_l. This sub-array contains all the starting positions of the k_r-occurrences of x in S represented by the leaves of the subtrie having root x. We note that these positions can be repeated. Indeed, suppose that x occurs without mismatches in position l. Then all words that have x as a prefix, have length $R(S,k,r)$ and k_r-occur in position l, belong to the subtrie

of $\mathcal{T}(I,k,r)$ having root x. The number h of these words cannot be bounded as $R(S,k,r)-|x|$ is unbounded too. Therefore, position l appears at least h times in the array Pos between j_r and j_s. But we want to add the number l to the list $occ(x)$ just once, and, moreover we want to do this in constant time in order to achieve the optimal output time.

To solve the problem of finding the different positions $occ(x)$ of all k_r-occurrences of x in S in time $O(|output|)$, where $|output|$ is the size of the list $occ(x)$ of all different positions of all k_r-occurrences of x in S, we use the solution of the *Colored range query problem* presented in [9]. The statement of this problem is the following. Given an array $A[1..n]$ of colors to be preprocessed, online query consists of an interval $[r,l]$, and the goal is to return the set of distinct colors that appears in $A[r..l]$. This function, called Colored-Range-Query, has as input the sub-array $Pos[j_r,j_l]$ and returns the list of different positions $occ(x)$ of all k_r-occurrences of x in S, in time proportional to $|occ(x)|$. The description of this function is given in [9].

Before giving the formal description of this algorithm, we state the following proposition on its searching time which proof is straightforward.

Proposition 5. *The overall running time of* Search-in-Indexing-Structure *Algorithm is proportional to* $|x|+|occ(x)|$.

```
Search-in-Indexing-Structure(T(I,k,r),x,S)
1.  q ← Read-Trie(x);
2.  if q is a leaf and |x| = R(S,k,r)
3.      then return i
4.  else if q is a leaf and |x| > R(S,k,r)
5.              then if Check-Errors(x,i,S,k,r)
6.                      then return i
7.                   else return ("x is a false positive")
8.       else occ(x) ← Colored-Range-Query(Pos[j_r, j_l]);
9.              return occ(x).
```

Analogously as proved in [4], we can use the data structure built for $r=R(S,k,r)$ to search for all occurrences of any word x up to k mismatches without window (or equivalently for $r=|S|$), i.e., to settle the k-mismatch problem, as noticed in Remark 1. Indeed, it is sufficient to modify previous algorithm by using Check-Errors$(x,i,S,k,|S|)$ instead of Check-Errors(x,i,S,k,r).

This fact, together with Proposition 4 and Proposition 5, gives the following result.

Theorem 3. *The k-mismatch problem on text S over a fixed alphabet can be settled by a data structure having average size $O(|S|\cdot\log^{k+1}|S|)$ that answers queries in time $O(|x|+|occ(x)|)$ for any query word x.*

References

1. R. Arratia and M. Waterman. The Erdös-Rényi strong law for pattern matching with given proportion of mismatches. *Annals of Probability*, 4:200–225, 1989.
2. R. Cole, L. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of Annual ACM Symposium on Theory of Computing (STOC 2004)*, 2004.
3. M. Crochemore, C. Hancart, and T. Lecroq. *Algorithmique du texte*. Vuibert, 2001. 347 pages.
4. A. Gabriele, F. Mignosi, A. Restivo, and M. Sciortino. Indexing structure for approximate string matching. In *Proc. of CIAC'03*, volume 2653 of *LNCS*, pages 140–151, 2003.
5. A. Gabriele, F. Mignosi, A. Restivo, and M. Sciortino. Approximate string matching: indexing and the k-mismatch problem. Technical Report 244, University of Palermo, Department of Mathematics and Applications, 2004.
6. D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997. ISBN 0 521 58519 8 hardback. 534 pages.
7. T. N. D. Huynh, W. K. Hon, T. W. Lam, and W. K. Sung. Approximate string matching using compressed suffix arrays. In *Proc. of CPM 2004*, volume 3109 of *LNCS*, pages 434–444.
8. M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002. ISBN 0-521-81220-8 hardback.
9. S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 657–666, 2002.
10. J. Pelfrêne, S. Abdeddaïm, and J. Alexandre. Extracting approximate patterns. In *Proceedings of CPM 2003*, pages 328–347, 2003.
11. N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot. A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum. In *Proceedings of MFCS 2003*, pages 622–631, 2003.
12. N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot. A comparative study of bases for motif inference. In *String Algorithmics, C. Iliopoulos and T.Lecroq editors*. King's College London Publications, 2004.
13. W. Szpankowski. *Average Case Analysis of Algorithms on Sequences*. John Wiley & Sons, 2001.

Bidimensional Sturmian Sequences and Substitutions

Thomas Fernique

[1] LIRMM CNRS-UMR 5506 and Université Montpellier II,
161 rue Ada, 34392 Montpellier Cedex 5, France
thomas.fernique@ens-lyon.org
[2] PONCELET Lab. CNRS-UMI 2615 and Independent University of Moscow,
Bol'shoj Vlas'evskij per 11, 119002 Moscow, Russia

Abstract. Substitutions are powerful tools to study combinatorial properties of sequences. There exist strong characterizations through substitutions of the Sturmian sequences that are S-adic, substitutive or a fixed-point of a substitution. In this paper, we define a bidimensional version of Sturmian sequences and look for analogous characterizations. We prove in particular that a bidimensional Sturmian sequence is always S-adic and give sufficient conditions under which it is either substitutive or a fixed-point of a substitution.

Introduction

Substitutions are non-erasing morphisms of the free monoid $\mathcal{A}^*$ and generate infinite sequences by iteration, replacing a letter of $\mathcal{A}$ by a word of $\mathcal{A}^*$. One of the most interesting property of sequences obtained in this way is that they are algorithmically easily generated and have a strongly ordered structure, though not restricted to the single periodic case.

The connection between substitutions and *Sturmian* sequences has been widely studied. Roughly speaking, a Sturmian sequence $\mathcal{S}_{\alpha,\rho}$ over the alphabet $\{1,2\}$ encodes the way the line $y = \alpha x + \rho$, α being irrational, crosses the unit squares of the lattice $\mathbb{Z}^2$ (see Fig. 1 and for more details [9, 11]).

A Sturmian sequence is said *substitutive*, according to the terminology of [7], if it is the image under a morphism of a fixed-point of a (nontrivial) substitution:

$$\mathcal{S}_{\alpha,\rho} = \tau(\mathcal{S}') \quad \textit{and} \quad \mathcal{S}' = \sigma(\mathcal{S}').$$

It is proved that such sequences are exactly the Sturmian sequences $\mathcal{S}_{\alpha,\rho}$ with a *quadratic* irrational slope α and an intercept $\rho \in \mathbb{Q}(\alpha)$ (see [4]). If we furthermore require that $\mathcal{S}_{\alpha,\rho}$ be *itself* a fixed-point of a substitution, the previous characterization becomes that α is a *reduced* quadratic irrational, with some additional conditions on ρ (see e.g. [6, 14]). Let us recall (theorems of Lagrange and Galois) that an irrational number is quadratic (resp. reduced quadratic) if and only if its continued fraction expansion is *ultimately periodic* (resp. *purely periodic*).

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 236–247, 2005.
© Springer-Verlag Berlin Heidelberg 2005

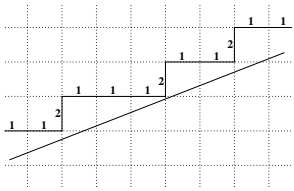


Fig. 1. The Sturmian sequence $\cdots 112111211211 \cdots$

In this paper, we would like to proceed by analogy in the bidimensional case in order to obtain similar results. The first difficulty arises from the analogy itself, which is not so obvious and with whom we deal in the first three sections. Section 1 defines *bidimensional Sturmian sequences*, our analogue of Sturmian sequences. Sections 2 and 3 give the definition of, respectively, the bidimensional substitutions and the bidimensional continued fraction expansion we have chosen, namely the *generalized substitutions* introduced in [3] and the Brun's algorithm (see [5]). It is indeed a *choice* since there is no canonical multidimensional definition of a substitution or of a continued fraction expansion.

Our main results are given in Section 4. We here restricted ourselves to the case of *homogenous* bidimensional Sturmian sequences, which correspond to the Sturmian sequences $\mathcal{S}_{\alpha,\rho}$ for which $\rho = 0$. Theorem 3 proves that such bidimensional sequences are S-adic (see e.g. [13] for more details about S-adicity), while Theorem 4 gives a partial characterization very similar to the unidimensional case: a bidimensional Sturmian sequence is proved to be *substitutive* (resp. *fixed-point of a substitution*) if its parameters – the equivalent of the slope α of a Sturmian sequence – have an *ultimately periodic* (resp. a *purely periodic*) bidimensional continued fraction expansion. Notice that a true characterization (for example α is a cubic irrational if and only if the corresponding bidimensional sequence is substitutive) is probably very hard, since there is still no generalization of the theorems of Lagrange and Galois which would say us which vectors have purely periodic or ultimatily periodic expansions for *some* continued fraction expansion (maybe not the one here used).

In Section 5, we examine the result of Section 4 from a more practical point of view: can we use the substitutions to *effectively* generate bidimensional Sturmian sequences? Though it does not completely solve the problem, Theorem 5 give a non trivial result in the substitutive case. We end the paper giving in Section 6 future extensions of the work presented here.

1 Stepped Planes and Bidimensional Sequences

We here show how to associate to a plane a bidimensional sequence, by analogy to the one-dimensional case. This analogy also leads to define Sturmian bidimensional sequences. One denotes (e_1, e_2, e_3) the canonical basis of $\mathbb{R}^3$.
The *face* (x, i^*), for $x \in \mathbb{Z}^3$ and $i \in \{1, 2, 3\}$ is defined by (see Fig. 2):

$$(x, i^*) = \{x + re_j + te_k \mid 0 \leq r, t \leq 1 \text{ and } i \neq j \neq k\}.$$

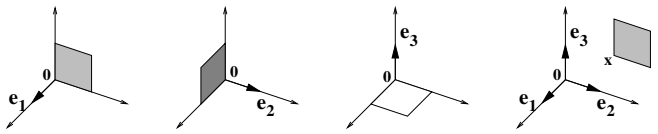


Fig. 2. From left to right: the faces $(0, i^*)$, $i = 1, 2, 3$ and $(x, 1^*) = x + (0, 1^*)$

These faces generate the $\mathbb{Z}$-module of the formal sums of weighted faces $\mathcal{G} = \{\sum m_{x,i}(x, i^*) \mid m_{x,i} \in \mathbb{Z}\}$, on which the lattice $\mathbb{Z}^3$ acts by translation: $y + m_{x,i}(x, i^*) = m_{x,i}(y + x, i^*)$.
One then uses faces to approximate planes of $\mathbb{R}^3$:

Definition 1. *Let $\mathcal{P}_{\alpha,\beta}$ be the homogenous plane of $\mathbb{R}^3$ defined by:*

$$\mathcal{P}_{\alpha,\beta} = \{x \in \mathbb{R}^3 \mid \langle x, {}^t(1, \alpha, \beta)\rangle = 0\}.$$

The stepped plane $\mathcal{S}_{\alpha,\beta}$ *associated to $\mathcal{P}_{\alpha,\beta}$ is defined by:*

$$\mathcal{S}_{\alpha,\beta} = \left\{(x, i^*) \mid \langle x, {}^t(1, \alpha, \beta)\rangle > 0 \text{ and } \langle x - e_i, {}^t(1, \alpha, \beta)\rangle \leq 0\right\},$$

and a patch *of $\mathcal{S}_{\alpha,\beta}$ is a finite subset of the faces of $\mathcal{S}_{\alpha,\beta}$.*

Notice that a patch of $\mathcal{S}_{\alpha,\beta}$ belongs to the $\mathbb{Z}$-module $\mathcal{G}$, but is geometric, that is, without multiple faces. According to the terminology introduced by Reveillès in [12], the stepped plane corresponds to the notion of *standard arithmetic plane* in discrete geometry.

We now recall from [1] (see also [2]) the way one can bijectively encode a stepped plane by a bidimensional sequence over three letters. We first define a bijective map from the faces of a stepped plane to its set of vertices:

Proposition 1 ([1]). *Let v be the map from the faces of $\mathbb{R}^3$ to the vertices of $\mathbb{Z}^3$ defined by (see Fig. 3, left):*

$$v : \begin{array}{l} (x, 1^*) \to x \\ (x, 2^*) \to x + e_1 \\ (x, 3^*) \to x + e_1 + e_2. \end{array}$$

Then v maps bijectively the faces of a stepped plane to its set of vertices.

We then define a bijective map from the vertices of a stepped plane to $\mathbb{Z}^2$:

Proposition 2 ([1]). *Let $\mathcal{S}_{\alpha,\beta}$ be a stepped plane. The orthogonal projection π on the plane $x + y + z = 0$ is a bijection from the vertices of $\mathcal{S}_{\alpha,\beta}$ to the lattice $\mathbb{Z}\pi(e_1) + \mathbb{Z}\pi(e_2)$. Thus the map $\tilde{\pi}$ defined by $\tilde{\pi}(x) = (m, n)$ if and only if $\pi(x) = m\pi(e_1) + n\pi(e_2)$ is a bijection from the vertices of $\mathcal{S}_{\alpha,\beta}$ to $\mathbb{Z}^2$. Moreover, one has the explicit formulas:*

$$\tilde{\pi}\begin{pmatrix} p \\ q \\ r \end{pmatrix} = (p - r, q - r),$$

$$\tilde{\pi}^{-1}(m,n) = \begin{pmatrix} m \\ n \\ 0 \end{pmatrix} + \left(1 - \left\lceil \frac{m+\alpha n}{1+\alpha+\beta} \right\rceil\right) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

And finally we give the encoding $\mathcal{U}_{\alpha,\beta}$ of a stepped plane $\mathcal{S}_{\alpha,\beta}$ (where $\mathcal{U}_{\alpha,\beta}$ (m,n) is the letter at position (m,n) in the bidimensional sequence $\mathcal{U}_{\alpha,\beta}$):

Proposition 3 ([1]). *Let ϕ be mapping a stepped plane $\mathcal{S}_{\alpha,\beta}$ to the bidimensional sequence $\mathcal{U}_{\alpha,\beta}$ over the alphabet $\{1,2,3\}$ defined by:*

$$(x,i^*) \in \mathcal{S}_{\alpha,\beta} \quad \Leftrightarrow \quad \mathcal{U}_{\alpha,\beta}(\tilde{\pi} \circ v(x,i^*)) = i,$$

Then ϕ is one-to-one from the set $\{\mathcal{S}_{\alpha,\beta} \mid 0 < \alpha, \beta < 1\}$ to the set of the bidimensional sequences over $\{1,2,3\}$ (see Fig. 3). Notice that not all the bidimensional sequences over $\{1,2,3\}$ correspond to a stepped plane.

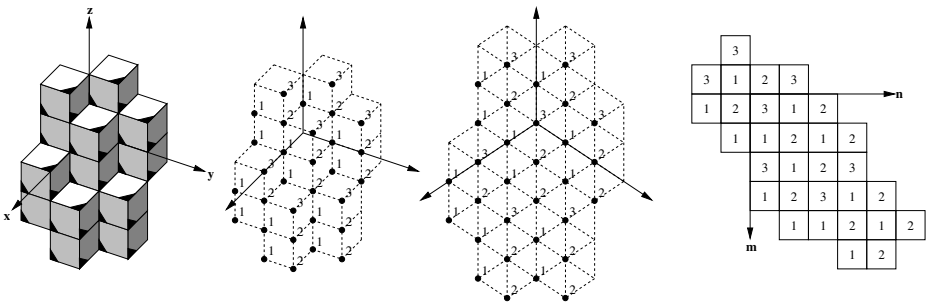


Fig. 3. From left to right: to each face corresponds a proper vertex (at its blacked corner); type 1, 2 or 3 of a vertex depends on the type of its corresponding face; the projection $\tilde{\pi}$ on the plane $x+y+z=0$ maps the vertices to a 2-dimensional lattice; we thus obtain a bidimensional sequence over $\{1,2,3\}$

One then defines *Sturmian stepped planes* and *bidimensional Sturmian sequences* by analogy with the unidimensional case:

Definition 2. *A stepped plane $\mathcal{S}_{\alpha,\beta}$ is a* Sturmian stepped plane *if* 1, *α and β are linearly independent over $\mathbb{Q}$. A* bidimensional Sturmian sequence *is the image under ϕ of a Sturmian stepped plane.*

Thus, ϕ is a bijection between the Sturmian stepped planes and the bidimensional Sturmian sequences, for which we furthermore have explicit formulas.

2 Generalized Substitutions

We here define substitutions that act on stepped planes (or, equivalently, on the bidimensional sequences corresponding to stepped planes). These substitutions are the *generalized substitutions*, introduced in [3] (see also [11], Chap. 8).

Let us recall that the *incidence matrix* M_σ of a (classic) substitution σ gives at position (i,j) the number of occurences of the letter i in the word $\sigma(j)$. Moreover, σ is said *unimodular* if $detM_\sigma = \pm 1$. We are now in a position to define the *generalized substitutions*:

Definition 3. *The* generalized substitution *associated to the unimodular substitution σ is the endomorphism Θ_σ of $\mathcal{G}$ defined by:*

$$\begin{cases} \forall i \in \mathcal{A}, & \Theta_\sigma(0, i^*) = \sum_{j=1}^{3} \sum_{s:\sigma(j)=p\cdot i\cdot s} \left(M_\sigma^{-1}(f(s)), j^*\right), \\ \forall x \in \mathbb{Z}^3,\ \forall i \in \mathcal{A}, & \Theta_\sigma(x, i^*) = M_\sigma^{-1} x + \Theta_\sigma(0, i^*), \\ \forall \sum m_{(x,i)}(x, i^*) \in \mathcal{G}, & \Theta_\sigma\left(\sum m_{(x,i)}(x, i^*)\right) = \sum m_{(x,i)} \Theta_\sigma(x, i^*), \end{cases}$$

where $f(w) = (|w|_1, |w|_2, |w|_3)$ and $|w|_i$ is the number of occurences of the letter i in w.

Example 1. Let us consider the *Rauzy substitution* σ:

$$\sigma : \begin{array}{l} 1 \to 12 \\ 2 \to 13 \\ 3 \to 1 \end{array}, \qquad M_\sigma = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

σ is unimodular, and one easily computes (see Fig. 4):

$$\Theta_\sigma : \begin{array}{l} (e_1, 1^*) \mapsto (e_1, 1^*) + (e_2, 2^*) + (e_3, 3^*), \\ (e_2, 2^*) \mapsto (e_1 - e_3, 1^*), \\ (e_3, 3^*) \mapsto (e_2 - e_3, 2^*). \end{array}$$

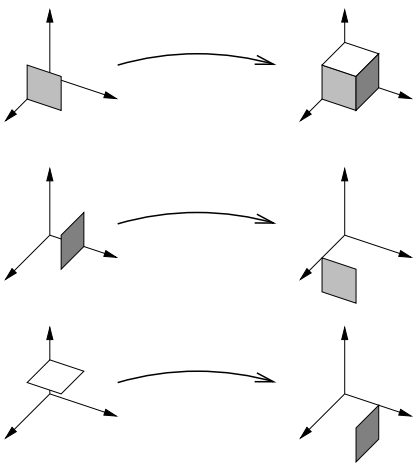


Fig. 4. The endomorphism Θ_σ for the Rauzy substitution: action on (e_i, i^*)

We now define an especially interesting type of substitution:

Definition 4. *A substitution σ is of* Pisot type *if its incidence matrix M_σ has eigenvalues μ_1, μ_2 and λ satisfying $0 < |\mu_1|, |\mu_2| < 1 < \lambda$. The generalized substitution Θ_σ is then also said of Pisot type.*

If σ is of Pisot type and if ${}^t(1,\alpha,\beta)$ is the *left* eigenvector of M_σ for the dominant eigenvalue λ (that is, ${}^tM_\sigma{}^t(1,\alpha,\beta) = \lambda^t(1,\alpha,\beta)$), the plane $\mathcal{P}_{\alpha,\beta}$ is called the *contracting invariant plane* of σ and satisfies:

Proposition 4. *$\exists\mu$, $0 < \mu < 1$, such that if $x \in \mathcal{P}_{\alpha,\beta}$, then $M_\sigma x \in \mathcal{P}_{\alpha,\beta}$ and one has:*

$$||M_\sigma x|| \leq \mu||x||.$$

The action of Θ_σ, when of Pisot type, on the stepped plane $\mathcal{S}_{\alpha,\beta}$ has some nice properties proved in [3]. Indeed, Θ_σ maps each patch of $\mathcal{S}_{\alpha,\beta}$ to a patch of $\mathcal{S}_{\alpha,\beta}$, the unit cube $\mathcal{U} = \{(e_i, i^*),\ i = 1,2,3\}$ is always a patch of $\mathcal{S}_{\alpha,\beta}$ and the sequence $(\Theta^n_\sigma(\mathcal{U}))$ is strictly increasing for inclusion and thus generates arbitrarily large patches of $\mathcal{S}_{\alpha,\beta}$ (see Fig. 5).

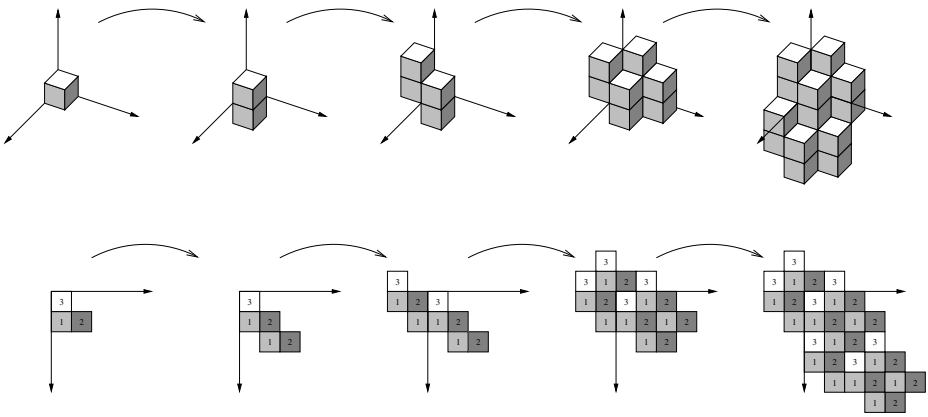


Fig. 5. $\Theta^n_\sigma(\mathcal{U})$ (top) and $(\phi\circ\Theta_\sigma\circ\phi^{-1})^n(\mathcal{U})$ (bottom) for the Rauzy substitution (which is of Pisot type), $n = 0,1,2,3,4$. Notice that the action of Θ_σ is not so obvious

3 Bidimensional Continued Fractions

Contrary to the unidimensional case with the Euclid's algorithm, there is no canonical continued fraction expansion in the bidimensional case. We thus fix here the expansion we will use further, that is the one produced by the *modified Jacobi-Perron algorithm*, which is a two-point extension of Brun's algorithm. Let us recall this algorithm (see e.g. [5] for more details):

Definition 5. *Let be $X = [0,1) \times [0,1)$ and T the map defined on $X\backslash(0,0)$ by:*

$$T(\alpha,\beta) = \begin{cases} \left(\frac{\beta}{\alpha}, \frac{1}{\alpha} - \left\lfloor\frac{1}{\alpha}\right\rfloor\right) & \text{if } \alpha \geq \beta, \\ \left(\frac{1}{\beta} - \left\lfloor\frac{1}{\beta}\right\rfloor, \frac{\alpha}{\beta},\right) & \text{if } \alpha < \beta. \end{cases}$$

For $n \geq 1$ and if possible (that is, while $\alpha_{n-1} \neq 0$), one denotes:

$$(\alpha_n, \beta_n) = T^n(\alpha, \beta),$$

and defines:

$$(a_n, \varepsilon_n) = \begin{cases} \left(\left\lfloor \frac{1}{\alpha_{n-1}} \right\rfloor, 0 \right) & \text{if } \alpha_{n-1} \geq \beta_{n-1}, \\ \left(\left\lfloor \frac{1}{\beta_{n-1}} \right\rfloor, 1 \right) & \text{if } \alpha_{n-1} < \beta_{n-1}. \end{cases}$$

The sequence (a_n, ε_n) is called the continued fraction expansion *of (α, β) (notice that $a_n \in \mathbb{N}^*$ and $\varepsilon_n \in \{0, 1\}$). This sequence is infinite iff 1, α and β are linearly independent over $\mathbb{Q}$*

Let us give a matrix viewpoint on this algorithm. For $a \in \mathbb{N}^*$, one defines the substitutions:

$$\sigma_{(a,0)} : \begin{array}{l} 1 \to \overbrace{11\cdots1}^{a \text{ times}}3 \\ 2 \to 1 \\ 3 \to 2 \end{array}, \qquad \sigma_{(a,1)} : \begin{array}{l} 1 \to \overbrace{11\cdots1}^{a \text{ times}}2 \\ 2 \to 3 \\ 3 \to 1 \end{array},$$

whose incident matrices are:

$$A_{(a,0)} = \begin{pmatrix} a & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \qquad A_{(a,1)} = \begin{pmatrix} a & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

So that, with $(\alpha_0, \beta_0) = (\alpha, \beta)$ and $\eta_k = \max(\alpha_{k-1}, \beta_{k-1})$, one has for $n \geq 1$:

$$\eta_n {}^t A_{(a_n,\varepsilon_n)} \begin{pmatrix} 1 \\ \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} 1 \\ \alpha_{n-1} \\ \beta_{n-1} \end{pmatrix}.$$

We can give an expanded formulation of the previous equality:

Proposition 5. *Let 1, α and β be linearly independent over $\mathbb{Q}$ and let (a_n, ε_n) be the continued fraction expansion of (α, β). Then there exists a sequence (α_n, β_n) of couples in $[0, 1)^2$ such that:*

$$\forall n \in \mathbb{N}, \qquad \begin{pmatrix} 1 \\ \alpha \\ \beta \end{pmatrix} = (\eta_1 \eta_2 \cdots \eta_n)^t A_{(a_1,\varepsilon_1)} {}^t A_{(a_2,\varepsilon_2)} \cdots {}^t A_{(a_n,\varepsilon_n)} \begin{pmatrix} 1 \\ \alpha_n \\ \beta_n \end{pmatrix}.$$

4 Substitutions and Bidimensional Sturmian Sequences

The previous sections have successively defined the bidimensional Sturmian sequences (or, equivalently, the Sturmian stepped planes), substitutions acting on these sequences and a bidimensional continued fraction expansion. We thus are now in a position to try to extend to the bidimensional case the results for (unidimensional) Sturmian sequences given in the introduction.

Let us first define a generalized substitution which plays a specific role:

Definition 6. *The generalized substitution associated to the unimodular substitution of Pisot type $\sigma_{(a,\varepsilon)}$ introduced in Section 3 is denoted $\Theta_{(a,\varepsilon)}$. Such a generalized substitution is said of* Brun type.

We then have the following fundamental theorem:

Theorem 1. *$\Theta_{(a_n,\varepsilon_n)}$ is a bijection from $\mathcal{S}_{\alpha_n,\beta_n}$ onto $\mathcal{S}_{\alpha_{n-1},\beta_{n-1}}$.*

We shall stress that there is no contradiction between Theorem 1 and the results recalled at the end of Section 2, which would yield here that $\Theta_{(a_n,\varepsilon_n)}$ is a one-to-one map from the stepped plane associated to its contracting invariant plane to itself. Indeed, neither $\mathcal{P}_{\alpha_n,\beta_n}$ nor $\mathcal{P}_{\alpha_{n-1},\beta_{n-1}}$ are invariant planes of $\Theta_{(a_n,\varepsilon_n)}$ (except if the expansion (a_n,ε_n) is purely periodic of period 1, in which case all these planes are identical).

As we did in Proposition 5, we can give an expanded formulation of Theorem 1:

Theorem 2. *Let $\mathcal{S}_{\alpha,\beta}$ be a Sturmian stepped plane and (a_n,ε_n) be the continued fraction expansion of (α,β). Then there exists a sequence $(\mathcal{S}_{\alpha_n,\beta_n})$ of Sturmian stepped planes such that:*

$$\forall n \in \mathbb{N}, \qquad \mathcal{S}_{\alpha,\beta} = \Theta_{(a_1,\varepsilon_1)} \circ \Theta_{(a_2,\varepsilon_2)} \circ \cdots \circ \Theta_{(a_n,\varepsilon_n)} \left(\mathcal{S}_{\alpha_n,\beta_n}\right).$$

We thus obtain for $\mathcal{S}_{\alpha,\beta}$ an equation – called *expansion* – which looks like the classic S-adic expansion of a Sturmian sequence (see e.g. [13] for more details on S-adicity), though the number of different substitutions of our expansion may be unbounded. We will fix this last point thanks to the following proposition:

Proposition 6. *Let us define the substitutions:*

$$\sigma_0 : \begin{array}{l} 1 \to 1 \\ 2 \to 2 \\ 3 \to 13 \end{array}, \qquad \gamma_0 : \begin{array}{l} 1 \to 3 \\ 2 \to 1 \\ 3 \to 2 \end{array}, \qquad \sigma_1 : \begin{array}{l} 1 \to 1 \\ 2 \to 12 \\ 3 \to 3 \end{array}, \qquad \gamma_1 : \begin{array}{l} 1 \to 2 \\ 2 \to 3 \\ 3 \to 1 \end{array}.$$

These substitutions are unimodular and verify:

$$\forall (a,\varepsilon) \in \mathbb{N} \times \{0,1\}, \qquad \sigma_{(a,\varepsilon)} = \sigma_\varepsilon^a \circ \gamma_\varepsilon.$$

An induction easily proves Proposition 6. Let Σ_ε and Γ_ε be the generalized substitutions associated to σ_ε and γ_ε. A computation yields $\Theta_{\sigma\circ\sigma'} = \Theta_{\sigma'} \circ \Theta_\sigma$, and Proposition 6 allows us to rewrite Theorem 2 in the following way:

Theorem 3. *Let $\mathcal{S}_{\alpha,\beta}$ be a Sturmian stepped plane and (a_n,ε_n) be the continued fraction expansion of (α,β). Then there exists a sequence $(\mathcal{S}_{\alpha_n,\beta_n})$ of Sturmian stepped planes such that:*

$$\forall n \in \mathbb{N}, \qquad \mathcal{S}_{\alpha,\beta} = \Gamma_{\varepsilon_1} \circ \Sigma_{\varepsilon_1}^{a_1} \circ \Gamma_{\varepsilon_2} \circ \Sigma_{\varepsilon_2}^{a_2} \circ \cdots \circ \Gamma_{\varepsilon_n} \circ \Sigma_{\varepsilon_n}^{a_n} \left(\mathcal{S}_{\alpha_n,\beta_n}\right),$$

where Γ_ε and Σ_ε are associated to the substitutions defined in Proposition 6.

We now consider the case of periodic expansions. Let us recall that a sequence (u_n) is *ultimately periodic with period p and threshold d* if $n > d \Rightarrow u_{n+p} = u_n$. If moreover $d = 0$, the sequence is said *purely periodic*. In this case, one has:

Theorem 4. *Let $\mathcal{S}_{\alpha,\beta}$ be a Sturmian stepped plane and (a_n, ε_n) be the continued fraction expansion of (α, β). If this expansion is ultimately periodic, then there exist two generalized substitutions Θ_d and Θ_p, and a stepped plane $\mathcal{S}_p$ such that:*

$$\mathcal{S}_{\alpha,\beta} = \Theta_d(\mathcal{S}_p), \quad \text{with} \quad \mathcal{S}_p = \Theta_p(\mathcal{S}_p).$$

And if the expansion is purely periodic, one has simply:

$$\mathcal{S}_{\alpha,\beta} = \Theta_p(\mathcal{S}_{\alpha,\beta}).$$

Proof. It follows easily from Theorem 2 with:

$$\begin{aligned}
\Theta_d &= \Theta_{(a_1,\varepsilon_1)} \circ \Theta_{(a_2,\varepsilon_2)} \circ \cdots \circ \Theta_{(a_d,\varepsilon_d)},\\
\Theta_p &= \Theta_{(a_{d+1},\varepsilon_{d+1})} \circ \Theta_{(a_{d+2},\varepsilon_{d+2})} \circ \cdots \circ \Theta_{(a_{d+p},\varepsilon_{d+p})},\\
\mathcal{S}_p &= \mathcal{S}_{\alpha_d,\beta_d},
\end{aligned}$$

where p is the period of the expansion of (α, β) and d its threshold. □

According to the terminology of [7, 13], Theorem 3 and 4 state that a bidimensional Sturmian sequence $\mathcal{U}_{\alpha,\beta}$ has always a S-adic expansion, and is substitutive (resp. a fixed-point of a substitution) if the expansion of (α, β) is ultimately periodic (resp. purely periodic). Notice that, contrary to the unidimensional case, we do not yet obtain a complete characterization of bidimensional Sturmian sequences that are substitutive or fixed-point of a substitution. We will discuss this more carefully in the last section.

5 Effective Generation of Stepped Planes

It is to notice that successive applications of generalized substitutions on a **finite** initial patch do not necessarily cover, to infinity, the **whole** stepped plane but only an infinite subset of it (think for example about a non simply-connected subset or a cone ...). Such a problem, that we investigate in this section, can however be of great practical interest, for example to effectively generate standard arithmetic plane in discrete geometry.

The following lemma deals with the "almost" expansivity of a generalized substitution of Pisot type:

Lemma 1. *Let σ be a unimodular substitution σ of Pisot type (all the notations are those of Section 2). Then there exist $k \in [0, 1)$ and $C \in \mathbb{R}^+$ such that:*

$$\begin{cases} (x, i^*) \in \mathcal{S}_{\alpha,\beta} \\ (y, j^*) \in \Theta_\sigma(x, i^*) \\ ||y|| \geq C \end{cases} \quad \Rightarrow \quad ||x|| \leq k||y||.$$

It provides us a case in which one we can generate the whole stepped plane:

Theorem 5. *Let $\mathcal{S}_{\alpha,\beta}$ be a substitutive stepped plane, that is, a stepped plane such that there exist two generalized substitutions Θ_d and Θ_p verifying:*

$$\mathcal{S}_{\alpha,\beta} = \Theta_d(\mathcal{S}_p), \quad \textit{with} \quad \mathcal{S}_p = \Theta_p(\mathcal{S}_p).$$

If Θ_p is of Pisot type and bijective on the set of faces of $\mathcal{S}_p$, then there exists a finite patch P of $\mathcal{S}_p$ such that:

$$\mathcal{S}_{\alpha,\beta} = \Theta_d\left(\lim_{n\to+\infty} \Theta_p^n(P)\right).$$

Proof. Let C and k be the constants of Lemma 1 for the substitution Θ_p and let P be the patch formed by the faces (x, i^*) of $\mathcal{S}_p$ such that $||x|| \leq C$.

Let (y, j^*) be a face of $\mathcal{S}_p$. Consider the sequence $(y_m, j_m^*)_{m\geq 1}$ such that $(y_1, j_1^*) = (y, j^*)$ and $\Theta_p(y_{m+1}, j_{m+1}^*) = (y_m, j_m^*)$. This sequence is well defined since Θ_p is bijective. While $||y_m|| \geq C$, Lemma 1 yields $||y_{m+1}|| \leq k||y_m||$, with $k < 1$. Hence for m large enough, one has $||y_m|| \leq C$, that is, $(y_m, j_m^*) \in P$ and $\Theta_p^m(y_m, j_m^*) = (y, j^*)$. □

In particular, by Theorem 1 and 4, the previous theorem holds if (α, β) has an ultimately periodic expansion of period p and threshold d, **under the hypothesis** that $\Theta_p = \Theta_{(a_{d+1},\varepsilon_{d+1})} \circ \Theta_{(a_{d+2},\varepsilon_{d+2})} \circ \cdots \circ \Theta_{(a_{d+p},\varepsilon_{d+p})}$ is of Pisot type (what can be false since for example $\Theta_{(1,1)} \circ \Theta_{(1,0)}$ is not of Pisot type). Proposition 6 can be used in practice to iterate on P only four different generalized substitutions, whatever Θ_p may be.

Example 2. Let (α, β) have the purely periodic expansion $[(1,1),(1,0),(1,0)]^*$ (of period three). One computes $A_{(1,0)}A_{(1,0)}A_{(1,1)} = M_\sigma^2$, where σ is the Rauzy substitution introduced in Section 2. Thus $\Theta_p = \Theta_{(1,1)} \circ \Theta_{(1,0)} \circ \Theta_{(1,0)}$ is of Pisot type and $\mathcal{S}_{\alpha,\beta}$, fixed-point of Θ_p, can be generated applying Θ_p to a finite patch.

Notice that it is easy to see in the previous example that Θ_p and Θ_σ have the same invariant plane, and thus both generate, starting from P, patches of the same stepped plane. But we shall stress that these patches are not necessarily the same: there is many way to generate growing parts of the stepped plane.

6 Perspectives

This paper has defined the bidimensional Sturmian sequences (or, equivalently, the Sturmian stepped planes), on which act the generalized substitutions introduced in [3]. We have proved that every bidimensional Sturmian sequence is S-adic (according to the terminology of [13]), which extends to the bidimensional case the analogous result already known for unidimensional Sturmian sequences.

Similarly, the sufficient condition (on ultimately or purely periodic continued fraction expansions), for unidimensional Sturmian sequences to be substitutive

or fixed-point of a substitution, has been here extended to an analogous condition (on Brun's expansions), for bidimensional Sturmian sequences. However, we did not prove that our condition is also necessary, though it holds in the unidimensional case. A way to fix that, could be to extend the notion of *return word* – introduced in [7] and used to prove the unidimensional case – to some suitable bidimensional analogous notion of "return pattern". Such a bidimensional extension of return word have already been done in [10]. We hence have good hopes to complete our characterization of substitutive bidimensional Sturmian sequences.

As noticed in the introduction, we focused on the *homogenous* case, that is the analogous of the unidimensional Sturmian sequences with intercept equals to zero (that is, $\mathcal{S}_{\alpha,\rho}$ with $\rho = 0$, according to the notation of the introduction). Indeed, instead of the plane $\mathcal{P}_{\alpha,\beta}$ of Definition 1, we should consider the general case of a plane $\mathcal{P}_{\alpha,\beta,\rho} = {}^t(0,0,\rho) + \mathcal{P}_{\alpha,\beta}$. In the unidimensional case, taking into account an intercept ρ just leads to additional conditions that are, roughly speaking, conditions on the Ostrowski expansion of ρ similarly to the conditions on the continued fraction expansion of α (see [4]). It remains to give and prove some similar conditions on the intercept in the bidimensional case.

Last, we could carry out some improvements to the more practical results of Section 5. Indeed, starting from a *finite* initial patch to iterate a substitution is certainly more convenient than starting from the whole plane. But it is not so easy to compute this finite patch. Could not the unit cube $\mathcal{U}$, which is proved to be a patch of *any* stepped plane, suffices to generate the whole plane, as it is the case for the Rauzy substitution? Some counter-examples prove that the answer is in general negative, but it would be interesting to characterize the "good" cases. Similarly, conditions to have the substitution Θ_p of Theorem 5 of Pisot type (and thus, suitable to generate the plane) would be interesting.

Acknowledgements

We would like to thank Valérie Berthé, Pierre Arnoux and the anonymous referees for many suggestions and corrections.

References

1. P. Arnoux, V. Berthé, S. Ito, *Discrete planes, $\mathbb{Z}^2$-actions, Jacobi-Perron algorithm and substitutions*. Ann. Inst. Fourier (Grenoble) **52** (2002), 1001–1045.
2. P. Arnoux, V. Berthé, A. Siegel, *Two-dimensional iterated morphisms and discrete planes*. Theoret. Comput. Sci. **319** no. 1-3 (2004), 145–176.
3. P. Arnoux, S. Ito, *Pisot substitutions and Rauzy fractals*. Bull. Belg. Math. Soc. Simon Stevin **8** no. 2 (2001), 181–207.
4. V. Berthé, C. Holton, L. Q. Zamboni, *Initial powers of Sturmian sequences*. Acta Arithmetica, to appear.
5. A. J. Brentjes, *Multi-dimensional continued fraction algorithms*. Mathematical Centre Tracts 145, Matematisch Centrum, Amsterdam, 1981.

6. D. Crisp, W. Moran, A. Pollington, P. Shiue, *Substitution invariant cutting sequences*. J. Théor. Nombres Bordeaux **5** (1993), 123–137.
7. F. Durand, *A characterization of substitutive sequences using return words*. Inventiones Math. **132** (1998), 179–188.
8. S. Ferenczi, *Rank and symbolic complexity*. Ergodic Theory Dynam. Systems **16** (1996), 663–682.
9. M. Lothaire, *Algebraic combinatorics on words*, Cambridge University Press, 2002.
10. N. Priebe, *Towards a characterization of self-similar tilings in terms of derived Voronoï tessellations*. Geom. Dedicata **79** (2000), 239–265.
11. N. Pytheas Fogg, *Substitutions in Dynamics, Arithmetics and Combinatorics*. Lecture Notes in Math. **1794** (2002), Springer Verlag.
12. J.-P. Reveillès, *Calcul en nombres entiers et algorithmique*. Thèse d'etat, Univ. Louis Pasteur, Strasbourg, France, 1991.
13. K. Wargan, *S-adic dynamical systems and Bratelli diagrams*. PhD thesis, George Washington University, 2001.
14. S. Yasutomi, *On Sturmian sequences which are invariant under some substitution*. Number theory and its applications (Kyoto, 1997), 347–373, Dev. Math. **2**, Luwer Acad. Publ., Dordrecht, 1999.

Unambiguous Morphic Images of Strings

Dominik D. Freydenberger, Daniel Reidenbach*,**, and Johannes C. Schneider

Fachbereich Informatik, Technische Universität Kaiserslautern,
Postfach 3049, 67653 Kaiserslautern, Germany
{d_freyde,reidenba,j_schne}@informatik.uni-kl.de

Abstract. Motivated by the research on pattern languages, we study a fundamental combinatorial question on morphisms in free semigroups: With regard to any string α over some alphabet we ask for the existence of a morphism σ such that $\sigma(\alpha)$ is unambiguous, i.e. there is no morphism ρ with $\rho \neq \sigma$ and $\rho(\alpha) = \sigma(\alpha)$. Our main result shows that a rich and natural class of strings is provided with unambiguous morphic images.

1 Introduction

In the past decades a lot of effort has been spent on investigating the properties of those *morphisms* which map a string over some alphabet Σ onto a string over a second alphabet Σ' (cf., e.g., Lothaire [9], Choffrut and Karhumäki [2], Harju and Karhumäki [3]). In this context, many problems only arise if Σ contains more symbols than Σ', and therefore – in order to address these difficulties as precisely as possible – we assume, for the remainder of our paper, $\Sigma = \mathbb{N}$ and $\Sigma' = \{\mathtt{a}, \mathtt{b}\}$. Consequently, we regard the set of morphisms mapping the strings in an infinitely generated free semigroup to the strings in a free monoid with two generators. According to the closely related research on *pattern languages* (cf. Mateescu and Salomaa [10]) we call an element of $\mathbb{N}^+$ a *pattern* and an element of $\{\mathtt{a}, \mathtt{b}\}^*$ a *word*. We separate all symbols in a pattern by a dot (see, e.g., the example pattern α' below) so as to avoid any confusion.

Quite a number of the basic questions to be asked on suchlike mappings deals with the problem of finding a morphism which, in spite of the resulting alphabet reduction, preserves the structure of its input string as far as possible; this is a manifest topic, e.g., in the theory of codes (cf. Jürgensen and Konstantinidis [5]). Even though any answer to this question strongly depends on the formal definition of what is considered to be a "structure-preserving" morphism, from a very intuitive point of view, one surely would agree that, for instance, the shape of the pattern $\alpha' = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 1 \cdot 4 \cdot 3 \cdot 2$ is not adequately reflected by its morphic image $w_1 = \mathtt{a}^{10}$. Obviously, for such a task, a code – that, in our sense, is nothing but an *injective* morphism – is a more appropriate choice: If we apply the morphism $\sigma'(i) = \mathtt{a}\,\mathtt{b}^i$, $i \in \mathbb{N}$, to α' then we receive the word $w_2 = \sigma'(\alpha) = \mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}^2\,\mathtt{a}\,\mathtt{b}^3\,\mathtt{a}\,\mathtt{b}^4\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}^4\,\mathtt{a}\,\mathtt{b}^3\,\mathtt{a}\,\mathtt{b}^2$ which, due to the distinct lengths

* Corresponding author

** Partly supported by Deutsche Forschungsgemeinschaft (DFG), Grant Wi 1638/1-3

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 248–259, 2005.
© Springer-Verlag Berlin Heidelberg 2005

of subwords over the single symbol $\mathtt{b}$, that allow the definition of an inverse morphism, seems to describe its preimage quite accurately.

However, in those settings where we are confronted with a variety of morphic images of one and the same pattern α – such as in *inductive inference of pattern languages* (cf., e.g., Angluin [1]), that deals with the inferrability of an (unknown) pattern from the set of *all* of its morphic images – the mere injectivity of morphisms can turn out to be insufficient for reflecting the shape of α. Surprisingly, with regard to these problems, a second property of a morphic image w demonstrably is much more important, namely its *ambiguity* (cf. Reidenbach [13]), i.e. the question whether there are at least two morphisms σ, ρ such that, for some symbol i in α, $\sigma(i) \neq \rho(i)$, but nevertheless $\sigma(\alpha) = w = \rho(\alpha)$. Returning to our example it can easily be seen that w_2 is ambiguous with respect to α' as it can, e.g., also be generated by the morphism ρ' with $\rho'(1) = \mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}^2$, $\rho'(2) = \varepsilon$, $\rho'(3) = \mathtt{a}\,\mathtt{b}^3\,\mathtt{a}\,\mathtt{b}^2$ and $\rho'(4) = \mathtt{b}^2$, where ε is the empty word:

$$\overbrace{\underbrace{\mathtt{a}\ \mathtt{b}}_{\sigma'(1)}\ \underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(2)}}^{\rho'(1)}\ \overbrace{\underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(3)}\ \underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(4)}}^{\rho'(3)}\ \overbrace{\underbrace{\mathtt{b}\ \mathtt{b}}_{\sigma'(4)}}^{\rho'(4)}\ \overbrace{\underbrace{\mathtt{a}\ \mathtt{b}}_{\sigma'(1)}\ \underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(4)}}^{\rho'(1)}\ \overbrace{\underbrace{\mathtt{b}\ \mathtt{b}}_{\sigma'(4)}}^{\rho'(4)}\ \overbrace{\underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(3)}\ \underbrace{\mathtt{a}\ \mathtt{b}\ \mathtt{b}}_{\sigma'(2)}}^{\rho'(3)}$$

Consequently, w_2 does not adequately substantiate the existence of the symbol 2 in α' since this symbol is not needed for generating w_2; thus, from that point of view and in spite of its injectivity, we do not consider σ' to meet our vague yet well-founded requirements for a structure-preserving morphism. But even if we restrict our examination to nonerasing morphisms, i.e. if we use the free semigroup $\{\mathtt{a},\mathtt{b}\}^+$ instead of $\{\mathtt{a},\mathtt{b}\}^*$ as value set of the morphisms, the multitude of potential generating morphisms blurs the evidence of α in w_2.

Unfortunately, this ambiguity of words is a frequent property of many patterns, and, effortlessly, examples can be given for which there is no morphism at all leading to an unambiguous word; on the other hand, it is by no means obvious for which patterns there exist such structure-preserving morphic images. In the present paper, we examine this combinatorial problem of intrinsic interest systematically. To this end, we concentrate on the ambiguity of those words that are images of injective morphisms, and we explicitly distinguish between the general case where the set of all morphisms $\rho_{\mathrm{E}} : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^*$ is considered and the restricted case that focuses on nonerasing morphisms $\rho_{\mathrm{NE}} : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^+$. Our paper is organised as follows: After some brief formal definitions we collect a number of rather evident preliminary results before we show that a rich and natural class of patterns is characterised by the ability of morphically generating unambiguous words. This main result answers a question posed in [12].

Obviously, our work shows some connections to *equality sets* (cf., e.g., Harju and Karhumäki [3], Lipponen and Păun [8]): If, for some pattern α, we find a morphism σ such that $\sigma(\alpha)$ is unambiguous then α is a "non-solution" to the Post Correspondence Problem for σ and any other morphism ρ. Finally, it seems worth mentioning that, in a sense, our work complements a research that has been initiated by Mateescu and Salomaa [11]. As explained above we show that, for *every* pattern in some class, there exists *at least one* word that has exactly

one generating morphism, whereas, in a more general context, [11] examines the question whether, for an arbitrary upper bound $n \in \mathbb{N}$, there exists *at least one* pattern such that *each* of its morphic images has at most n distinct generating morphisms. In our restricted setting, for all patterns with occurrences of at least two different symbols, this question has a trivial answer in the negative.

2 Definitions and Basic Notes

We begin the formal part of this paper with a number of basic definitions. Major parts of our terminology are adopted from the research on pattern languages (cf. Mateescu and Solomaa [10]). Additionally, for notions not explained explicitly, we refer the reader to Choffrut and Karhumäki [2].

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Let Σ be an *alphabet*, i.e. an enumerable set of symbols. We regard two different alphabets: $\mathbb{N}$ and $\{\mathtt{a}, \mathtt{b}\}$ with $\mathtt{a} \neq \mathtt{b}$. Henceforth we call any symbol in $\mathbb{N}$ a *variable* and any symbol in $\{\mathtt{a}, \mathtt{b}\}$ a *letter*. A *string (over Σ)* is a finite sequence of symbols from Σ. For the *concatenation* of two strings $w_1, w_2 \in \Sigma^*$ we write $w_1 \cdot w_2$ or simply $w_1 w_2$. The string that results from the n-fold concatenation of a string w occasionally is denoted by w^n. $|\, x \,|$ stands for the size of a set x or the length of a string x, respectively. We denote the *empty string* by ε, i.e. $|\varepsilon| = 0$. In order to distinguish between a string over $\mathbb{N}$ and a string over $\{\mathtt{a}, \mathtt{b}\}$, we call the former a *pattern* and the latter a *word*. We name patterns with lower case letters from the beginning of the Greek alphabet such as α, β, γ. With regard to an arbitrary pattern α, $\mathrm{var}(\alpha)$ denotes the set of all variables occurring in α. For every alphabet Σ, Σ^* is the set of all (empty and non-empty) strings over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. We say that a string $v \in \Sigma^*$ is a *substring* of a string $w \in \Sigma^*$ if and only if, for some $u_1, u_2 \in \Sigma^*$, $w = u_1 v\, u_2$. Subject to the concrete alphabet considered, we call a substring a *subword* or *subpattern*. Additionally, we use the notions $w = \ldots v \ldots$ if v is a substring of w, $w = v \ldots$ if v is a *prefix* of w, and $w = \ldots v$ if v is a *suffix* of w. $|w|_v$ denotes the number of occurrences of a substring v in a string w. We do not use this notion for substrings with overlapping occurrences.

Since we deal with free semigroups, a *morphism* σ is a mapping that is compatible with the concatenation, i.e. for patterns $\alpha, \beta \in \mathbb{N}^+$, a morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ satisfies $\sigma(\alpha \cdot \beta) = \sigma(\alpha) \cdot \sigma(\beta)$. Hence, a morphism is fully explained as soon as it is declared for all variables in $\mathbb{N}$. Note that we restrict ourselves to total morphisms, even though we normally declare a morphism only for those variables explicitly that, in the respective context, are relevant.

Let $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ be a morphism. Then σ is called *nonerasing* provided that, for every $i \in \mathbb{N}$, $\sigma(i) \neq \varepsilon$. Note that σ necessarily is nonerasing if it is injective. For any pattern $\alpha \in \mathbb{N}^+$ with $\sigma(\alpha) \neq \varepsilon$, we call $\sigma(\alpha)$ *weakly unambiguous (with respect to α)* if there is no nonerasing morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ such that $\rho(\alpha) = \sigma(\alpha)$ and, for some $i \in \mathrm{var}(\alpha)$, $\rho(i) \neq \sigma(i)$. If, in addition, there is no (arbitrary) morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ with $\rho(\alpha) = \sigma(\alpha)$ and $\rho(i) \neq \sigma(i)$ for some $i \in \mathrm{var}(\alpha)$, then $\sigma(\alpha)$ is called *(strongly) unambiguous (with respect to α)*. Obviously, if $\sigma(\alpha)$ is strongly unambiguous then it is weakly unambiguous as

well. Finally, $\sigma(\alpha)$ is *ambiguous (with respect to α)* if and only if it is not weakly unambiguous.

As mentioned above, our subject is closely related to pattern languages (cf., e.g., Mateescu and Salomaa [10]), and therefore we consider it useful to provide an adequate background for some explanatory remarks. The *pattern language* of a pattern is the set of all of its possible morphic images in some fixed free monoid Σ^* (in our case $\Sigma = \{\mathtt{a}, \mathtt{b}\}$). More precisely and with regard to any $\alpha \in \mathbb{N}^+$, we distinguish between its *E-pattern language* $L_{\mathrm{E}}(\alpha) = \{\sigma(\alpha) \mid \sigma : \mathbb{N}^+ \longrightarrow \Sigma^*\}$ and its *NE-pattern language* $L_{\mathrm{NE}}(\alpha) = \{\sigma(\alpha) \mid \sigma : \mathbb{N}^+ \longrightarrow \Sigma^+\}$. Note that this definition implies that the full *class of E-(resp. NE-)pattern languages*, i.e. the set $\{L_{\mathrm{E}}(\alpha) \mid \alpha \in \mathbb{N}^+\}$ resp. $\{L_{\mathrm{NE}}(\alpha) \mid \alpha \in \mathbb{N}^+\}$, considered in this paper merely covers a special case which, in literature, usually is referred to as *terminal-free* (or: *pure*) pattern languages. This is due to the fact that, contrary to our view, a pattern commonly is seen as a string in $(\mathbb{N} \cup \Sigma)^+$ and not just in $\mathbb{N}^+$.

We conclude the definitions in this section with a crucial partition of the set of all patterns subject to the following criterion:

Definition 1. *We call any $\alpha \in \mathbb{N}^+$* succinct *if and only if there exists no decomposition $\alpha = \beta_0\,\gamma_1\,\beta_1\,\gamma_2\,\beta_2\,\ldots\,\beta_{n-1}\,\gamma_n\,\beta_n$ with $n \geq 1$, $\beta_k \in \mathbb{N}^*$ and $\gamma_k \in \mathbb{N}^+$, $k \leq n$, such that*

1. *for every k, $1 \leq k \leq n$, $|\gamma_k| \geq 2$,*
2. *for every k, $1 \leq k \leq n$, and for every k', $0 \leq k' \leq n$, $\mathrm{var}(\gamma_k) \cap \mathrm{var}(\beta_{k'}) = \emptyset$,*
3. *for every k, $1 \leq k \leq n$, there exists an $i_k \in \mathrm{var}(\gamma_k)$ such that $|\gamma_k|_{i_k} = 1$ and, for every k', $1 \leq k' \leq n$, if $i_k \in \mathrm{var}(\gamma_{k'})$ then $\gamma_k = \gamma_{k'}$.*

We call $\alpha \in \mathbb{N}^+$ prolix *if and only if it is not succinct.*

Example 1. Obviously, any pattern α, $|\alpha| \geq 2$, necessarily is prolix if there is a variable $i \in \mathbb{N}$ such that $|\alpha|_i = 1$. Our initial example $\alpha' = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 1 \cdot 4 \cdot 3 \cdot 2$ and the pattern $\alpha_1 = 1 \cdot 1$ are succinct, whereas $\alpha_2 = 1 \cdot 2 \cdot 3 \cdot 3 \cdot 1 \cdot 2 \cdot 3$ is prolix with $\beta_0 = \varepsilon$, $\gamma_1 = 1 \cdot 2$, $\beta_1 = 3 \cdot 3$, $\gamma_2 = 1 \cdot 2$ and $\beta_2 = 3$. Note that this obligatory decomposition of a prolix pattern does not have to be unique. Additional and more complex examples can be found in the subsequent sections and in [13].

According to Reidenbach [13] the succinct patterns are the shortest generators for their respective E-pattern language – this explains the terms "succinct" and "prolix". In other words, for every succinct pattern α and for every pattern β, if $L_{\mathrm{E}}(\alpha) = L_{\mathrm{E}}(\beta)$ then $|\alpha| \leq |\beta|$. Consequently, the class of E-pattern languages equals the set $\{L_{\mathrm{E}}(\alpha) \mid \alpha \in \mathbb{N}^+,\ \alpha \text{ is succinct}\}$ although the set of all patterns is a proper superset of the set of all succinct patterns.

In addition to this view, the set of prolix patterns exactly corresponds to the class of finite *fixed points* of nontrivial morphisms, i.e. for every prolix pattern α there exists a morphism $\phi : \mathbb{N}^* \longrightarrow \mathbb{N}^*$ such that, for an $i \in \mathrm{var}(\alpha)$, $\phi(i) \neq i$ and yet $\phi(\alpha) = \alpha$ (cf., e.g., Head [4], Levé and Richomme [7]).

Finally note that all results in this paper hold for morphisms to arbitrary finitely generated free monoids with three or more generators instead of $\{\mathtt{a}, \mathtt{b}\}^*$ as well. With regard to the positive results, this follows by definition, and the proofs of the negative results can be adapted with little effort.

3 Weakly Unambiguous Words

We begin our examination with some momentuous statements on weakly unambiguous morphic images. The first is an evident yet strong positive result:

Proposition 1. *There is a nonerasing morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ such that, for every $\alpha \in \mathbb{N}^+$, $\sigma(\alpha)$ is weakly unambiguous.*

Proof. For every $i \in \mathbb{N}$, let $|\sigma(i)| = 1$. Then, for every $\alpha \in \mathbb{N}^+$, $|\sigma(\alpha)| = |\alpha|$ and, consequently, $\sigma(\alpha)$ is weakly unambiguous. □

In Proposition 1 we fully restrict ourselves to nonerasing morphisms, and therefore the view applied therein exactly corresponds to the concept of NE-pattern languages. Indeed, the weak unambiguity of the words referred to in the proof is of major importance for inductive inference of NE-pattern languages: Due to the fact given in Proposition 1, for every NE-pattern language L a pattern α with $L = L_{\mathrm{NE}}(\alpha)$ can be inferred from the set of all of the shortest words in this language (shown by Lange and Wiehagen [6]). With regard to E-pattern languages, however, this is provably impossible since, in general, these words are not strongly unambiguous (cf. Reidenbach [12]). Consequently, in respect of pattern inference, the unambiguity of certain words – which are not generated by an injective morphism – is surprisingly powerful.

For the main goal of our approach (see Section 1), however, injectivity of morphisms is vital. Unfortunately, for those morphisms the outcome significantly differs from Proposition 1:

Theorem 1. *There is no injective morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ such that, for every $\alpha \in \mathbb{N}^+$, $\sigma(\alpha)$ is weakly unambiguous.*

Proof. Assume to the contrary that there is such a morphism σ. Since σ is injective, $\sigma(\alpha) \neq \sigma(\beta)$ for every $\alpha \neq \beta$. In particular, this implies $\sigma(i) \neq \sigma(i')$ for every $i, i' \in \mathbb{N}$ with $i \neq i'$. Hence, there must be a $j \in \mathbb{N}$ with $\sigma(j) = w_1 w_2$ for some $w_1, w_2 \in \{\mathtt{a}, \mathtt{b}\}^+$. Now, for an arbitrary $j' \neq j$, let $\alpha := j \cdot j'$. Then, for the morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ given by $\rho(j) := w_1$ and $\rho(j') := w_2\, \sigma(j')$, $\rho(\alpha) = \sigma(\alpha)$, and, thus, $\sigma(\alpha)$ is ambiguous. This contradicts the assumption. □

Obviously, Theorem 1 includes the analogous result for strong unambiguity. Consequently, there is no single injective morphism which, when applied to arbitrary patterns, leads to unambiguous words. Thus, two natural questions arise from Theorem 1: Is there a significant subclass of all patterns for which the opposite of Theorem 1 holds true? Is there at least a way to find for every pattern an individual injective morphism that leads to an unambiguous word? In the following section we examine these questions with regard to strong unambiguity.

4 Strongly Unambiguous Words

Bearing the consequences of Theorem 1 in mind the present section deals with strongly unambiguous words. We begin with the observation that the example

pattern α in the proof of Theorem 1 is prolix. However, if we focus on succinct patterns then the analogue turns out to be true; as we now ask for strong unambiguity we even can prove the opposite of Proposition 1:

Proposition 2. *There is no nonerasing morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^+$ such that, for every succinct $\alpha \in \mathbb{N}^+$, $\sigma(\alpha)$ is strongly unambiguous.*

Proof. Assume to the contrary that there is such a morphism. Then there exist $j, j' \in \mathbb{N}$, $j \neq j'$, and an $\mathtt{c} \in \{\mathtt{a},\mathtt{b}\}$ with $\sigma(j) = v\,\mathtt{c}$ and $\sigma(j') = v'\,\mathtt{c}$, $v, v' \in \Sigma^*$. For some $k, k' \in \mathbb{N}$, $k \neq k'$, $j \neq k \neq j'$ and $j \neq k' \neq j'$, we then regard the pattern $\alpha := j \cdot k \cdot j \cdot k' \cdot j' \cdot k \cdot j' \cdot k'$. Obviously, α is succinct. Now consider the morphism ρ, given by $\rho(j) := v$, $\rho(j') := v'$, $\rho(k) := \mathtt{c}\,\sigma(k)$, $\rho(k') := \mathtt{c}\,\sigma(k')$. Then evidently $\sigma(\alpha) = \rho(\alpha)$, but, e.g., $\sigma(j) \neq \rho(j)$. This is a contradiction. □

Consequently, for every succinct pattern, it is necessary to give an individual injective morphism that leads to a strongly unambiguous word – provided that such a morphism exists. The hope for a positive answer to this question is supported by the following fact whereby, for many patterns, even completely inappropriate looking morphisms generate a strongly unambiguous word.

Proposition 3. *For every nonerasing morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^*$ there exists a succinct $\alpha \in \mathbb{N}^+$, $|\operatorname{var}(\alpha)| \geq 2$, such that $\sigma(\alpha)$ is strongly unambiguous.*

Proof. Since our argumentation solely deals with the length of the morphic images of the variables, we can utilise the following fact on linear combinations:

Claim 1. For all $p, q \in \mathbb{N}$ there exist $r, s \in \mathbb{N}$, $r, s \geq 2$ such that there are no $p', q' \in \mathbb{N}_0 \setminus \{p, q\}$ satisfying $rp + sq = rp' + sq'$.

With $r > q$, $s > p$, and $\gcd(r, s) = 1$, Claim 1 can be proven with a bit of effort.

Now, for some $i, j \in \mathbb{N}$, $i \neq j$, let $p := |\sigma(i)|$, $q := |\sigma(j)|$. Furthermore, let $\alpha := i^r \cdot j^s$ with r, s derived from Claim 1. Obviously, α is succinct. Assume to the contrary that there is a morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^*$ with $\rho(\alpha) = \sigma(\alpha)$ and, for some $k \in \{i, j\}$, $\rho(k) \neq \sigma(k)$. Then ρ must satisfy $|\rho(i)| \neq |\sigma(i)|$, $|\rho(j)| \neq |\sigma(j)|$ and $|\rho(\alpha)| = |\sigma(\alpha)|$. Consequently, with $p' := |\rho(i)|$, $q' := |\rho(j)|$, $rp + sq = |\sigma(\alpha)| = |\rho(\alpha)| = rp' + sq'$. This contradicts Claim 1. □

Before we go further into this matter of strongly unambiguous morphic images for succinct patterns (see Theorem 3), we turn our attention to prolix patterns. Here we can easily give a definite answer, which, alternatively, can be seen as a consequence of the fact that every prolix pattern is a fixed point of some nontrivial morphism (cf. Section 2):

Theorem 2. *For any prolix $\alpha \in \mathbb{N}^+$ and for any nonerasing morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a},\mathtt{b}\}^+$, $\sigma(\alpha)$ is not strongly unambiguous.*

Proof. Assume to the contrary that there are a prolix pattern α and a nonerasing morphism σ such that $\sigma(\alpha)$ is strongly unambiguous. Then, as α is prolix, there exists a decomposition $\alpha = \beta_0\gamma_1\beta_1\gamma_2\beta_2 \ \ldots \ \beta_{n-1}\gamma_n\beta_n$ satisfying the conditions

of Definition 1. With regard to this decomposition and for every k, $1 \leq k \leq n$, let i_k be the smallest $i \in \mathrm{var}(\gamma_k)$ such that $|\gamma_k|_{i_k} = 1$ and, for every k', $1 \leq k' \leq n$, if $i_k \in \mathrm{var}(\gamma_{k'})$, then $\gamma_k = \gamma_{k'}$. By definition, for every γ_k, $1 \leq k \leq n$, such an i_k exists, and, for every $\beta_{k'}$, $0 \leq k' \leq n$, $i_k \notin \mathrm{var}(\beta_{k'})$. Now we define ρ as follows: For all k, $1 \leq k \leq n$, $\rho(i_k) := \sigma(\gamma_k)$, for all $i \in \mathrm{var}(\gamma_k) \setminus \{i_k\}$, $\rho(i) := \varepsilon$, and, for all k, $0 \leq k \leq n$ and for all $i \in \mathrm{var}(\beta_k)$, $\rho(i) := \sigma(i)$. Then $\sigma(\alpha) = \rho(\alpha)$, but, since σ is nonerasing and every γ_k contains at least two variables, there is an $i \in \mathrm{var}(\alpha)$ with $\sigma(i) \neq \rho(i)$. Thus, $\sigma(\alpha)$ is not strongly unambiguous. □

Thus, for every prolix pattern there is no strongly unambiguous word at all – at least as long as we restrict ourselves to the images of nonerasing morphisms. If this requirement is omitted then we face a fairly intricate situation:

Example 2. Let $\alpha_1 := 1 \cdot 1 \cdot 2 \cdot 2 \cdot 3 \cdot 4 \cdot 3 \cdot 4$, $\alpha_2 := 1 \cdot 2 \cdot 2 \cdot 1 \cdot 3 \cdot 4 \cdot 3 \cdot 4$, and $\beta_1 := 1 \cdot 2 \cdot 3 \cdot 3 \cdot 1 \cdot 2$, $\beta_2 := 1 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 1 \cdot 2 \cdot 2$. The patterns are prolix. For α_1 and β_1 there is no morphism σ such that $\sigma(\alpha_1)$ or $\sigma(\beta_1)$ are unambiguous. Contrary to this, for α_2 and β_2 there exist suitable words such as $\mathtt{abba}$ and $\mathtt{baab}$.

As shown in Example 2, there are prolix patterns for which we can unambiguously map certain subpatterns onto strings in $\{\mathtt{a}, \mathtt{b}\}^*$, whereas for different, quite similar appearing patterns this is impossible. Furthermore, these subpatterns can consist of parts of some β_k as well as parts of some γ_k in the required decomposition of the patterns (cf. Definition 1). We now briefly discuss this phenomenon, and we begin with a criterion which covers both prolix and succinct patterns:

Condition 1. *A pattern $\alpha \in \mathbb{N}^+$ satisfies Condition 1 if and only if there exists an $i \in \mathrm{var}(\alpha)$ such that, for $n = |\mathrm{var}(\alpha)| - 1$, for all $j_1, j_2, \ldots, j_n \in \mathrm{var}(\alpha) \setminus \{i\}$ and for all $k_1, k_2, \ldots, k_n \in \mathbb{N}_0$, $|\alpha|_i \neq k_1|\alpha|_{j_1} + k_2|\alpha|_{j_2} + \ldots + k_n|\alpha|_{j_n}$.*

For those patterns satisfying Condition 1 we can give a positive result:

Proposition 4. *For every $\alpha \in \mathbb{N}^+$ satisfying Condition 1 there exists a morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ such that $\sigma(\alpha)$ is strongly unambiguous.*

Proof. With $\sigma(i) := \mathtt{a}$ (i as defined in Condition 1) and, for all $j \in \mathbb{N}$ with $j \neq i$, $\sigma(j) := \varepsilon$, Proposition 4 follows immediately. □

For prolix patterns with exactly two different variables, Condition 1 even characterises the subclass for which there are strongly unambiguous words:

Proposition 5. *Let $\alpha \in \mathbb{N}^+$ be prolix, $\mathrm{var}(\alpha) := \{i, j\}$. Then there exists a morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ such that $\sigma(\alpha)$ is strongly unambiguous if and only if $|\alpha|_i \neq |\alpha|_j$.*

Proof. For the *if* part, w. l. o. g. assume $|\alpha|_i < |\alpha|_j$. Then the existence of a morphism σ such that $\sigma(\alpha)$ is strongly unambiguous is guaranteed by Proposition 4. We proceed with the *only if* part: Let $|\alpha|_i = |\alpha|_j$. Then, since α is prolix, α can only be of the form $(i \cdot j)^n$ (or $(j \cdot i)^n$), $n \in \mathbb{N}$. Thus, there is no strongly unambiguous morphic image for α. □

Note that Propositions 4 and 5 utilise a morphism that is non-empty for a single variable only. In general, of course, one might wish to find a morphism that assigns non-empty words to a preferably large number of variables in a prolix pattern and, nevertheless, leads to a strongly unambiguous word (cf. Example 2). However, as soon as the number of variables to be mapped onto non-empty words exceeds the number of letters in the target alphabet, we consider it an extraordinarily challenging problem to find reasonably strong criteria.

We now return to the remaining crucial question of this paper left open after Propositions 2 and 3 and Theorem 2, namely the existence of *injective* morphisms generating strongly unambiguous words for *succinct* patterns. Particularly the proof of Proposition 2 suggests that a finitely generated free monoid might not be rich enough to include strongly unambiguous morphic images for all succinct patterns. On the other hand, the proof of the comprehensive negative result for prolix patterns (cf. Theorem 2) strongly utilises the properties of these patterns as declared in Definition 1, and, indeed, our main result (to be proven in Section 4.1) shows the opposite of Theorem 2 to be true for succinct patterns:

Theorem 3. *For every succinct $\alpha \in \mathbb{N}^+$, there is an injective morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ such that $\sigma(\alpha)$ is strongly unambiguous.*

Consequently, for every succinct string in an infinitely generated free semigroup there is a morphic image in a free monoid with two generators that – in accordance with our requirements explained in Section 1 – sufficiently preserves its structure. With regard to pattern languages, Theorem 3 proves that in every E-pattern language there is an unambiguous word with respect to any shortest generating pattern. For a restatement of our main result in terms of equality sets or fixed points of morphisms, see the notes in Section 1 or Section 2, respectively.

Finally, we can use Theorems 2 and 3 for a characterisation of succinctness:

Corollary 1. *Let $\alpha \in \mathbb{N}^+$. Then α is succinct if and only if there exists an injective morphism $\sigma : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^+$ such that $\sigma(\alpha)$ is strongly unambiguous.*

4.1 Proof of Theorem 3

We begin this section with a procedure which, for every succinct pattern, constructs a morphism that generates a strongly unambiguous word:

Definition 2. *Let $\alpha \in \mathbb{N}^+$. For every $j \in \mathrm{var}(\alpha)$, consider the following sets: $L_j := \{k \mid \alpha = \ldots \cdot k \cdot j \cdot \ldots\}$ and $R_j := \{k \mid \alpha = \ldots \cdot j \cdot k \cdot \ldots\}$. Thus, L_j consists of all "left neighbours" of j in α and R_j of all "right neighbours". With these sets, construct two relations $\sim_l$ and $\sim_r$ on $\mathrm{var}(\alpha)$: For all $k, k' \in \mathrm{var}(\alpha)$*

- *$k \sim_l k'$ if and only if there are $j_1, j_2, \ldots, j_t \in \mathrm{var}(\alpha)$, $t \geq 1$, such that*
  1. *$L_{j_1} \cap L_{j_2} \neq \emptyset$, $L_{j_2} \cap L_{j_3} \neq \emptyset$, …, $L_{j_{t-1}} \cap L_{j_t} \neq \emptyset$ and*
  2. *$k \in L_{j_1}$ and $k' \in L_{j_t}$.*
- *$k \sim_r k'$ if and only if there are $j_1, j_2, \ldots, j_s \in \mathrm{var}(\alpha)$, $s \geq 1$, such that*
  1. *$R_{j_1} \cap R_{j_2} \neq \emptyset$, $R_{j_2} \cap R_{j_3} \neq \emptyset$, …, $R_{j_{s-1}} \cap R_{j_s} \neq \emptyset$ and*
  2. *$k \in R_{j_1}$ and $k' \in R_{j_s}$.*

Evidently, $\sim_l$ and $\sim_r$ are equivalence relations, and, for every $k \in \mathrm{var}(\alpha)$, there exist equivalence classes $L^{\sim}$ and $R^{\sim}$ with $k \in L^{\sim}$ and $k \in R^{\sim}$. Given in arbitrary order each, let $L_1^{\sim}, L_2^{\sim}, \ldots, L_p^{\sim}$ be all equivalence classes resulting from $\sim_l$ and $R_1^{\sim}, R_2^{\sim}, \ldots, R_q^{\sim}$ all equivalence classes resulting from $\sim_r$. Consequently, $L_1^{\sim} \cup L_2^{\sim} \cup \ldots \cup L_p^{\sim}$ and $R_1^{\sim} \cup R_2^{\sim} \cup \ldots \cup R_q^{\sim}$ are two disjoint partitions of $\mathrm{var}(\alpha)$ induced by $\sim_l$ and $\sim_r$. Then, for $i \in \{1, 2, \ldots, p\}$, $i' \in \{1, 2, \ldots, q\}$ and for every $k \in \mathrm{var}(\alpha)$, the morphism $\sigma_\alpha^{\mathrm{su}}$ is given by

$$\sigma_\alpha^{\mathrm{su}}(k) := \begin{cases} \mathtt{a}\,\mathtt{b}^{3k}\,\mathtt{a}\;\mathtt{a}\,\mathtt{b}^{3k+1}\,\mathtt{a}\;\mathtt{a}\,\mathtt{b}^{3k+2}\,\mathtt{a}\,, & \nexists\, i : \; k = \min L_i^{\sim} \;\wedge\; \nexists\, i' : \; k = \min R_{i'}^{\sim}, \\ \mathtt{b}\,\mathtt{a}^{3k}\,\mathtt{b}\;\mathtt{a}\,\mathtt{b}^{3k+1}\,\mathtt{a}\;\mathtt{a}\,\mathtt{b}^{3k+2}\,\mathtt{a}\,, & \nexists\, i : \; k = \min L_i^{\sim} \;\wedge\; \exists\, i' : \; k = \min R_{i'}^{\sim}, \\ \mathtt{a}\,\mathtt{b}^{3k}\,\mathtt{a}\;\mathtt{a}\,\mathtt{b}^{3k+1}\,\mathtt{a}\;\mathtt{b}\,\mathtt{a}^{3k+2}\,\mathtt{b}\,, & \exists\, i : \; k = \min L_i^{\sim} \;\wedge\; \nexists\, i' : \; k = \min R_{i'}^{\sim}, \\ \mathtt{b}\,\mathtt{a}^{3k}\,\mathtt{b}\;\mathtt{a}\,\mathtt{b}^{3k+1}\,\mathtt{a}\;\mathtt{b}\,\mathtt{a}^{3k+2}\,\mathtt{b}\,, & \exists\, i : \; k = \min L_i^{\sim} \;\wedge\; \exists\, i' : \; k = \min R_{i'}^{\sim}. \end{cases}$$

Obviously, for every $\alpha \in \mathbb{N}^+$, $\sigma_\alpha^{\mathrm{su}}$ is injective.

As an illustration of Definition 2 we now identify $\sigma_\alpha^{\mathrm{su}}$ for an example pattern:

Example 3. Let $\alpha := 1 \cdot 2 \cdot 3 \cdot 1 \cdot 2 \cdot 4 \cdot 3 \cdot 1 \cdot 2 \cdot 3 \cdot 2 \cdot 4$. Evidently, α is succinct (cf. Definition 1). Then $L_1 = \{3\}$, $L_2 = \{1, 3\}$, $L_3 = \{2, 4\}$, $L_4 = \{2\}$ and $R_1 = \{2\}$, $R_2 = \{3, 4\}$, $R_3 = \{1, 2\}$, $R_4 = \{3\}$. This leads to $L_1^{\sim} = \{1, 3\}$, $L_2^{\sim} = \{2, 4\}$ and $R_1^{\sim} = \{1, 2\}$, $R_2^{\sim} = \{3, 4\}$, and, thus, $\sigma_\alpha^{\mathrm{su}}(1) = \mathtt{b} \ldots \mathtt{b}$, $\sigma_\alpha^{\mathrm{su}}(2) = \mathtt{a} \ldots \mathtt{b}$, $\sigma_\alpha^{\mathrm{su}}(3) = \mathtt{b} \ldots \mathtt{a}$ and $\sigma_\alpha^{\mathrm{su}}(4) = \mathtt{a} \ldots \mathtt{a}$.

Note that, for the pattern in Example 3, the injective morphism $\sigma'(i) = \mathtt{a}\,\mathtt{b}^i$, $i \in \mathbb{N}$, generates an unambiguous word as well, and even the non-injective morphism σ'' given by $\sigma''(2) := \mathtt{b}$, $\sigma''(1) = \sigma''(3) = \sigma''(4) := \mathtt{a}$ has this property. Additionally, for every pattern α satisfying, for some $n \geq 1$ and $p_1, p_2, \ldots, p_n \geq 2$, $\alpha = 1^{p_1} \cdot 2^{p_2} \cdot \ldots \cdot n^{p_n}$, $\sigma'(\alpha)$ is known to be strongly unambiguous (cf. Reidenbach [12]). Thus, $\sigma_\alpha^{\mathrm{su}}$ is "sufficient" for generating an unambiguous word (as to be shown in the subsequent lemmata), but, in general, it is not "necessary" since there can be significantly shorter words with the desired property. Contrary to this, for our initial example α', it is obviously necessary to give a morphism which is more sophisticated than σ' (cf. Section 1).

As the underlying principles of both Definition 2 and the subsequent lemmata are fairly complex we now briefly discuss the line of reasoning in this section: The basic idea for Definition 2 is derived from the proof of Proposition 2. Therein, we can observe that, for the abstract example pattern, the ambiguity of the regarded word is caused by the fact that, for all of the left neighbours of some variables (i.e., in terms of Definition 2, for some L_i), the morphic images end with the same letter. We call an L_i *(morphically) homogeneous* (*with respect to* a morphism σ) if it shows such a property. Thus, it seems reasonable to choose a morphism such that in each L_i with $|L_i| \geq 2$ there are two variables whose morphic images end with different letters (or, in other words, convert L_i into a *(morphically) heterogeneous* set), but this idea may lead to conflicting assignments:

Example 4. Let $\alpha := 1 \cdot 2 \cdot 3 \cdot 2 \cdot 1 \cdot 3 \cdot 1$. Thus, $L_1 = \{2, 3\}$, $L_2 = \{1, 3\}$ and $L_3 = \{1, 2\}$. Then, for a binary alphabet, there is no morphism σ such that, at a time, L_1, L_2 and L_3 are morphically heterogeneous with respect to σ.

Fortunately, a thorough combinatorial consideration shows that it suffices to guarantee heterogeneity of each $L_i^{\sim}$ (cf. Lemma 2); as these sets are disjoint, this avoids any contradictory assignments. Note that these statements analogously hold for R_i and $R_i^{\sim}$ (regarding the first letter of the morphic images of the variables in these sets instead of the last one).

Before we formally analyse the consequences of morphic heterogeneity we now address the injectivity of $\sigma_\alpha^{\mathrm{su}}$. Of course, according to our goal of finding a structure-preserving morphic image, we have to choose injective morphisms; in addition, however, we can observe that non-injectivity can cause ambiguity:

Example 5. Let $\alpha := 1\cdot2\cdot1\cdot3\cdot2\cdot1\cdot4\cdot3\cdot4\cdot5\cdot6\cdot7\cdot6\cdot5\cdot6\cdot7\cdot1\cdot7\cdot2$. This pattern is succinct, and $L_1^{\sim} = \mathrm{var}(\alpha)$, $R_1^{\sim} = \mathrm{var}(\alpha)$. Then, for $L_1^{\sim}$ and $R_1^{\sim}$, the non-injective morphism σ given by $\sigma(1) := \mathtt{b}$ and $\sigma(i) := \mathtt{a}$, $i \in \mathrm{var}(\alpha) \setminus \{1\}$ leads to the desired heterogeneity. Nevertheless, there is a morphism ρ with $\rho(\alpha) = \sigma(\alpha)$ and $\rho \neq \sigma$, namely $\rho(4) := \varepsilon$, $\rho(5) := \mathtt{a}\,\mathtt{a}$ and $\rho(i) := \sigma(i)$, $i \in \mathrm{var}(\alpha) \setminus \{4, 5\}$.

The injectivity of $\sigma_\alpha^{\mathrm{su}}$ is brought about by the assignment of three unique *segments* $\mathtt{c}\,\mathtt{d}^m\,\mathtt{c}$, $\mathtt{c}, \mathtt{d} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, $m \in \mathbb{N}$, to each variable. This allows to prove the following phenomenon, which, in a similar way, has been examined in [13]:

Lemma 1. *Let $\alpha \in \mathbb{N}^+$ be succinct. Then, for every morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ with $\rho(\alpha) = \sigma_\alpha^{\mathrm{su}}(\alpha)$ and for every $i \in \mathrm{var}(\alpha)$, $\rho(i) = \ldots\, \mathtt{g}\,\mathtt{a}\,\mathtt{b}^{3i+1}\,\mathtt{a}\,\mathtt{h}\,\ldots$, $\mathtt{g}, \mathtt{h} \in \{\mathtt{a}, \mathtt{b}\}$.*

Lemma 1 requires an extensive reasoning. Due to space constraints, we omit the proof and refer the reader to Lemma 1 in [13], which can give a rough idea of it.

We conjecture that strong unambiguity can also be ensured by a morphism which, for every variable $i \in \mathbb{N}$, assigns only the first and the last segment of $\sigma_\alpha^{\mathrm{su}}(i)$ to i, but, in this case, we expect the proof of the equivalent to the following lemma to be significantly more difficult.

As explained above, we now conclude the proof of our main result with the examination of the use of morphic heterogeneity:

Lemma 2. *Let $\alpha \in \mathbb{N}^+$ be succinct. If, for every morphism $\rho : \mathbb{N}^+ \longrightarrow \{\mathtt{a}, \mathtt{b}\}^*$ with $\rho(\alpha) = \sigma_\alpha^{\mathrm{su}}(\alpha)$ and for every $i \in \mathrm{var}(\alpha)$, $\rho(i) = \ldots\, \mathtt{a}\,\mathtt{b}^{3i+1}\,\mathtt{a}\,\ldots$ then $\sigma_\alpha^{\mathrm{su}}(\alpha)$ is strongly unambiguous.*

Proof. If $|\,\mathrm{var}(\alpha)| = 1$ then every morphic image of α is strongly unambiguous, and therefore, in this case, Lemma 2 holds trivially. Hence, let $|\,\mathrm{var}(\alpha)| \geq 2$. We start the proof with a small remark that is needed at several stages of the proof:

Claim 1. For every $i \in \mathrm{var}(\alpha)$, $\mathtt{c}, \mathtt{d} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, and $y \in \{0, 1, 2\}$, $\rho(i) \neq \ldots\, \mathtt{c}\,\mathtt{d}^{3i+y}\,\mathtt{c}\,\ldots\,\mathtt{c}\,\mathtt{d}^{3i+y}\,\mathtt{c}\,\ldots$.

Proof (Claim 1). Claim 1 directly follows from the precondition $\rho(\alpha) = \sigma_\alpha^{\mathrm{su}}(\alpha)$ since, obviously, $|\,\sigma_\alpha^{\mathrm{su}}(\alpha)|_{\mathtt{c}\,\mathtt{d}^{3i+y}\,\mathtt{c}} = |\alpha|_i$. □(*Claim 1*)

Now assume to the contrary that there is a morphism ρ with $\rho(\alpha) = \sigma_\alpha^{\mathrm{su}}(\alpha)$ such that, for every $i \in \mathrm{var}(\alpha)$, $\rho(i) = \ldots\, \mathtt{a}\,\mathtt{b}^{3i+1}\,\mathtt{a}\,\ldots$ and, for some $i' \in \mathrm{var}(\alpha)$, $\rho(i') \neq \sigma_\alpha^{\mathrm{su}}(i')$. Then there necessarily is a $j \in \mathrm{var}(\alpha)$ such that, for some $\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, $\mathtt{e} \neq \mathtt{f}$,

(a) $\rho(j) = \ldots \mathtt{g}\,\mathtt{c}\,\mathtt{d}^{3j}\,\mathtt{c}\,\mathtt{a}\,\mathtt{b}^{3j+1}\,\mathtt{a} \ldots$ or
(b) $\rho(j) = \ldots \mathtt{a}\,\mathtt{b}^{3j+1}\,\mathtt{a}\,\mathtt{e}\,\mathtt{f}^{3j+2}\,\mathtt{e}\,\mathtt{h} \ldots$.

We restrict the following reasoning to case (a) since an analogous argumentation can be applied to case (b) (using $\sim_r$ instead of $\sim_l$): Note that, due to $|\operatorname{var}(\alpha)| \geq 2$ and the succinctness of α, $|\alpha|_j \geq 2$ and therefore $L_j \neq \emptyset$ (for the definition of L_j, see Definition 2). Hence, let k be an arbitrary variable in L_j. Consequently, for any $\mathtt{c}, \mathtt{d} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, $\rho(k) \neq \ldots\, \mathtt{c}\,\mathtt{d}^{3k+2}\,\mathtt{c}$.

Case 1: $\alpha = j \ldots$.
Then we can directly follow from Claim 1: $\sigma^{\mathrm{su}}_{\alpha}(\alpha) = \mathtt{e}\,\mathtt{f}^{3j}\,\mathtt{e} \ldots \neq \rho(\alpha) = \ldots\, \mathtt{g}\,\mathtt{c}\,\mathtt{d}^{3j}\,\mathtt{c} \ldots$, with $\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, $\mathtt{e} \neq \mathtt{f}$. This contradicts the condition $\sigma^{\mathrm{su}}_{\alpha}(\alpha) = \rho(\alpha)$.

Case 2: $\alpha = \ldots\, k$.
Then $\sigma^{\mathrm{su}}_{\alpha}(\alpha) = \ldots\, \mathtt{c}\,\mathtt{d}^{3k+2}\,\mathtt{c} \neq \rho(\alpha)$ since $\rho(k) \neq \ldots\, \mathtt{c}\,\mathtt{d}^{3k+2}\,\mathtt{c}$ for $\mathtt{c}, \mathtt{d} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$. This again contradicts the condition $\sigma^{\mathrm{su}}_{\alpha}(\alpha) = \rho(\alpha)$.

Case 3: $\alpha \neq j \ldots$ and $\alpha \neq \ldots\, k$.
For the equivalence classes $L_1^{\sim}, L_2^{\sim}, \ldots, L_p^{\sim}$ derived from the construction of $\sigma^{\mathrm{su}}_{\alpha}$, let $\iota \in \{1, 2, \ldots, p\}$ with $L_j \subseteq L_{\iota}^{\sim}$. Then, since all $L_1^{\sim}, L_2^{\sim}, \ldots, L_p^{\sim}$ are pairwise disjoint, this ι is unique. Now we can collect a number of facts that facilitate the argumentation in Case 3. The first holds as α is succinct:

Claim 2. If $\alpha \neq j \ldots$ and $\alpha \neq \ldots\, k$ then $|L_{\iota}^{\sim}| \geq 2$.

Proof (Claim 2). If $|L_j| \geq 2$ then Claim 2 holds trivially. Hence, let $L_j = \{k\}$. Then, for every occurrence of j in α, the conditions $\alpha \neq j \ldots$ and $|\alpha|_j \geq 2$ lead to $\alpha = \ldots\, k \cdot j \ldots$. Thus, due to the succinctness of α, there are some $j_1, j_2, \ldots, j_m \in \operatorname{var}(\alpha)$, $m \geq 2$, with $\alpha = \ldots\, k \cdot j_r \ldots$, $1 \leq r \leq m$. Additionally, because of $\alpha \neq \ldots\, k$, there must be an $s \in \{1, 2, \ldots, m\}$ and a $\bar{k} \in \operatorname{var}(\alpha)$, $\bar{k} \neq k$, with $\alpha = \ldots\, \bar{k} \cdot j_s \ldots$, since, otherwise, α would either be prolix or start with a j_r, $r \in \{1, 2, \ldots, m\}$, leading to the same argumentation as in Case 1. Consequently, $L_{j_s} \supseteq \{k, \bar{k}\}$ and therefore $L_j \subset \{k, \bar{k}\} \subseteq L_{j_s} \subseteq L_{\iota}^{\sim}$. $\square$(*Claim 2*)

Now, for every $L^{\sim}$ among $L_1^{\sim}, L_2^{\sim}, \ldots, L_p^{\sim}$, the next fact follows by definition since these equivalence classes are composed by union of *non-disjoint* sets (cf. Definition 2 and, e.g., Example 3):

Claim 3. If $|L^{\sim}| \geq 2$ then, for every $\hat{k} \in L^{\sim}$, there is an $L_{\hat{\jmath}} \subseteq L^{\sim}$ with $|L_{\hat{\jmath}}| \geq 2$ and $\hat{k} \in L_{\hat{\jmath}}$.

We conclude the list of preliminary claims with the following one, that deals with a crucial phenomenon which is reflected in the transitivity of $\sim_l$:

Claim 4. For every $\hat{k} \in L_{\iota}^{\sim}$ and any $\mathtt{e}, \mathtt{f} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{e} \neq \mathtt{f}$, $\rho(\hat{k}) \neq \ldots\, \mathtt{e}\,\mathtt{f}^{3\hat{k}+2}\,\mathtt{e} \ldots$.

Proof (Claim 4). With regard to any $\hat{k}' \in L_j \subseteq L_{\iota}^{\sim}$, Claim 4 holds because of the precondition $\rho(i) = \ldots\, \mathtt{a}\,\mathtt{b}^{3i+1}\,\mathtt{a} \ldots$, $i \in \operatorname{var}(\alpha)$, because of Claim 1 and the fact that, for some $\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f} \in \{\mathtt{a}, \mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$, $\mathtt{e} \neq \mathtt{f}$, $\rho(j) = \ldots\, \mathtt{e}\,\mathtt{c}\,\mathtt{d}^{3j}\,\mathtt{c}\,\mathtt{a}\,\mathtt{b}^{3j+1}\,\mathtt{a} \ldots$ and

$$\rho(\hat{k}' \cdot j) = \ldots\, \mathtt{a}\,\mathtt{b}^{3\hat{k}'+1}\,\mathtt{a}\,\mathtt{e}\,\mathtt{f}^{3\hat{k}'+2}\,\mathtt{e}\,\mathtt{c}\,\mathtt{d}^{3j}\,\mathtt{c}\,\mathtt{a}\,\mathtt{b}^{3j+1}\,\mathtt{a} \ldots .$$

We now regard all $\hat{k}'' \in L_\iota^\sim$ for which there is an $L_{j'}$ with $\hat{k}', \hat{k}'' \in L_{j'}$ (recall that $\hat{k}' \in L_j$). Then – since Claim 4 is satisfied for $\hat{k}'$ and, consequently, $\rho(j') = \ldots \mathtt{e}\,\mathtt{c}\mathtt{d}^{3j'}\,\mathtt{c}\,\mathtt{a}\mathtt{b}^{3j'+1}\mathtt{a}\ldots$, $\mathtt{c},\mathtt{d},\mathtt{e} \in \{\mathtt{a},\mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$ – Claim 4 holds for these $\hat{k}''$ as well. Now we proceed to all $\hat{k}''' \in L_\iota^\sim$ for which there is an $L_{j''}$ with $\hat{k}'', \hat{k}''' \in L_{j''}$ (recall that $\hat{k}'' \in L_{j'}$). Then, as Claim 4 is satisfied for $\hat{k}''$, Claim 4 holds for all $\hat{k}'''$ and so on. Consequently, according to the construction of $L_\iota^\sim$ (cf. Definition 2) Claim 4 holds for every $\hat{k} \in L_\iota^\sim$. □(*Claim 4*)

We now can conclude our argumentation on Case 3: According to Claim 2, $|L_\iota^\sim| \geq 2$. Let $k_\sharp := \min L_\iota^\sim$; then, due to Claim 3, there is an $j_\sharp \in \mathrm{var}(\alpha)$ with $k_\sharp \in L_{j_\sharp}$ and $|L_{j_\sharp}| \geq 2$. Consequently, let $\bar{k}_\sharp \in \mathrm{var}(\alpha)$ with $k_\sharp \neq \bar{k}_\sharp$ and $\{k_\sharp, \bar{k}_\sharp\} \subseteq L_{j_\sharp}$. Then, because of $k_\sharp = \min L_\iota^\sim$ and $\bar{k}_\sharp \in L_\iota^\sim$, $\sigma_\alpha^{\mathrm{su}}(k_\sharp) = \ldots\,\mathtt{b}$ and $\sigma_\alpha^{\mathrm{su}}(\bar{k}_\sharp) = \ldots\,\mathtt{a}$. Referring to the condition $\rho(i) = \ldots\,\mathtt{a}\mathtt{b}^{3i+1}\,\mathtt{a}\ldots$, $i \in \mathrm{var}(\alpha)$, to Claim 1 and to Claim 4, these different endings of $\sigma_\alpha^{\mathrm{su}}(k_\sharp)$ and $\sigma_\alpha^{\mathrm{su}}(\bar{k}_\sharp)$ imply

$$\ldots\,\boxed{\mathtt{b}}\,\mathtt{c}\,\mathtt{d}^{3j_\sharp}\,\mathtt{c}\,\mathtt{a}\mathtt{b}^{3j_\sharp+1}\,\mathtt{a}\ldots = \rho(j_\sharp) = \ldots\,\boxed{\mathtt{a}}\,\mathtt{c}\,\mathtt{d}^{3j_\sharp}\,\mathtt{c}\,\mathtt{a}\mathtt{b}^{3j_\sharp+1}\,\mathtt{a}\ldots\,,$$

for some $\mathtt{c},\mathtt{d} \in \{\mathtt{a},\mathtt{b}\}$, $\mathtt{c} \neq \mathtt{d}$. This contradicts $\mathtt{a} \neq \mathtt{b}$. □

With Lemma 1 and Lemma 2, Theorem 3 follows immediately.

Acknowledgements. The authors are indebted to Frank Hechler and the referees for numerous useful comments and, in particular, for pointing out the analogy between prolix patterns and fixed points of morphisms.

References

1. D. Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21:46–62, 1980.
2. C. Choffrut and J. Karhumäki. Combinatorics of words. In [14].
3. T. Harju and J. Karhumäki. Morphisms. In [14].
4. T. Head. Fixed languages and the adult languages of 0L schemes. *Intern. J. of Computer Math.*, 10:103–107, 1981.
5. H. Jürgensen and S. Konstantinidis. Codes. In [14].
6. S. Lange and R. Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Comput.*, 8:361–370, 1991.
7. F. Levé and G. Richomme. On a conjecture about finite fixed points of morphisms. *Theor. Comp. Sci.*, to appear.
8. M. Lipponen and G. Păun. Strongly prime PCP words. *Discrete Appl. Math.*, 63:193–197, 1995.
9. M. Lothaire. *Combinatorics on Words.* Addison-Wesley, Reading, MA, 1983.
10. A. Mateescu and A. Salomaa. Patterns. In [14].
11. A. Mateescu and A. Salomaa. Finite degrees of ambiguity in pattern languages. *RAIRO Inform. Théor. Appl.*, 28(3–4):233–253, 1994.
12. D. Reidenbach. A non-learnable class of E-pattern languages. *Theor. Comp. Sci.*, to appear.
13. D. Reidenbach. A discontinuity in pattern inference. In *Proc. STACS 2004*, volume 2996 of *LNCS*, pages 129–140, 2004.
14. G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*, volume 1. Springer, Berlin, 1997.

Complementing Two-Way Finite Automata*

Viliam Geffert[1], Carlo Mereghetti[2], and Giovanni Pighizzini[3]

[1] Department of Computer Science, P. J. Šafárik University
Jesenná 5, 04154 Košice, Slovakia
geffert@kosice.upjs.sk
[2] Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano
via Comelico 39, 20135 Milano, Italy
mereghetti@dsi.unimi.it
[3] Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano
via Comelico 39, 20135 Milano, Italy
pighizzi@dico.unimi.it

Abstract. We study the relationship between the sizes of two-way finite automata accepting a language and its complement. In the deterministic case, by adapting Sipser's method, for a given automaton (2dfa) with n states we build an automaton accepting the complement with at most $4n$ states, independently of the size of the input alphabet. Actually, we show a stronger result, by presenting an equivalent $4n$–state 2dfa that always halts.
For the nondeterministic case, using a variant of inductive counting, we show that the complement of a unary language, accepted by an n–state two-way automaton (2nfa), can be accepted by an $O(n^8)$–state 2nfa. Here we also make the 2nfa halting. This allows the simulation of unary 2nfa's by probabilistic Las Vegas two-way automata with $O(n^8)$ states.

Keywords: automata and formal languages; descriptional complexity.

1 Introduction

Automata theory is one of the oldest topics in computer science. In spite of that, there is a renewed interest in this subject recently. In particular, two aspects of automata theory have been extensively investigated: nonstandard models and descriptional complexity.

Nonstandard models of automata (among others, probabilistic [15], Las Vegas [6], self-verifying [1], and quantum [9, 13]) differ from classical ones in the transition rules and/or in the acceptance conditions.

Descriptional complexity compares formal systems with respect to their conciseness. (For a recent survey, see [4].) Several variants of finite automata are

* This work was partially supported by the Science and Technology Assistance Agency under contract APVT-20-004104, by the Slovak Grant Agency for Science (VEGA) under contract "Combinatorial Structures and Complexity of Algorithms", and by MIUR under the projects FIRB "Descriptional complexity of automata and related structures" and COFIN "Linguaggi formali e automi: metodi, modelli e applicazioni"

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 260–271, 2005.
© Springer-Verlag Berlin Heidelberg 2005

known from the literature (one-way or two-way, deterministic or nondeterministic, ..., see, e.g., [5]). They all have the same computational power. In fact, they characterize the class of regular languages. However, two different models may require a considerably different number of states for the same language. The first widely known result in this sense compares nondeterminism with determinism for one-way finite automata (1nfa versus 1dfa): each n–state 1nfa can be simulated by a 1dfa with 2^n states. Moreover, for each n, there is a language accepted by an n–state 1nfa such that each equivalent 1dfa has at least 2^n states. Thus, in general, we cannot do any better [12, 14].

The corresponding gap for two-way machines (2nfa versus 2dfa), conjectured by Sakoda and Sipser in 1978 [16], is still open. In the unary case, i.e., for automata with a single letter input alphabet, a subexponential simulation of 2nfa's by 2dfa's has been obtained [3].

In this paper, we study the relationship between the sizes of two-way automata accepting a language and its complement. (Related topics for one-way automata have been recently considered in [8, 10]).

In the deterministic case, for a given 2dfa with n states, we show how to build a 2dfa accepting the complement with at most $4n$ states. This improves the known upper bound [18], from $O(n^2)$ to $O(n)$. The construction does not depend on the size of the input alphabet[1]. Actually, our result is stronger: we prove that each n–state 2dfa can be simulated by a 2dfa with $4n$ states that halts on any input.

For the nondeterministic case, we show that the complement of a unary language accepted by an n–state 2nfa, can be accepted by an $O(n^8)$–state 2nfa. The construction is based on a variant of inductive counting [2, 7, 19]. Here we also prove a stronger result, namely, each unary n–state 2nfa can be replaced by an equivalent $O(n^8)$–state self-verifying automaton (2svfa) which halts on any input. (Self-verifying machines are a special case of nondeterministic machines. The complement for a self-verifying automaton can be immediately obtained by exchanging accepting with rejecting states.)

We were not able to resolve the problem of the complement for 2nfa's in the general (nonunary) case. As we will discuss in Section 6, the problem of stating the gap (in terms of the number of states) between 2nfa's and 2nfa's accepting the complement turns out to be harder than the conjecture of Sakoda and Sipser, mentioned above.

As a consequence of our result concerning the complementation of unary 2nfa's, we also state a connection with Las Vegas automata. In particular, we show that unary 2nfa's can be simulated by two-way Las Vegas automata with a polynomial number of states.

[1] In [18], it was pointed out that, besides $O(n^2)$, we can use a modified construction with $O(n \cdot m^2)$ states, where n is the number of states of the original machine and m the size of the input alphabet. This gives a linear upper bound for languages over a *fixed* input alphabet, but not in the general case. (For example, the results presented in [8] consider witness regular languages with the alphabet size growing exponentially in n)

Due to lack of space, the details of the proofs are omitted in this version of the paper.

2 Basic Definitions

Here we briefly recall some basic definitions concerning finite state automata. For a detailed exposition, we refer the reader to [5]. Given a set S, $|S|$ denotes its cardinality and 2^S the family of all its subsets. Given an alphabet Σ, the *complement* of a language $L \subseteq \Sigma^*$ is the language $L^c = \Sigma^* \setminus L$.

A *two-way nondeterministic finite automaton* (2nfa, for short) is defined as a quintuple $A = (Q, \Sigma, \delta, q_0, F)$ in which Q is the finite set of states, Σ is the finite input alphabet, $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times \{-1,0,+1\}}$ is the transition function, $\vdash, \dashv \notin \Sigma$ are two special symbols, called the left and the right endmarker, respectively, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting (final) states. Input is stored on the input tape surrounded by the two endmarkers. The cells of the input tape are numbered from left to right, beginning with zero for the left endmarker. In one move, A reads an input symbol, changes its state, and moves the input head one cell to the right, left, or keeps it stationary, depending on whether δ returns $+1$, -1, or 0, respectively. If, for some $q \in Q$ and $a \in \Sigma \cup \{\vdash, \dashv\}$, we have $|\delta(q, a)| > 1$, the machine makes *a nondeterministic choice*. If $|\delta(q, a)| = 0$, the machine *halts*. The machine accepts the input, if there exists a computation path from the initial state q_0 with head on the left endmarker to some accepting state $q \in F$. The language accepted by A, denoted by $L(A)$, consists of all input strings that are accepted. The automaton A is said to be *deterministic* (2dfa), whenever $|\delta(q, a)| \leq 1$ for all $q \in Q$ and $a \in \Sigma \cup \{\vdash, \dashv\}$. With a slight abuse of notation, we shall then write $\delta(q, a) =$ undefined, instead of $\delta(q, a) = \emptyset$, and $\delta(q, a) = (q', d)$, instead of $\delta(q, a) = \{(q', d)\}$. A (non)deterministic automaton is *one-way* (1nfa or 1dfa), if it never moves the head to the left, i.e., if $(q', d) \in \delta(q, a)$ implies $d \neq -1$.

We call *unary* any automaton that works with a single letter input alphabet. An automaton is *halting* if no computation path can get into an infinite loop, that is, on every input, each computation path halts after a finite number of steps. We say that an automaton A is *almost equivalent* to an automaton A', if the languages accepted by A and A' coincide, with the exception of a finite number of strings. If these two languages coincide on all strings, with no exceptions, A and A' are *(fully) equivalent*.

In what follows, we are particularly interested in weaker versions of 2nfa's and 2dfa's:

Definition 2.1.

– *A* quasi-sweeping *automaton (qsnfa) is a 2nfa performing* both *input head reversals and nondeterministic choices only at the endmarkers [11]. If, moreover, the above automaton is deterministic, we call it* sweeping *(qsdfa) [17].*

– *A two-way* self-verifying *automaton (2svfa) A [1] is a 2nfa which, besides the set of accepting states $F \subseteq Q$, is equipped also with $F^r \subseteq Q$, a set of so-called rejecting states. For each input $w \in L(A)$, there exists at least one computation path halting in an accepting state $q \in F$, and no path halts in a state $q \in F^r$. Conversely, for $w \notin L(A)$, there exists at least one path halting in a rejecting $q \in F^r$, and no path halts in a state $q \in F$. A 2svfa can be quasi-sweeping (qssvfa) or one-way (1svfa), with the obvious meaning.*

Note that some computation paths of a 2svfa may result in a "don't-know" answer, since the machine may also halt in states that are neither in F nor in F^r, or it may get into an infinite loop. Self-verifying two-way automata can be regarded as a generalization of 2dfa's. For example, a machine for the complement of $L(A)$ can be immediately obtained from A by exchanging accepting with rejecting states.

3 Complement for Deterministic Machines

In this section, we show that, for deterministic two-way finite state automata, the construction of an automaton for the complement of the language accepted by the original machine requires only a linear increase in the number of states. More precisely, we show that any n–state 2dfa A can be transformed into a $4n$–state halting 2dfa A' that accepts $L(A)^c$. Moreover, if the original machine already halts on every input, then n states are sufficient, that is, converting A into A' does not increase the number of states. As a consequence, we also get that any n–state 2dfa A can be replaced by an equivalent $4n$–state halting 2dfa A'. This reduces the known upper bound [18], from $O(n^2)$ to $O(n)$.

First of all, we notice that if A accepts either the empty language or all strings in Σ^* then the result is trivial. Therefore, we assume that $L(A)$ does not coincide with $\emptyset$ or Σ^*, and hence A has at least one accepting state and that the initial state q_0 is not accepting. We can make the following assumptions on the given 2dfa A, *without increasing* its number of states:

Lemma 3.1. *Each n–state 2dfa can be replaced by an equivalent 2dfa A, with at most n states, such that*

– *A has exactly one accepting state q_f, different from the initial state q_0,*
– $\delta(q_f, a) = \begin{cases} (q_f, -1) & \text{if } a \neq \vdash, \\ \text{undefined} & \text{if } a = \vdash, \end{cases}$
– *A does not perform stationary moves.*
– *If, moreover, the original machine halts on every input, then so does A.*

Now we can turn our attention to the problem of replacing the given automaton A by a machine A' accepting the complement of $L(A)$. Since A is deterministic, such a construction causes no problems, *provided that A halts on every input*: First, put the machine A into the normal form presented in Lemma 3.1. Then δ', the transition function for the automaton A', is obtained as follows.

- If, for some $q \in Q \backslash \{q_f\}$, $a \in \Sigma \cup \{\vdash, \dashv\}$, and $d \in \{-1, +1\}$, we have $\delta(q, a) = (q_f, d)$, then let $\delta'(q, a) =$ undefined.
- Similarly, if $\delta(q, a) =$ undefined, then $\delta'(q, a) = (q_f, -1)$ for $a \neq \vdash$, but $\delta'(q, a) = (q_f, +1)$ for $a = \vdash$.
- Otherwise, $\delta'(q, a) = \delta(q, a)$. This includes the case of $q = q_f$, that is, $\delta'(q_f, a) = \delta(q_f, a)$, as presented in Lemma 3.1.
- The initial and final states of A' are the same as those of A.

Since, by assumption, A is halting and, by Lemma 3.1, $q_0 \neq q_f$, it is obvious that A' accepts if and only if A does not accept.

Corollary 3.2. *For each n–state halting 2dfa A, there exists an n–state halting 2dfa A' accepting $L(A)^c$.*

Note that the assumption about halting is essential for the trivial construction above. If, for some input $w \in \Sigma^*$, A gets into an infinite loop, then A' will also get into an infinite loop, and hence w will be rejected both by A and A'.

To avoid this problem, we shall now present the construction of the halting 2dfa A', by suitably refining Sipser's construction for space bounded Turing machines [18]. To make our result more readable, we first recall Sipser's original construction (restricted to the case of 2dfa's in the normal form presented in Lemma 3.1).

For each $w \in \Sigma^*$, a deterministic machine accepts w if and only if there is a "backward" path, following the history of computation in reverse, from the unique accepting configuration $(q_f, 0)$ to the unique initial configuration $(q_0, 0)$. A "configuration" is a pair (q, i), where $q \in Q$ is a state and $i \in \{0, \ldots, |w| + 1\}$ a position of the input head.

Consider the graph whose nodes represent configurations and edges computation steps. Since A is deterministic and $\delta(q_f, \vdash) =$ undefined, the component of the graph containing $(q_f, 0)$ is a tree rooted at this configuration, with backward paths branching to all possible predecessors of $(q_f, 0)$. In addition, no backward path starting from $(q_f, 0)$ can cycle (hence, it is of finite length), because the halting configuration $(q_f, 0)$ cannot be reached by a forward path from a cycle.

Thus, the machine for the complement of $L(A)$ can perform a depth-first search of this tree in order to detect whether the initial configuration $(q_0, 0)$ belongs to the predecessors of $(q_f, 0)$. If this is the case, the simulator rejects. On the other hand, if the whole tree has been examined without reaching $(q_0, 0)$, the simulator accepts.

The depth-first search strategy visits the first (in some fixed lexicographic order) immediate predecessor of the current configuration that has not been visited yet. If there are no such predecessors, the machine travels along the edge toward the unique immediate successor. (Traveling forward along an edge is simulated by executing a single computation step of A, traveling backward is a corresponding "undo" operation).

For this search, the simulator has only to keep, in its finite control, the state q related to the currently visited configuration (q, i) (the input head position i is represented by its own input head), together with the information about the

previous visited configuration (its state and input head position relative to the current position, i.e., a number ± 1). Hence, the simulator uses $O(n^2)$ states.

Now we present our improvements to this procedure. First, fix a linear order on the state set of the original automaton, so that the final state q_f is the maximum. Hence, $q < q_f$ for any $q \in Q \backslash \{q_f\}$. As usual, the symbols "$<$" and "$>$" denote the ordering relation.

Our implementation of the depth-first search examines each configuration (q, i) in two modes; (1) examination of the "left" predecessors of (q, i), that is, immediate predecessors with input head at the position $i-1$, (2) examination of the "right" predecessors, positioned at $i+1$. For each $q \in Q$ and each of these modes, we introduce a starting and a finishing state. So the machine for the complement uses the following set of states:

$$Q' = \{q_{\nwarrow}, q_{\downarrow_1}, q_{\nearrow}, q_{\downarrow_2} : q \in Q\}\,.$$

These $4n$ states are interpreted as follows:

$q_{\nwarrow}$ starting state for Mode 1, examination of left predecessors for the configuration (q, i). A left predecessor is a configuration $(p, i-1)$, with the input head scanning a symbol a, such that $\delta(p, a) = (q, +1)$. Left predecessors will be examined one after another, according to the linear order induced by the relation "$<$". To inspect the content of the input square $i-1$ (that is, the symbol a), the simulator A' (if it is in the state $q_{\nwarrow}$) has its input head one position to the left of the actual position of the original machine A in configuration (q, i).

$q_{\downarrow_1}$ finishing state for Mode 1. All the left predecessors of (q, i) have been examined, but we still have to examine the right predecessors of (q, i). In the state $q_{\downarrow_1}$, the input head of the simulator A' is in the actual position, i.e., the position i.

$q_{\nearrow}$ starting state for Mode 2, examination of right predecessors for (q, i), when the left predecessors have been finished. A right predecessor is a configuration $(p, i+1)$, with the head scanning a symbol a, such that $\delta(p, a) = (q, -1)$. The right predecessors will also be examined in the linear order induced by "$<$". In the state $q_{\nearrow}$, the simulator A' has its input head one position to the right of the actual position of the configuration (q, i), to inspect the symbol a in the input square $i+1$.

$q_{\downarrow_2}$ finishing state for Mode 2. Both the left and the right predecessors of (q, i) have been examined. In the state $q_{\downarrow_2}$, the input head of A' is in the actual position, i.e., the position i.

The formal definition of the transition function $\delta' : Q' \times (\Sigma \cup \{\vdash, \dashv\}) \to Q' \times \{-1, 0, +1\}$ implementing this strategy is omitted for lack of space.

Since the original machine A accepts by halting in $(q_f, 0)$, and this configuration does not have any left predecessor, to decide whether an input is accepted, we can start the depth-first search from the state $q_{f\downarrow_1}$, with the head on the left endmarker. Furthermore, δ' is defined in such a way that (i) either the input is accepted by A, and then the machine A' aborts its search in the state $q_{0\downarrow_2}$ at the

left endmarker, (ii) or the input is rejected by A, and then A' stops by reaching $q_{f\downarrow_2}$ at the right endmarker. In conclusion, we can set $q'_0 := q_{f\downarrow_1}$ as initial state, and $q'_f := q_{f\downarrow_2}$ as final state. This construction leads to

Theorem 3.3. *For each n–state 2dfa A, there exists a $4n$–state halting 2dfa A' accepting $L(A)^c$.*

By Theorem 3.3 and Corollary 3.2, we get:

Corollary 3.4. *Each n–state 2dfa can be replaced by an equivalent $4n$–state halting 2dfa.*

4 Complement for Unary Nondeterministic Machines

This section is devoted to the problem of the complement for nondeterministic two-way automata. As we will discuss in Section 6, for arbitrary alphabets this problem is harder than the most famous open question in this field, posed by Sakoda and Sipser [16].

However, the situation is different for the case of unary regular languages. Using a modified version of inductive counting [2, 7, 19], we first replace a given unary 2nfa by an equivalent two-way automaton that is quasi-sweeping, self-verifying, and halting, using only a polynomial number of states. From such an automaton, building an automaton for the complement is straightforward.

First of all, we make our 2nfa quasi-sweeping, paying only by a linear increase in the number of states:

Theorem 4.1. (Geffert, Mereghetti, and Pighizzini [3, Thm. 2]) *For each n–state unary 2nfa, there exists an almost equivalent qsnfa A with no more than $2n+2$ states. The language $L(A)$ coincides with the original language on strings of length greater than $5n^2$.*

Moreover, for each accepted input, A has at least one computation path halting with the head at the left endmarker, in a unique accepting state q_f. The machine A does not perform any stationary moves, with the exception of the last computation step, when it enters q_f.

Theorem 4.1 allows us to consider a qsnfa A almost equivalent to the original 2nfa. Hence, any accepting computation follows a very regular pattern consisting of an alternation of nondeterministic choices at the endmarkers with deterministic *left-to-right* and *right-to-left* input traversals (or sweeps), until the final state q_f is reached by a single stationary move at the left endmarker.

In our simulation, we shall need a linear order on the state set Q, as in Section 3. Moreover, it will be useful to introduce a set $Q_f \subseteq Q$ of states possibly leading to acceptance, that is, the set from which the final state q_f is reachable by a single stationary move on the left endmarker:

$$Q_f = \{q \in Q : (q_f, 0) \in \delta(q, \vdash)\}\,.$$

Now we are ready to present the main result of this section, the simulation of A by an automaton that is self-verifying. For reader's ease of understanding, we prefer to present the simulating qssvfa A' in the form of an algorithm, written as high-level code. We will then informally discuss the actual implementation, evaluating the number of states required. In the following code, we use two subroutines, whose implementation is omitted for lack of space:

- *simulation*(t): a nondeterministic function, returning a nondeterministically chosen state q that is reachable by a computation path of A in exactly t traversals from the initial configuration. The call of this function may also abort the entire simulation by halting in a "don't-know" state $q_?$, due to a wrong sequence of nondeterministic guesses, if the chosen path halts too early, not having completed t traversals.
- *reach*$(q_{\rm prev}, q', {\rm dir})$: a deterministic function, with dir $\in \{0, 1\}$. It returns *true*/*false*, depending on whether the state q' can be reached from the state $q_{\rm prev}$ by a left-to-right traversal of the input (for dir $= 0$), or by a right-to-left traversal (for dir $= 1$), without performing stationary moves.

The nondeterministic simulation algorithm, based on the well-known inductive counting technique, is displayed in Figure 1.

```
 1:     m' := 1;
 2:     for t := 0 to 2|Q| − 1 do
 3:         m := m'; m' := 0;
 4:         foreach q' ∈ Q do
 5:             for i := 1 to m do
 6:                 q := simulation(t);
 7:                 if i > 1 and q ≤ q_prev then halt in q_?;
 8:                 q_prev := q;
 9:                 if reach(q_prev, q', t mod 2) then
10:                     if t is odd and q' ∈ Q_f then halt in q_yes;
11:                     m' := m'+1; goto next_q'
12:             end end
13: next_q':  end
14:     end;
15:     halt in q_no
```

Fig. 1. The simulation procedure

Basically, the algorithm proceeds by counting, for $t = 0, \ldots, 2|Q| - 1$, the number of states reachable by A at the endmarkers, by all computation paths starting from the initial configuration and traversing the input exactly t+1 times. As a side effect of this counting, the algorithm generates all states reachable at the endmarkers, and hence it can correctly decide whether to accept or reject the given input.

For each accepted input, A has at least one accepting path that halts at the left endmarker; furthermore, the input must also be accepted by a path with no more than $2|Q| - 1$ traversals. Otherwise, the machine A would repeat

the same state on the same endmarker. For this reason, the loop running for $t = 0, \ldots, 2|Q|-1$ (nested between lines 2 and 14) suffices to detect an accepting computation.

At the beginning of the t-th iteration of this loop, a variable m' contains the exact number of states reachable at the endmarkers by all computation paths that traverse the input exactly t times. (Initially, in line 1, we prepare $m' = 1$ for $t = 0$, the only state reachable by no traversals is the initial state q_0.) In line 3, we save the "old" value of m' in the variable m, and clear m' for counting the number of states reachable upon completing one more traversal (i.e., with exactly $t+1$ traversals). The value of m' is computed in the loop nested between lines 4 and 13, running for each state $q' \in Q$. For each q', we test whether it is reachable by a path with exactly $t+1$ traversals. If it is, we increment the value of m'.

It is easy to see that: (i) if the input is accepted by A, at least one computation path of A' halts in the accepting state q_{yes}, and no path halts in the rejecting state q_{no}, (ii) if the input is rejected, at least one path halts in q_{no}, and no path halts in q_{yes}. (iii) Due to wrong sequences of nondeterministic guesses, some computation paths halt in $q_?$ (don't-know), but no path can get into an infinite loop. The automaton A' performs nondeterministic choices and input head reversals at the endmarkers only, during the calls of *simulation* and *reach*.

Let us now quickly examine the "amount of information" required during the computation, by considering the involved variables. Let n be the number of states of A. The simulation procedure uses the variables m, m', q, q', q_{prev}, and i, each one ranging over at most n possible different values, and the variable t with at most $2n$ different values. Furthermore, it can be shown that both the subroutines *simulation* and *reach* can be implemented by introducing a single variable containing one of at most $2n$ different values. All this information can be accommodated in $O(n^8)$ states.

In conclusion, we get that any quasi-sweeping 2nfa with n states can be replaced by an equivalent 2nfa that is quasi-sweeping, self-verifying, and halting, with $O(n^8)$ states.

Using Theorem 4.1, we can make any unary 2nfa quasi-sweeping, which therefore allows us to simulate an arbitrary n–state unary 2nfa. However, the new machine may disagree with the original one on "short" inputs, of length not exceeding $5n^2$. This problem can be easily solved by adding an initial phase, consisting of a single left-to-right traversal, followed by a single right-to-left traversal, to accept or reject all inputs of length not exceeding $5n^2$. This can be done deterministically, with $O(n^2)$ states. Hence

Theorem 4.2. *Each n–state unary 2nfa A can be replaced by an equivalent $O(n^8)$–state halting qssvfa A'.*

As pointed out in Section 2, by simply exchanging accepting with rejecting states in the self-verifying A' of Theorem 4.2, we get a machine for the complement of $L(A')$.

Corollary 4.3. *For each n–state unary 2nfa A, there exists an $O(n^8)$–state 2nfa A' accepting the complement of $L(A)$. Moreover, A' is quasi-sweeping, self-verifying, and halting.*

5 Simulation by Unary Probabilistic Machines

Theorem 4.2 and Corollary 4.3 allow us to easily draw some further consequences. In the case of unary two-way automata, the reduction of nondeterminism to self-verifying nondeterminism can go down one more level, to Las Vegas probabilistic automata, still with only an $O(n^8)$–state penalty.

A two-way *Las Vegas* finite state automaton (2lvfa) [6] may be viewed as a 2svfa equipped with some probabilistic transitions. If $w \in L(A)$, the machine halts in some accepting state $q \in F$ with a probability of at least $1/2$, but the probability of halting in a rejecting state $q \in F^r$ is zero. Conversely, for $w \notin L(A)$, the probability of halting in some rejecting state $q \in F^r$ is at least $1/2$, but it is zero for $q \in F$. Hence, a Las Vegas automaton never returns a wrong answer. Moreover, the probability of a "don't-know" answer is below $1/2$.

For the size (the number of states) of one-way finite automata, a polynomial relation between determinism and Las Vegas has been established [1], which implies a superpolynomial gap between Las Vegas and nondeterminism. In the case of two-way automata, we do not know whether there is a polynomial relation between determinism and Las Vegas, or between Las Vegas and nondeterminism. However in [6, Thm. 1] it is proved that each n–state 2svfa can be replaced by an equivalent $O(n)$–state 2lvfa. Combining this result with Theorem 4.2, we get that, in the case of unary two-way automata, even unrestricted nondeterminism and Las Vegas are polynomially related:

Theorem 5.1. *Each n–state unary 2nfa can be replaced by an equivalent $O(n^8)$–state 2lvfa.*

This might give additional evidence that nondeterministic two-way automata, when restricted to unary inputs, are not as powerful as they might seem at first glance. Compare with Theorem 4 in [3], where a subexponential (though not polynomial) simulation of unary 2nfa's by 2dfa's is presented, with a $2^{O(\log^2 n)}$–state penalty.

6 Concluding Remarks

We have shown that, for deterministic two-way finite automata, the construction of an automaton for the complement of the language accepted by the original machine requires only a linear increase in the number of states. For nondeterministic two-way automata, when restricted to unary input alphabet, this relationship is polynomial.

We were not able to resolve the problem of complement for 2nfa's in the general (nonunary) case. We conjecture that there is no polynomial complementation of 2nfa's. However, proving this gap is a more complicated task than an

argument resolving the Sakoda and Sipser open question [16] that asks whether the simulation of 2nfa's by 2dfa's is polynomial in the number of states. It is generally conjectured that the answer is negative, so we can formulate this open problem as follows:

(a) Prove that there exists an exponential gap (or at least superpolynomial, in the number of states) between nondeterministic and deterministic two-way finite automata.

Now we present two open problems that are even more difficult:

(b) Prove that there exists an exponential, or at least superpolynomial, gap between self-verifying and deterministic two-way finite automata.
(c) Prove that there exists an exponential, or at least superpolynomial, gap between nondeterministic and self-verifying two-way finite automata.

Clearly, an argument for (b) would immediately prove (a), since a self-verifying machine is also a nondeterministic machine. In this sense the problem (b) is more difficult to solve than (a).

The same holds for (c), since, by Corollary 3.4, we can make each 2dfa halting, with $O(n)$ states. Then, by introducing an extra rejecting state q_f^r (if the original machine does not accept), we make the given deterministic machine self-verifying. However, the problem (c) is equivalent to the following:

(d) Prove that there exists an exponential, or at least superpolynomial, gap between 2nfa's and 2nfa's accepting the complement.

Suppose, in fact, that some $p(n)$ states are sufficient to make an arbitrary 2nfa self-verifying. Then $p(n)$ states are sufficient to build a machine for the complement, by exchanging accepting with rejecting states in the 2svfa. Conversely, if there exists a polynomial complementation of 2nfa's, using $p(n)$ states, then $n+p(n)$ states are sufficient to make an arbitrary 2nfa self-verifying. We simply combine A_1 and A_2, the respective machines for the language and its complement, into a single machine. The set of accepting states of the new machine is exactly the set of accepting states of A_1, the set of its rejecting states exactly the set of accepting states of A_2. The new machine starts in $q_{0,1}$, the initial state of A_1, however, it can switch to the initial state of A_2 at the left endmarker, by $(q_{0,2}, 0) \in \delta(q_{0,1}, \vdash)$.

Thus, the problem (d) is equivalent to (c), and hence more complicated than (a).

Finally, taking into account the upper bounds presented in Corollary 3.2 and Theorem 3.3, the following open problem arises:

(e) Prove (or disprove) the existence of *any* gap between 2dfa's and 2dfa's accepting the complement.

By Corollary 3.2, it follows that a minimized 2dfa for the witness language (if it exists) must reject some inputs by getting into an infinite loop.

Acknowledgments

The authors wish to thank anonymous referees for helpful comments and remarks.

References

1. Dietzelfelbinger M., Kutylowski M., Reischuk R.: Exact lower bounds for computing Boolean functions on CREW PRAMs. J. Comput. System Sci., **48** (1994) 231–54.
2. Geffert V.: Tally versions of the Savitch and Immerman-Szelepcsényi theorems for sublogarithmic space. SIAM J. Comput., **22** (1993) 102–13.
3. Geffert V., Mereghetti C., Pighizzini G.: Converting two-way nondeterministic automata into simpler automata. Theoret. Comput. Sci., **295** (2003) 189–203.
4. Goldstine J., Kappes M., Kintala C., Leung H., Malcher A., Wotschke D.: Descriptional complexity of machines with limited resources. J. Universal Comput. Sci., **8** (2002) 193–234.
5. Hopcroft J., Ullman J.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.
6. Hromkovič J., Schnitger G.: On the power of Las Vegas II: Two-way finite automata. Theoret. Comput. Sci., **262** (2001) 1–24.
7. Immerman N.: Nondeterministic space is closed under complementation. SIAM J. Comput., **17** (1988) 935–38.
8. Jirásek J., Jirásková G., Szabari A.: State complexity of concatenation and complementation of regular languages. Pre-Proc. 9$^{\text{th}}$ Conf. Implementation and Application of Automata (CIAA 2004). Queen's Univ., Kingston, Ontario, Canada, 2004, pp. 132–42.
9. Kondacs A., Watrous J.: On the power of quantum finite state automata. Proc. 38$^{\text{th}}$ Symp. Found. Comp. Sci. (FOCS 1997). IEEE Computer Society Press, 1997, pp. 66–75.
10. Mera F., Pighizzini G.: Complementing unary nondeterministic automata. Theoret. Comput. Sci., **330** (2005) 349–360.
11. Mereghetti C., Pighizzini G.: Two-way automata simulations and unary languages. J. Aut. Lang. Combin., **5** (2000) 287–300.
12. Meyer A., Fischer M.: Economy of description by automata, grammars, and formal systems. Proc. 12$^{\text{th}}$ Ann. IEEE Symp. on Switching and Automata Theory, 1971, pp. 188–91.
13. Moore C., Crutchfield J.: Quantum automata and quantum grammars. Theoret. Comput. Sci., **237** (2000) 275–306.
14. Moore F.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata by deterministic automata. IEEE Trans. Comput., **C-20** (1971) 1211–19.
15. Rabin M.O.: Probabilistic automata. Inform. & Control, **6** (1963), 230–45.
16. Sakoda W., Sipser M.: Nondeterminism and the size of two-way finite automata. Proc. 10$^{\text{th}}$ ACM Symp. Theory of Computing, 1978, pp. 275–86.
17. Sipser M.: Lower bounds on the size of sweeping automata. J. Comput. System Sci., **21** (1980) 195–202.
18. Sipser M.: Halting space bounded computations. Theoret. Comput. Sci., **10** (1980) 335–38.
19. Szelepcsényi R.: The method of forced enumeration for nondeterministic automata. Acta Inform., **26** (1988) 279–84.

On Timed Automata with Discrete Time – Structural and Language Theoretical Characterization

Hermann Gruber[1], Markus Holzer[1], Astrid Kiehn[2], and Barbara König[3,⋆]

[1] Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching bei München, Germany
{gruberh,holzer}@in.tum.de
[2] Department of Computer Science and Engineering,
Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India
astrid@cse.iitd.ernet.in
[3] Institut für Formale Methoden der Informatik, Universität Stuttgart,
Universitätsstraße 38, D-70569 Stuttgart, Germany
koenigba@fmi.uni-stuttgart.de

Abstract. We develop a structural and language theoretical characterization of timed languages over discrete time in terms of a variant of Büchi automata and languages. The so-called tick automaton is a standard Büchi automaton with a special "clock-tick"-input symbol modeling the discrete flow of time. Based on these characterizations we give an alternative proof for the fact that the class of regular timed languages is closed under complementation and formulate a time-warp lemma which, similar to a pumping lemma, can be used to show that a timed language is not regular. The characterizations hold alike for timed automata with and without periodic clock constraints.

1 Introduction

Timed automata have been introduced in [1] in order to model real-time systems from a quantitative perspective, to make specification and verification of models of real-time systems easier. A timed automaton is a Büchi automaton with a finite set of clocks, which can be independently reset, and where the automaton can keep track of the time elapsed since the last reset. Several generalizations have been investigated as, in particular, timed automata with silent transitions [5] and timed automata with periodic clock constraints [6]. Both these extensions are equally powerful and strictly increase the expressive power of timed automata, regardless of the used time semantics, i.e., the interpretation of the clocks, which assigns some value to each clock. The two major semantics discussed in the literature are the discrete-time and the dense-time model. Although the latter is more natural from a physical point of view and allows an easy modelling of real-time systems, it requires decision methods for real-time

⋆ Research partially supported by DFG project SANDS and EPSRC grant R93346/01

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 272–283, 2005.
© Springer-Verlag Berlin Heidelberg 2005

logics, which in general are undecidable [2]. In fact, the class of timed languages accepted by automata interpreted under the dense-time model is closed under all positive Boolean operations, but is not closed under complementation [1]. The latter is also true for timed languages over discrete time, except for the complementation closure, which was shown to hold in [9] in terms of a variant of monadic second order logic.

However, as recently pointed out by Asarin [3], despite the availability of tools such as UPPAAL and KRONOS, the foundations of the theory of timed languages are comparatively weak compared to the classical theory of finite automata and regular languages. For instance Asarin states in his open question 3: "Give a simple and natural algebraic characterization of a known class of timed languages – confirming that it is the correct class of languages." With the present paper we provide such a structural characterization for the class of discrete-time languages accepted by timed automata with or without periodic clock constraints (or equivalently with or without silent transitions). To this end we transform timed automata into so-called tick automata. A tick automaton is a Büchi automaton where the input alphabet is equipped with an additional "clock-tick"-symbol ($\surd$) modelling the discrete flow of time[1]. Intuitively, actions are occurring inbetween such clock ticks, hence, our time semantics is such that it allows (finite or infinite) stuttering of actions on a particular time stamp during the computation. It turns out that any tick automaton induced by a timed automaton has a particular $\surd$-transition structure, namely deterministic tick-paths ending in (trivial or non-trivial) tick-loops. On the other hand, we can show that any tick automaton (not necessarily induced by a timed automaton) can be converted into this normal form. This nicely corresponds to the intuitive behaviour of time as a sort of deterministic or non-branching flow.

While timed automata in general induce non-trivial tick-loops, the tick-loops become trivial for aperiodic timed automata. Besides this structural characterization of timed languages in terms of tick automata, we provide a language theoretical characterization for languages accepted by aperiodic timed automata by a simple condition, which will be called $\surd$-stretchiness. This condition is closely related to the aperiodicity condition of regular languages on finite words. Loosely speaking, $\surd$-stretchiness tells us that tick actions in direct sequence can only be counted up to a given threshold, which depends on the language only. We hope that these characterizations give some further insights into the behaviour of timed languages.

The provided characterizations can be used to show an alternative way to reprove the complementation closure of timed aperiodic languages, directly in the framework of Büchi automata, thus reproving the result given in [9]. Moreover, we develop a sort of pumping lemma, the so called time-warp lemma, which shows that certain actions in a run of a word on the timed automaton can be moved along the time axis to the future keeping the acceptance invariant. To illustrate the strength of the time-warp lemma, we give very simple proofs for languages that cannot be accepted by any timed (aperiodic) automaton. In

[1] Note that our model of time is called the fictitious-clock model in [1]

addition we mention that the time-warp lemma is not limited to the discrete time semantics, but holds for dense-time semantics as well. Hence this is a (partial) answer to open question 14 – "Develop simple techniques allowing to prove that a given timed language is not regular" – stated in [3]. All presented results are effectively constructible and therefore can be used for the verification of discrete timed systems.

The paper is organized as follows: The next section contains preliminaries on timed automata and timed languages. Section 3 introduces tick automata, and shows how to transform timed automata to tick automata, by using the region automata. The next section is devoted to the structural and language theoretical characterization of timed languages in terms of tick automata. As a byproduct we provide the reader with an alternative proof of the complementation problem of timed languages, by automata theoretical constructions only. Then in Section 5 we develop the time-warp lemma and give some further applications. Finally we summarize our results in Section 6.

2 Definitions

Timed automata can be considered as Büchi-automata which have been equipped with a finite number of clocks. These clock run simultaneously and can individually be reset to 0 by a change of state, which in turn depends on the current values of the clocks. Timed automata are usually interpreted under a dense time domain, that is, a clock value can be any real number. This paper only considers a discrete time semantics which assumes an underlying fictitious clock. For a discussion of the different time models we refer to [1, 5, 6].

Many variations of timed automata can be found in the literature. The most general form allows for silent (ε) transitions and as constraints any Boolean expression over atomic assertions of the form $x = c$, $x < c$, and $x =_m c$ (comparison modulo m), where x is a clock and $c, m \in \mathbb{N}$, or even direct comparisons of different clock values. It has been shown in [1] that direct comparisons of clock values provide an even more succinct representation, and in [5] and [6] that silent transitions and modulo constraints are mutually expressible and do increase the power of classical timed automata introduced in [1] (which coincide with aperiodic timed automata in our setting). To keep the technical presentation as simple as possible we use the results of [6, 7] and assume just one clock and a restricted constraint language. Every timed automaton can effectively brought into this form though its size might increase substantially.

Definition 1. *A* timed automaton $A = \langle Z, \Sigma, E, z_S, R, \{x\}\rangle$ *is given by a finite set of states* Z*, the input alphabet* Σ*, a start state* $z_S \in Z$*, an acceptance set* $R \subseteq Z$*, one clock variable* x *and the set of transitions* $E \subseteq Z \times \Sigma \times 2^{\Phi} \times 2^{\{x\}} \times Z$*, where* $\Phi = \{x = i, x \neq i, x =_m i, x \neq_m i \mid 0 \leq i < m\}$ *and* $m \in \mathbb{N}$*. An* aperiodic *timed automaton differs only in the constraint universe* $\Phi = \{x = i, x \neq i, x \geq m \mid 0 \leq i < m\}$.

We will sometimes use the term *periodic timed automaton* in order to distinguish timed automata from aperiodic timed automata.

We denote $\langle z, a, \varphi, X, z' \rangle \in E$ by $z \overset{a,\varphi,X}{\longrightarrow} z'$, where φ is called the constraint set (or simply constraint) and X the reset set. A configuration $\langle z, v \rangle$ of A consists of a state and a clock assignment. As we consider just one clock, a clock assignment reduces to an integer $v \in \mathbb{N}$. A *timed run* on a timed automaton is an infinite sequence

$$\pi = \langle z_0, v_0 \rangle \overset{a_1,t_1}{\longrightarrow} \langle z_1, v_1 \rangle \overset{a_2,t_2}{\longrightarrow} \langle z_2, v_2 \rangle \overset{a_3,t_3}{\longrightarrow} \cdots,$$

where $z_0 = z_S$, $a_i \in \Sigma$ and $t_i \in \mathbb{N}$ such that $v_0 = 0$ and for each $i \geq 0$ there is an underlying transition $z_i \overset{a_{i+1},\varphi,X}{\longrightarrow} z_{i+1}$ with (a) $v_i + \Delta_i \models \varphi$ and (b) $v_{i+1} = v_i + \Delta_i$, if $X = \emptyset$, and $v_{i+1} = 0$, if $X = \{x\}$, where $\Delta_i := t_{i+1} - t_i$ and $t_0 = 0$, with $\Delta_i \geq 0$. Condition (a) expresses that the guarding constraint φ is satisfied before the transition is taken, while (b) ensures that the clock is adjusted correctly. A clock assignment satisfies a set of constraints if it satisfies every constraint contained in the set.

A timed run π is called *accepting* if $Inf(\pi) \cap R \neq \emptyset$, where $Inf(\pi)$ denotes the set of states occurring infinitely often in π. From a timed run π we extract the timed word $w(\pi) = \langle a_1, t_1 \rangle \langle a_2, t_2 \rangle \langle a_3, t_3 \rangle \ldots \in (\Sigma \times \mathbb{N})^\omega$, which, as usual, is the constituent of the timed language of A:

$$L^\omega(A) = \{\, w(\pi) \in (\Sigma \times \mathbb{N})^\omega \mid \pi \text{ is an accepting run on } A \,\}.$$

Note, that by definition the time stamp sequence is monotonously increasing. Moreover, we diverge from the literature in that we do not demand non-zenoness, that is, each run indeed diverges in time. By doing so, we simplify the constructions to come. As shown in [5], non-zenoness can be enforced by an additional automata theoretical construction.

3 From Timed Automata to Tick Automata

The fictitious clock semantics suggests another way of behaviour description: rather than equipping an event with the current clock time, the ticking of the clock can be modeled by a particular *tick-action*, $\surd$. For instance, the finite timed word $\langle a, 1 \rangle \langle b, 3 \rangle \langle a, 3 \rangle \langle a, 7 \rangle$ corresponds to $\surd a \surd \surd b a \surd \surd \surd \surd a$. We call the latter representation the *ticked-version* of the former.

Definition 2. *Let $w = \langle a_1, t_1 \rangle \langle a_2, t_2 \rangle \ldots \langle a_i, t_i \rangle \ldots$ be a timed word. Its ticked version is $w_\surd = \surd^{\Delta_0} a_1 \surd^{\Delta_1} a_2 \ldots a_{i-1} \surd^{\Delta_{i-1}} a_i \ldots$ with $\Delta_i = t_{i+1} - t_i$ and $t_0 = 0$. Let $untick(w_\surd) = w$ be the inverse operation. Then the ticked version of a timed language L is defined as $L_\surd := \{\, w_\surd \in (\Sigma \uplus \{\surd\})^\omega \mid untick(w_\surd) \in L \,\}$, where $\uplus$ denotes the disjoint union of sets.*

We show in this section that every (discrete) timed automaton A can be represented as a Büchi-automaton with a distinguished $\surd$-action such that $(L^\omega(A))_\surd$ coincides with the ω-language of the Büchi automaton. We denote this Büchi automaton by $A_\surd$ and call it the *tick automaton of A*. In general, a tick automaton is a Büchi automaton with a new action $\surd$ representing a tick of an underlying

fictitious clock. A tick automaton B is given by $\langle Z, \Sigma \uplus \{\surd\}, E, z_S, R\rangle$, where Z, Σ, z_S and R are interpreted as in the case of timed automata but E simplifies to $E \subseteq Z \times \Sigma \uplus \{\surd\} \times Z$. A word w is in the accepted language $L^\omega(B)$ if (1) the word w contains infinitely many non-$\surd$ actions and (2) the underlying run of w contains infinitely many occurrences of acceptance states (the usual Büchi acceptance condition). Note that condition (1) and (2) give an adequate counterpart to languages accepted by timed automata. Since we do not require non-zenoness there, we do not demand infinitely many $\surd$-actions occurring in w as acceptance condition.

To express the timed behaviour of an automaton as a tick automaton we use the concept of regions which have been introduced in [1] in order to describe the untimed behaviour of a timed automaton. A region equates all those clock valuations which are undistinguishable under progress of time or clock resetting with respect to the evaluation of constraints, i.e., a region is an equivalence class.

As we deal with one clock x only, the regions are simply described by the region expressions $x = i$ and $x \geq m \wedge x =_m i$, for $0 \leq i < m$, and in case of aperiodic timed automata by $x = i$ and $x \geq m$, for $0 \leq i < m$, where m is the constant in the definition of an automaton. Note that in both cases, the clock regions provide a partition on the time domain $\mathbb{N}$. Two clock valuations v and v' are equivalent, $v \sim v'$, if they satisfy the same region expression. The following lemma ensures soundness of our construction.

Lemma 3. *Let v and v' be clock valuations with $v \sim v'$. Then*

1. *$v \models \varphi$ if and only if $v' \models \varphi$ for any clock constraint φ occurring in A, and*
2. *$(v + \Delta) \sim (v' + \Delta)$ for any $\Delta \in \mathbb{N}$.*

For the tick automaton $A_\surd$ we pair the states of A with its regions. The transitions are induced by the transitions of A which are now split into a delay and an action part.

Definition 4 (Automaton $A_\surd$). *Let $A = \langle Z, \Sigma, E, z_S, R, \{x\}\rangle$ be a timed automaton. Then $A_\surd$ is the tick automaton $\langle Z', \Sigma \uplus \{\surd\}, E', z'_S, R'\rangle$, where the set of states is $Z' = \{\, \langle z, \alpha\rangle \mid z \in Z, \alpha \text{ a region expression} \,\}$, the initial state equals $z'_S = \langle z_S, x = 0\rangle$, $R' = \{\, \langle z, \alpha\rangle \mid z \in R \,\}$ and E' contains*

1. *$\langle z, \alpha\rangle \xrightarrow{a} \langle z', \alpha\rangle$ if there is $z \xrightarrow{a,\varphi,\emptyset} z'$ in A such that $\alpha \models \varphi$ (non-reset transitions),*
2. *$\langle z, \alpha\rangle \xrightarrow{a} \langle z', x = 0\rangle$ in $A_\surd$ if there is $z \xrightarrow{a,\varphi,\{x\}} z'$ in A such that $\alpha \models \varphi$ (reset transitions),*
3. *$\langle z, \alpha\rangle \xrightarrow{\surd} \langle z, \alpha + 1\rangle$ for all states $\langle z, \alpha\rangle \in Z'$ (tick transitions).*

Observe that the non-reset and reset transitions are executed without any delay. The delay that might be necessary in A to move from state z to z' is performed in $A_\surd$ by the respective number of $\surd$-transitions. These transitions lead to the immediate time successor of the current region but do not leave state z. A time successor $\alpha + k$ of a region expression α is defined by $\alpha + k = [v + k]$ for $\alpha = [v]$.

Theorem 5. *Let A be a timed automaton. Then*

$$(L^\omega(A))_\surd = L^\omega(A_\surd) \cap \{\, w \in (\Sigma \uplus \{\surd\})^\omega \mid w \text{ contains infinitely many } a \in \Sigma \,\}.$$

The size of $A_\surd$ is determined by the number of states and transitions. The number of states of $A_\surd$ is $|Z'| = |Z| \times$ (number of regions). We have $|Z'| = |Z| \cdot 2m$ for the case of periodic clock constraints, and $|Z'| = |Z| \cdot (m+1)$ for automata using only aperiodic constraints. The number of transitions is bound by $|E| \cdot (m+1) + |Z| \cdot (m+1) = (|E| + |Z|)(m+1)$ for the aperiodic case, and by $|E| \cdot 2m + |Z| \cdot 2m = (|E| + |Z|) \cdot 2m$ in general.

Example 6. Figure 1 shows an aperiodic timed automaton with accepted language

$$L_1 = \{\, \langle a_i, t_i \rangle_{i \geq 1} \mid \exists k \geq 0 : (\forall i < k : a_i = a \wedge t_i = 2i) \\ \wedge\, (a_k = b \wedge t_k > 2k) \wedge (\forall j > k : a_j = a) \,\}.$$

Its tick automaton is also given in Figure 1, where $\alpha_0 = (x = 0)$, $\alpha_1 = (x = 1)$, $\alpha_2 = (x = 2)$, and $\beta = (x \geq 3)$ are the region expressions.

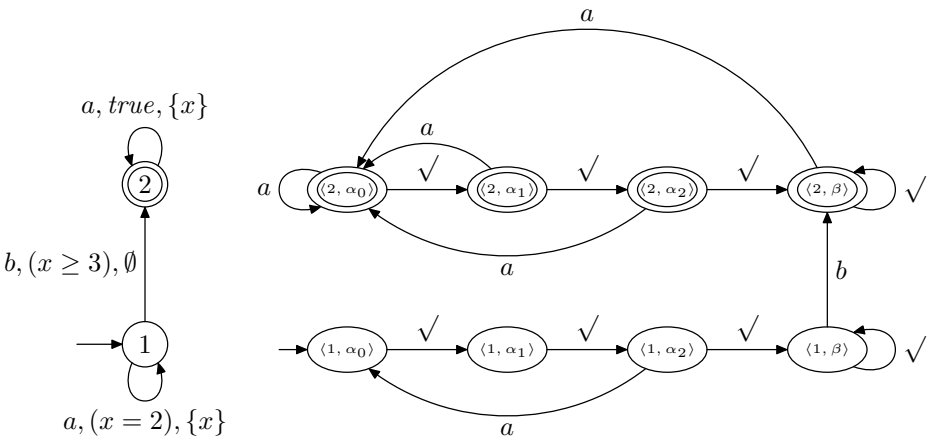


Fig. 1. A timed automaton with aperiodic clock constraints (Example 6) and its corresponding tick automaton

We now have a closer look at the structure of the tick automaton $A_\surd$. It follows immediately from the definition that there is no branching with respect to tick-transitions. For aperiodic automata we can additionally observe that there are no non-trivial cycles in which the transitions are labelled by $\surd$ only, while in arbitrary timed automata the length of tick-cycles is m, where m is the constant of the modulo constraints. Formally, these properties read as follows:

1. Tick determinism: $z \xrightarrow{\surd} z'$ and $z \xrightarrow{\surd} z''$ imply $z' = z''$.
2. Tick-cycle freedom: $z_1 \xrightarrow{\surd} z_2 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n$ and $z_1 \neq z_2$ imply $z_1 \neq z_n$.

3. Constant tick-cycle length: $z_1 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n \xrightarrow{\surd} z_1$ and $z_1, \ldots, z_n$ are pairwise distinct implies $n = m$ where m is the constant given in the constraint universe of periodic timed automata.

Note that a tick-loop, i.e., a transition $z \xrightarrow{\surd} z$, does not count as a tick-cycle.

Theorem 7. *Let A be a timed automaton and $A_{\surd}$ its corresponding tick automaton. Then the following implications hold:*

1. *If A is aperiodic then $A_{\surd}$ is tick-deterministic and tick-cycle free, and*
2. *if A is periodic then $A_{\surd}$ is tick-deterministic and has a constant tick-cycle length.*

The converse of the theorem given above is shown in Section 4 on structural characterizations. Observe that for timed languages in general, due to the correspondence of ε-transitions and modulo constraints [6], there is an easy construction that transforms an arbitrary tick automaton into an equivalent timed automaton. We introduce a new clock x, replace each tick transition $z \xrightarrow{\surd} z'$ in the tick automaton by an ε-transition $z \xrightarrow{\varepsilon, x=1, \{x\}} z'$ and then use the fact that timed automata with silent actions can be transformed into equivalent ε-free timed automata with modulo constraints. In passing we have hence shown the following corollary:

Corollary 8. *For each tick automaton A there is a periodic timed automaton B such that $L^{\omega}(A) = (L^{\omega}(B))_{\surd}$.*

4 Characterizations of Aperiodic Timed Languages

In this section we develop a structural and language theoretical characterization of aperiodic timed languages in terms of tick automata and languages, respectively. An application of both characterizations to the complementation problem is given in Subsection 4.3.

4.1 Structural Characterization of Aperiodic Timed Languages

We give a structural characterization of timed languages accepted by aperiodic timed automata. We state first that every tick-cycle free aperiodic tick automaton can be made tick-deterministic.

Lemma 9. *Each tick-cycle free tick automaton A can be effectively transformed into a tick automaton B such that $L^{\omega}(A) = L^{\omega}(B)$ and B is tick-deterministic and tick-cycle free.*

For a tick-deterministic tick automaton A, which is tick-cycle free, it is now straightforward to define a timed automaton B with one clock x such that $(L^{\omega}(B))_{\surd} = L^{\omega}(A)$. Here B contains the set of states of A. For each path

$z \xrightarrow{a} z_0 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n \xrightarrow{b} z_{n+1}$ where a and b are non-$\surd$ actions, we either introduce a transition $z_0 \xrightarrow{b,x=n,\{x\}} z_{n+1}$ if there is no tick loop at z_n or $z_0 \xrightarrow{b,x\geq n,\{x\}} z_{n+1}$ in the presence of such a loop. For the initial state z_S we proceed similarly. Furthermore we introduce appropriate acceptance states and convert the constraints into the form of Definition 1. Hence we can give the following structural characterization of aperiodic timed languages.

Theorem 10. *A tick automaton A is language equivalent to a tick-deterministic and tick-cycle free tick automaton if and only if there is an aperiodic timed automaton B such that $L^\omega(A) = (L^\omega(B))_\surd$.*

4.2 Language Theoretical Characterization of Aperiodic Timed Languages

We give a language theoretical characterization of timed language accepted by aperiodic timed automata. To be more precise, we define a condition, which holds exactly for all languages which are tick versions of aperiodic timed languages and *vice versa*. The condition reads as follows:

Definition 11 ($\surd$-stretchy). *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is called $\surd$-stretchy if and only if there exists an $n \in \mathbb{N}$ such that (1) for each infinite sequence $w_1, w_2, w_3, \ldots$ of finite words over the alphabet $\Sigma \uplus \{\surd\}$ and for each infinite sequence $i_1, i_2, i_3, \ldots$ of nonnegative numbers*

$$w_1\surd^n w_2\surd^n w_3\surd^n \ldots \in L \iff w_1\surd^{n+i_1} w_2\surd^{n+i_2} w_3\surd^{n+i_3} \ldots \in L$$

and (2) for every $w \in (\Sigma \uplus \{\surd\})^*$, $v \in (\Sigma \uplus \{\surd\})^\omega$ and each nonnegative number i,

$$w\surd^n v \in L \iff w\surd^{n+i} v \in L.$$

The following lemma immediately follows by definition.

Lemma 12. *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is $\surd$-stretchy if and only if the complement of L, i.e., the language $(\Sigma \uplus \{\surd\})^\omega \setminus L$, is $\surd$-stretchy.*

Moreover, it is not hard to see that every language accepted by a tick automaton $A_\surd$, where A is aperiodic, is $\surd$-stretchy. This is due to the fact, that whenever $A_\surd$ has read a block of $\surd$ which is long enough it must be in a state, which corresponds to the maximal clock region. Since this state must have a tick-loop, one can enlarge the $\surd$-block by an arbitrary number of $\surd$'s without changing the acceptance of the original word. Thus, the $\surd$-stretchy condition is satisfied for $n = m + 1$, where m is the maximal constant occurring in the set of clock constraints of the timed aperiodic automaton A. In terms of $A_\surd$, one can choose the number of its states as a suitable n.

Before we prove the converse relation, i.e., that every $\surd$-stretchy ω-regular language over the alphabet $\Sigma \uplus \{\surd\}$ accepted by a tick automaton $A_\surd$ corresponds to an aperiodic timed language, we need the following technical lemma.

Lemma 13. *Let $L \subseteq (\Sigma \cup \{\surd\})^\omega$ be a $\surd$-stretchy language accepted by a tick automaton A. Then a tick automaton B can be effectively constructed from A such that $L^\omega(B) = L$ and B is tick-cycle free.*

Proof. Let n be the constant from the $\surd$-stretchy condition, which is satisfied by L. It suffices to show that every non-trivial strongly connected component of $\surd$-transitions can be eliminated from the tick automaton A, without changing the accepted language.

Fix one non-trivial strongly connected component of $\surd$-transitions and let S be the set of all states contained within this component. Without loss of generality we may assume that there is no a-transition leading from z to z' with $z, z' \in S$. Define $R_{s,s'}$ be the set of all words representing a path from s to s', which lies completely within S. Observe that $R_{s,s'}$ is regular for every $s, s' \in S$. Then for every $s, s' \in S$ such that s has an in-going non-$\surd$ transition and s' an outgoing a-transition to a state t, which is not contained in S, we proceed in three steps: (1) Introduce a new edge from s to t labeled with the regular expressions: (i) All expressions of the form wa with $w \in R_{s,s'}$ and $|w| < n$ and (ii) $\surd^n \surd^* a$. (2) If there is an accepting state $s'' \in S$, then introduce a new accepting state t', which is appropriately connected to all successors of t, and introduce a new edge from s to t' labeled with the regular expressions: (i) All expressions of the form wa with $w \in R_{s,s''} R_{s'',s'}$ and $|w| < n$ and (ii) $\surd^n \surd^* a$. (3) Finally, one removes all $\surd$-transitions, which were once part of the strongly connected cycle, and converts the regular expression on the newly introduced edges to paths, the structure of which contains no non-trivial tick cycles. This completes the description of the construction. It remains to verify the correctness.

Consider step (1) in more detail. The only way to eventually accept a word w, which is not in the language is to use a word from the expression $\surd^n \surd^* a$ going from s to t *via* s'. Then we distinguish two cases: The new edge is traversed infinitely or finitely often. We only prove the former case, since the latter can be shown by similar arguments. Now assume that the edge under consideration is traversed infinitely often and $w = w_1 \surd^{n+c_1} a w_2 \surd^{n+c_2} a w_3 \surd^{n+c_3} \dots$ where the substrings of the form $\surd^{n+c_i} a$, for $c_i \geq 0$, are due to the new edge. Then we distinguish two cases: If $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \dots$ is in L, then so is w. Otherwise, assume that $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \dots$ is not a member of L, but w is accepted by the new machine. Then instead of using the newly introduced edge, we alter the computation such that one particular path from s to t *via* s'' within the strongly connected cycle is taken. Thus, we end up with a word $w' = w_1 \surd^{n+c} a w_2 \surd^{n+c} a w_3 \surd^{n+c} \dots$ for some constant c, which is also accepted by the original Büchi automaton, and hence lies in L – the accepting states that were seen infinitely often in the run of w do not belong to $S \setminus \{s\}$. Then we immediately obtain a contradiction, because by the $\surd$-stretchy condition also the word $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \dots$ has to be accepted, which was ruled out by our assumption. Thus, step (1) does not alter the accepted language.

Furthermore, when removing the $\surd$-transitions that were once part of the strongly connected component in step (3), we have to consider computations that visit a possible acceptance state belonging to S. This is done in step (2).

By a similar reasoning as above one observes that the newly introduced edge together with the acceptance state t' does not alter the underlying language. Moreover, the same holds for step (3), where the $\surd$-transitions that were part of the strongly connected component are deleted. Thus, $L^\omega(B) = L^\omega(A)$. □

Finally, we state the main result of this section, which is an immediate consequence of our previous considerations.

Theorem 14. *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is a $\surd$-stretchy language accepted by a tick automaton if and only if the language untick(L) is accepted by an aperiodic timed automaton, where untick$(L) := \{\, untick(w_\surd) \mid w_\surd \in L \,\}$.*

4.3 Application to Complementation

We show that the (discrete) languages obtained from timed automata are closed under complementation, thus reproving Wilke's result [9] by automata theoretical constructions only – observe that the complementation of timed languages is done with respect to timed words, where the time stamp sequence is monotonously increasing.

Theorem 15. *The classes of languages obtained from timed automata and aperiodic timed automata are closed under complementation.*

Proof. In the aperiodic case convert a timed automaton into a tick automaton according to the algorithm given in the previous section. Then we complement the Büchi automaton with any complementation algorithm (for instance the one described in [8]). This results in a tick automaton (with n states), where Lemma 13 can be applied, resulting in a Büchi automaton which is tick-cycle free. Finally, applying the conversion for a tick-deterministic and tick-cycle free Büchi automaton into an aperiodic timed automaton solves the complementation problem *via* our structural characterization. Observe that all steps are effectively constructible since the constant for the $\surd$-stretchy condition can be estimated by the size of the automaton (which is n in our case).

In the periodic case the proof is analogous and even simpler since it does not require Lemma 13. □

5 Time-Warp of Timed Languages

Based on our language theoretical characterization of timed languages we show that certain actions in a timed word can be "time-warped," i.e., all time stamps are shifted by a common distance to the future along the time axis. This leads us to the time-warp lemma, which is similar in flavour to a pumping lemma, and thus allows us to identify certain languages as not acceptable by any timed automaton – compare with the pumping lemmata of [4].

Before we introduce the time-warp lemma we need some more notations in order to simplify the presentation. For a timed word $w = \langle a_1, t_1\rangle\langle a_2, t_2\rangle \ldots$

$\langle a_i, t_i\rangle \ldots$ define its Δ-timed representation as $w_\Delta = \langle a_1, \Delta_1\rangle\langle a_2, \Delta_2\rangle \ldots \langle a_i, \Delta_i\rangle \ldots$, where $\Delta_i = t_i - t_{i-1}$ with $t_0 = 0$. Then the Δ-version of the timed language L is defined as $L_\Delta = \{\, w_\Delta \in (\Sigma \times \mathbb{N})^\omega \mid w \in L \,\}$. Now we are ready for the time-warp lemma, the proof of which immediately follows from the given characterizations of languages accepted by timed and aperiodic timed automata in Corollary 8 and Theorem 10. Thus we omit the proof.

Lemma 16 (Time-warp lemma[2]). *Let L be a the language accepted by an aperiodic timed automaton. Then there exists a constant n, such that for every word $w_\Delta = \langle a_i, \Delta_i\rangle_{i\geq 1}$, every index set $J \subseteq \{\, i \in \mathbb{N} \mid \Delta_i \geq n \,\}$, and for every function $f : J \to \mathbb{N}$ we have $w_\Delta \in L_\Delta$ if and only if time-warp$_{J,f}(w_\Delta) \in L_\Delta$, where time-warp$_{J,f}(w_\Delta)$ is defined to be the Δ-timed word $\langle a_i, \Delta'_i\rangle_{i\geq 1}$ with $\Delta'_i = \Delta_i + f(i)$, if $i \in J$, and $\Delta'_i = \Delta_i$ otherwise. The statement remains valid in case L is a language accepted by a periodic timed automaton in general, provided that the all quantified functions $f : J \to \mathbb{N}$ obey the additional property* range$(f) \subseteq \{\, kn \mid k \geq 0\}$.

With the time-warp lemma we can prove that certain languages are not acceptable by any timed automaton. For instance, consider the language of "convergent response time," which is defined as follows – the language is taken from [1]:

$$L = \{\, \langle a_i, t_i\rangle_{i\geq 1} \mid \forall i \geq 1 : a_{2i-1} = a \wedge a_{2i} = b \\ \wedge \exists c \geq 0 : \exists i \geq 1 : \forall j > i : t_{2j} < t_{2j-1} + c \,\}.$$

We show that L is *not* acceptable by any timed automaton. Assume to the contrary that the language L is accepted by some timed automaton A. Then let n be the constant mentioned in Lemma 16. Now consider the Δ-timed word $w_\Delta = \langle a, 1\rangle\langle b, n\rangle\langle a, 1\rangle\langle b, n\rangle\langle a, 1\rangle\langle b, n\rangle \ldots$, which obviously is in L_Δ. Let $J = \{\, 2i \mid i \geq 1 \,\}$ and define the function $f : \mathbb{N} \to \mathbb{N}$ by $f(i) = i \cdot n$. But then *time-warp*$_{J,f}(w_\Delta)$ is not an element of the Δ-representation of the language under consideration, since the response times clearly diverge for the warped word. Thus, language L is not acceptable by any (aperiodic) timed automaton.

For practical purposes it would be nice if at least the complement of L is acceptable by a timed automaton, as this would be enough to do timed model checking. Unfortunately, this is not the case as the closure under complementation and the following (stronger) theorem show.

Theorem 17. *Let $L \subseteq (\Sigma \times \mathbb{N})^\omega$ be a non-empty timed language such that $L \subseteq D$, where language D is implicitly defined via its Δ-representation, which is $D_\Delta = \{\, \langle a_i, \Delta_i\rangle_{i\geq 1} \mid \forall c \geq 0 : \exists i \geq 1 : \Delta_i > c \,\}$. Then L is not acceptable by any timed automaton.*

Finally, we come back to the language L of convergent response time. Obviously, language L can be parameterized according to the response time c. This

[2] It is worth mentioning, that the time-warp lemma generalizes to *dense time* semantics, i.e., real valued clocks. Note that the statements in the remaining part of this section are also valid for the dense time semantics, although in the results and arguments discrete time is used, only

leads us to languages L_c, for $c \geq 0$, which are appropriately defined. In [1] it was shown that these languages are acceptable by deterministic timed Muller automata – for a formal definition of timed Muller automata we refer to [1] – and moreover it was conjectured that these languages are not acceptable by any deterministic timed (Büchi) automaton. The theorem given below solves this conjecture.

Theorem 18. *Let $c \geq 2$. Then the timed language L_c of constant response time c is not acceptable by any* deterministic *timed automaton.*

6 Conclusions

We have given structural and language theoretical characterizations for regular discrete timed languages, in the periodic as well as in the aperiodic case by means of introducing so-called tick automata and tick languages. The characterizations have several applications and furthermore we have developed the time-warp lemma, a tool very similar to a pumping lemma which can be conveniently used in order to show that certain languages can not be accepted by (aperiodic) timed automata. We hope that these results contribute to a more basic theory for timed languages as envisioned in [3].

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. R. Alur and T. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory in Practice*, number 600 in LNCS. Springer, 1992.
3. Eugene Asarin. Challenges in timed languages: From applied theory to basic theory? *EATCS Bulletin*, 83:106–120, June 2004. Appeared in The Concurrency Column.
4. D. Beauquier. Pumping lemmas for timed automata. In *Proc. of FOSSACS '98*, number 1378 in LNCS, pages 81–94. Springer, January 1998.
5. B. Berard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.
6. Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5(4):371–404, 2000.
7. Th. A. Henzinger, P. W. Kopke, and H. Wong-Toi. The expressive power of clocks. In *Proc. of ICALP '95*, number 944 in LNCS, pages 417–428, Springer, July 1995.
8. A. Pnueli and M. Vardi. Automata-theoretic approach to automated verification – lecture notes. `http://www.cs.rice.edu/~vardi/av.html`, 1999.
9. Th. Wilke. Specifying time state sequences in powerful logics and timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 863 in LNCS, pages 694–715. Springer, 1994.

Monotone Deterministic RL-Automata Don't Need Auxiliary Symbols*

Tomasz Jurdziński[1], František Mráz[2], Friedrich Otto[3], and Martin Plátek[2]

[1] Institute of Computer Science, University of Wrocław
51-151 Wrocław, Poland
tju@ii.uni.wroc.pl
[2] Charles University, Faculty of Mathematics and Physics
Department of Computer Science, Malostranské nám. 25
118 00 PRAHA 1, Czech Republic
mraz@ksvi.ms.mff.cuni.cz, platek@ksi.ms.mff.cuni.cz
[3] Fachbereich Mathematik/Informatik, Universität Kassel
34109 Kassel, Germany
otto@theory.informatik.uni-kassel.de

Abstract. It is known that for monotone deterministic one-way restarting automata, the use of auxiliary symbols does not increase the expressive power. Here we show that the same is true for deterministic two-way restarting automata that are right- or left-monotone. Actually in these cases it suffices to admit delete operations instead of the more general rewrite operations. In addition, we characterize the classes of languages that are accepted by these types of two-way restarting automata by certain combinations of deterministic pushdown automata and deterministic transducers.

1 Introduction

The original motivation for introducing the restarting automaton in [5] was the desire to model the so-called *analysis by reduction* of natural languages. In fact, many aspects of the work on restarting automata are motivated by the basic tasks of computational linguistics. The notions developed in the study of restarting automata give a rich taxonomy of constraints for various models of analysers and parsers. There are already several programs that are being used in Czech and German (corpus) linguistics that are based on the idea of restarting automata (cf., e.g., [9, 13]).

A (two-way) restarting automaton, RLWW-automaton for short, is a device M with a finite-state control, a flexible tape, and a read/write window of a fixed size. This window moves along the tape which contains a word delimited by sentinels executing move-right and move-left instructions until its control decides

* The work of the first and third authors was supported by a grant from the Deutsche Forschungsgemeinschaft. It was performed while T. Jurdziński was visiting the University of Kassel. The second and forth authors were partially supported by the Grant Agency of the Czech Republic under Grant-No. 201/04/2102 and by the program 'Information Society' under project 1ET100300517

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 284–295, 2005.
© Springer-Verlag Berlin Heidelberg 2005

(nondeterministically) that the content of the window should be rewritten by some shorter string, in this way shortening the tape. In general, the new string may contain some auxiliary (that is, non-input) symbols. After a rewrite, M can continue to move its window until it either halts and accepts, or halts and rejects, or restarts, that is, it places its window over the left end of the tape, and reenters the initial state, in this way reaching a 'restarting configuration.' Each part of a computation of M between an initial configuration or a restarting configuration and the next restarting configuration is called a 'cycle.' Thus, each computation of M can be described through a sequence of cycles. In fact, M cannot only be considered as a device for accepting a language, but it can also be interpreted as a 'rewriting system,' as each cycle replaces a factor of its initial tape content by a shorter factor, in this way performing a rewrite of the tape content.

Also various restricted versions of the restarting automaton have been considered. The one-way restarting automaton, RRWW-automaton for short, does not have any move-left instructions, and the RWW-automaton is an RRWW-automaton that is in addition required to perform a restart step immediately after executing a rewrite operation. Furthermore, the deterministic variants of all these types of restarting automata have been considered, and a *monotonicity* property was introduced for restarting automata. It is based on the idea that from one cycle to the next in a computation, the actual place where a rewriting is performed does not increase its distance from the *right* end of the tape. The (right-) monotone restarting automata essentially model bottom-up one-pass parsers. Monotonicity conditions are of interest as monotone reductions play a significant role in the syntax of natural languages [6].

Finally, in [14] the notion of *left-monotonicity* was considered. This is based on the idea that from one cycle to the next in a computation, the actual place where a rewriting is performed does not increase its distance from the *left* end of the tape. Although the notions of monotonicity and left-monotonicity seem to be symmetric to each other, it turned out that for deterministic one-way restarting automata, these notions lead to completely different forms of behaviour. This stems from the fact that a restarting automaton starts each cycle of a computation at the left end of the tape. Hence, a monotone computation proceeds similarly to that of a pushdown automaton, while a left-monotone computation essentially processes the given input from right to left, rescanning the still to be processed prefix of the tape content in each cycle. For two-way restarting automata, however, the notions of left-monotonicity and of right-monotonicity are in fact symmetric.

While for nondeterministic restarting automata each two-way variant is just as powerful as the corresponding one-way variant [12, 14], the ability to move the window in both directions does increase the power of deterministic restarting automata. In particular, there exist monotone deterministic two-way restarting automata (even without auxiliary symbols) that accept languages that are not deterministic context-free [12], while it is known that all the different types of monotone deterministic one-way restarting automata accept the same class of languages, the class DCFL of deterministic context-free languages [6].

Here we investigate the expressive power of (right- and left-) monotone deterministic two-way restarting automata in more detail. After restating the basic definitions in Section 2, we will show in Section 3 that right- as well as left-monotone deterministic two-way restarting automata with auxiliary symbols are not more expressive than right- or left-monotone deterministic two-way restarting automata, respectively, that can only perform delete operations instead of general rewrite operations. The proof of this result entails a characterization of the class of languages that are accepted by left-monotone deterministic two-way restarting automata in terms of a combination of deterministic transducers and deterministic pushdown automata. In Section 4 we discuss some possible consequences and extentions of our results.

2 Definitions and Notation

For an alphabet Δ, we denote by Δ^+ the set of non-empty words over Δ, while Δ^* denotes the set of all words over Δ including the empty word, which we denote by λ. For a word x, $|x|$ denotes the length of x, and for an integer $i \geq 0$, $\Delta^{\leq i} := \{ x \in \Delta^* \mid |x| \leq i \}$. By x^R we denote the *reversal* of x, for a language L, $L^R := \{ x^R \mid x \in L \}$, and for a class of languages $\mathcal{C}$, $\mathcal{C}^R := \{ L^R \mid L \in \mathcal{C} \}$. Finally, for a word x and an integer i, where $1 \leq i \leq |x|$, $x[i]$ denotes the i-th letter of x.

If $\Delta_1, \ldots, \Delta_d$ $(d > 1)$ are alphabets, then $\Delta := \Delta_1 \times \Delta_2 \times \ldots \times \Delta_d$ denotes the alphabet that is obtained as the cartesian product of these alphabets. If $x_i \in \Delta_i^n$ $(1 \leq i \leq d)$ are words of equal length $n \geq 0$, then $(x_1, \ldots, x_d)$ will denote the word w from Δ^n that satisfies $w[i] = (x_1[i], \ldots, x_d[i])$ for all $i = 1, \ldots, n$. Further, by π_i $(1 \leq i \leq d)$ we denote the projection $\pi_i : \Delta \to \Delta_i$ that maps each letter onto its i-th component. Clearly π_i yields a morphism from Δ^* onto Δ_i^*.

Next we describe the type of restarting automaton we will be dealing with. We only give an informal description. The technical details can be found in [10].

A *two-way restarting automaton*, RLWW-automaton for short, is a nondeterministic machine M with a finite-state control Q, a flexible tape, and a read/write window of a fixed size $k \geq 1$. The work space is limited by the left sentinel ¢ and the right sentinel \$, which cannot be removed from the tape. In addition to the input alphabet Σ, the tape alphabet Γ of M may contain a finite number of so-called auxiliary symbols. The behaviour of M is decribed by a transition relation δ that associates to a pair (q, u) consisting of a state q and a possible content u of the read/write window a finite set of possible transition steps. There are five types of transition steps:

1. A *move-right step* (MVR) causes M to shift the read/write window one position to the right and to change the state. However, the read/write window cannot move across the right sentinel \$.
2. A *move-left step* (MVL) causes M to shift the read/write window one position to the left and to change the state. However, the read/write window cannot move across the left sentinel ¢.

3. A *rewrite step* causes M to replace the content u of the read/write window by a shorter string v, thereby shortening the tape, and to change the state. Further, the read/write window is placed immediately to the right of the string v.
4. A *restart step* causes M to place its read/write window over the left end of the tape, so that the first symbol it sees is the left sentinel ¢, and to reenter the initial state q_0.
5. An *accept step* causes M to halt and accept.

If $\delta(q,u) = \emptyset$ for some pair (q,u), then M necessarily halts, and we say that M *rejects* in this situation.

A *configuration* of M is a string $\alpha q \beta$ where q is a state, and either $\alpha = \lambda$ and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0¢w\$$, where q_0 is the initial state and $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0¢w\$$ is an *initial configuration*. Thus, initial configurations are a particular type of restarting configurations. By $\vdash_M$ we denote the *single-step computation relation* that M induces on the set of configurations, and $\vdash_M^*$ denotes the resulting *computation relation*.

In general, the automaton M is *nondeterministic*, that is, there can be two or more instructions with the same left-hand side (q,u). If this is not the case, the automaton is *deterministic*.

We observe that any finite computation of a two-way restarting automaton M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration, the window moves along the tape performing MVR, MVL, and rewrite operations until a restart operation is performed and thus a new restarting configuration is reached. If no further restart operation is performed, any finite computation necessarily finishes in a halting configuration – such a phase is called a *tail*. We require that M performs *exactly one* rewrite operation during any cycle – thus each new phase starts on a shorter word than the previous one. During a tail at most one rewrite operation may be executed.

An input *word* $w \in \Sigma^*$ *is accepted by* M, if there is a computation which, starting with the initial configuration $q_0¢w\$$, finishes by executing an accept instruction. By $L(M)$ we denote the language consisting of all words accepted by M; we say that M *recognizes (accepts) the language* $L(M)$.

Now we define those subclasses of RLWW-automata that are relevant for our investigation. An RRWW-*automaton* is an RLWW-automaton which does not use any MVL instructions. An RWW-*automaton* is an RRWW-automaton which restarts immediately after rewriting, that is, for such an automaton each rewrite transition is immediately followed by a restart transition. An RLW-automaton is an RLWW automaton which does not use auxiliary symbols. An RL-automaton is an RLW-automaton whose rewrite instructions can be viewed as deletions, that is, if $(q',v) \in \delta(q,u)$, then v is a (scattered) subword of u. Analogously, RRW- and RR-automata are obtained from RRWW-automata and RW- and R-automata from RWW-automata. We use the prefix det- to denote classes of *deterministic*

restarting automata. Further, for each type X of automata, we denote the class of languages that are accepted by automata from that class by $\mathcal{L}(\mathsf{X})$.

Next we turn to the various notions of monotonicity for restarting automata. The computation of a restarting automaton proceeds in cycles, where each cycle contains exactly one rewrite step. Thus, each cycle C contains a unique configuration $\alpha q \beta$ in which a rewrite instruction is applied. The number $|\beta|$ is called the *right distance* of C, denoted by $D_r(C)$, and $|\alpha|$ is the *left distance* of C, denoted by $D_l(C)$.

A *sequence of cycles* $S = (C_1, C_2, \cdots, C_n)$ is called (*right-*) *monotone* if $D_r(C_1) \geq D_r(C_2) \geq \ldots \geq D_r(C_n)$, and that it is called *left-monotone* if $D_l(C_1) \geq D_l(C_2) \geq \ldots \geq D_l(C_n)$. A computation is (*right-*) *monotone* or *left-monotone*, respectively, if the corresponding sequence of cycles is (right-) monotone or left-monotone. Observe that the tail of the computation does not play any role here. Finally, a restarting automaton M is called (*right-*) *monotone* or *left-monotone*, respectively, if all its computations that begin with an initial configuration are (right-) monotone or left-monotone. The prefixes mon- and left-mon- are used to indicate the various classes of right- and left-monotone restarting automata. Right-monotonicity of restarting automata is decidable [4], and it can be shown analogously that left-monotonicity is decidable.

Theorem 1. [6, 12]

(a) $\mathsf{CFL} = \mathcal{L}(\mathsf{mon\text{-}RWW}) = \mathcal{L}(\mathsf{mon\text{-}RRWW}) = \mathcal{L}(\mathsf{mon\text{-}RLWW})$.
(b) $\mathsf{DCFL} = \mathcal{L}(\mathsf{det\text{-}mon\text{-}R}) = \mathcal{L}(\mathsf{det\text{-}mon\text{-}RRWW}) \subsetneq \mathcal{L}(\mathsf{det\text{-}mon\text{-}RLWW})$.

The notion of left-monotonicity for restarting automata has been introduced only recently. Here the following results are known.

Theorem 2. [7, 14]

(a) $\mathsf{CFL} = \mathcal{L}(\mathsf{left\text{-}mon\text{-}R(R)WW}) = \mathcal{L}(\mathsf{left\text{-}mon\text{-}RLWW})$.
(b) $\mathsf{DCFL}^R \subsetneq \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}R(R)WW}) = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLWW})$.

Unlike for one-way restarting automata, the notions of left-monotonicity and (right)-monotonicity are completely symmetric for deterministic two-way restarting automata.

Lemma 1. *For each* $\mathsf{X} \in \{\lambda, \mathsf{W}, \mathsf{WW}\}$,

$$\mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLX}) = (\mathcal{L}(\mathsf{det\text{-}mon\text{-}RLX}))^R.$$

Proof. We only prove the inclusion from left to right, the other one being symmetric. Let $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLX})$, and let M be a det-left-mon-RLX-automaton for the language L. We have to verify that $L^R \in \mathcal{L}(\mathsf{det\text{-}mon\text{-}RLX})$. Based on M we construct a det-mon-RLX-automaton M' for the language L^R. Given an input word x, M' simulates the computation of M on input x^R cycle by cycle. In each cycle M' first moves its read/write window to the right delimiter, then it enters the initial state of M and simulates M step by step, replacing MVR steps by MVL steps and vice versa. It follows that M' accepts the language L^R, and that M' is monotone, if M is left-monotone. □

3 Right- and Left-Monotone Deterministic RL-Automata

In this section we derive our main result stating that right- and left-monotone deterministic two-way restarting automata don't need auxiliary symbols.

Theorem 3.

(a) $\mathcal{L}(\text{det-mon-RLWW}) = \mathcal{L}(\text{det-mon-RLW}) = \mathcal{L}(\text{det-mon-RL})$.
(b) $\mathcal{L}(\text{det-left-mon-RLWW}) = \mathcal{L}(\text{det-left-mon-RLW}) = \mathcal{L}(\text{det-left-mon-RL})$.

Part (a) is an immediate consequence of part (b) and Lemma 1. Thus, it remains to prove the equalities in (b). To derive them we will make use of a characterization of the language class $\mathcal{L}(\text{det-left-mon-RLWW})$ in terms of deterministic transducers (det-GSM, for short) and deterministic context-free languages. A det-GSM G is a deterministic finite-state acceptor with output. Formally it is specified by a 6-tuple $G = (Q, \Sigma, \Delta, \delta, q_0, F)$, where Q, Σ, and Δ are a set of states, an input alphabet, and an output alphabet, respectively, δ is a partial mapping from $Q \times \Sigma$ to $Q \times \Delta^*$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. If G is in state q scanning the input symbol a, and if $\delta(q, a) = (p, w)$, then in the next step G will enter state p, move its head one step to the right, and emit (output) the string w. With this det-GSM we can associate a partial mapping $G : \Sigma^+ \to \Delta^*$ as follows. Let $x \in \Sigma^+$. If G ends up in a final state when processing the input x starting from its initial state q_0, then $G(x) \in \Delta^*$ denotes the concatenation of all consecutive words that G outputs during this process [3].

We associate a deterministic transducer with every deterministic RRWW-automaton. Let M be a deterministic RRWW-automaton with state set Q and input alphabet Σ. Without loss of generality we can assume that during each cycle and also during the tail of each computation M scans its tape completely from left to right before it restarts, accepts, or rejects, respectively. Hence, given a word $w \in \Sigma^+$ as input for which M performs at least one rewrite step, M will first execute a sequence of transitions of the form

$$q_0 ¢ w \$ = q_0 ¢ w_1 u w_2 \$ \vdash_M^* \$ w_1 p_1 u w_2 \$ \vdash_M ¢ w_1 v p_2 w_2 \$ \vdash_M^* ¢ w_1 v w_2 p_3 \$,$$

where q_0 is the initial state, p_1, p_2, p_3 are states, $\delta(p_1, u) = (p_2, v)$ is the rewrite step applied during this cycle, and $|u| = k$. (The case that the first rewrite step is applied to a suffix of w of length smaller than k can be treated in an analogous manner).

For $i = k - 1, \ldots, |w|$, we associate a state q_i of M with the letter $w[i]$ as follows. For all i satisfying $k - 1 \leq i \leq |w_1 u|$ or $|w_1 u| + k \leq i \leq |w|$, we take q_i to denote the state of M during the above sequence at that moment when the rightmost position of the read/write window contains the letter $w[i]$, and for all values of i satisfying $|w_1 u| < i < |w_1 u| + k$, we take q_i to be state p_1. The latter alternative corresponds to the fact that, after performing the rewrite transition $u \to v$, the read/write window of M jumps to the right such that its rightmost position contains the k-th letter of w_2, that is, the first $k - 1$ letters of w_2 never

occur at that position in the read/write window. Finally, for $i = 1, \ldots, k-2$, we take $q_i := \bot$, where $\bot \notin Q$ is a new (dummy) symbol, as the rightmost position of the read/write window is never on a position $i < k-1$ (ignoring inputs shorter than k). The sequence $q_1, \ldots, q_{|w|}$ will be denoted by $\mathsf{trace}_M(w)$.

As M is a deterministic RRWW-automaton, there exists a deterministic transducer G_M that, given a word $w \in \Sigma^n$ as input, will produce the output

$$(w[1], q_1)(w[2], q_2) \ldots (w[n], q_n) \in (\Sigma \times (Q \cup \{\bot\}))^n,$$

that is, for each language $L \subseteq \Sigma^+$,

$$G_M(L) = \{\, y \in (\Sigma \times (Q \cup \{\bot\}))^+ \mid \pi_1(y) \in L \text{ and } \pi_2(y) = \mathsf{trace}_M(\pi_1(y)) \,\}.$$

Thus, $G_M(L)$ contains the words of L, 'enriched' with information on the behaviour of M during the first cycle of its computation on that particular input. The following result, which is a slight generalisation of Lemma 3.8 of [11], establishes a connection between left-monotone deterministic RRWW-automata, deterministic transducers, and DCFL.

Proposition 1.
For each left-monotone det-RRWW*-automaton* M, $G_M(L(M)) \in \mathsf{DCFL}^R$.

Proof. Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be a deterministic RRWW-automaton that is left-monotone. As $\mathsf{DCFL} = \mathcal{L}(\text{det-mon-RRWW})$, it suffices to construct a right-monotone deterministic RRWW-automaton $M' := (Q', \Omega, \Gamma', ¢, \$, q_0', k', \delta')$ for the language $(G_M(L(M)))^R$, where $\Omega := \Sigma \times (Q \cup \{\bot\})$ and $k' := 2k + 1$. This automaton will simulate M cycle by cycle.

Recall that each cycle of a computation of M consists of three phases:

(i) a MVR-phase, in which M scans a prefix of the current tape content from left to right using MVR instructions only;
(ii) a rewrite-phase, in which M applies a rewrite step;
(iii) another MVR-phase, in which M scans the suffix of the tape content to the right of the position where the rewrite step took place. This phase ends with either a restart step, or an accept instruction, or a reject when the read/write window contains only the right delimiter \$.

Let $C_1, C_2, \ldots, C_m$ be the sequence of cycles of a computation of M. As M is left-monotone, we have $D_l(C_1) \geq D_l(C_2) \geq \ldots \geq D_l(C_m)$. On the other hand, as M is deterministic, we have $D_l(C_{i+1}) > D_l(C_i) - k$, as M cannot make a rewrite step in cycle C_{i+1} while its read/write window is still completely on the prefix of the current tape content that was scanned in phase (i) of the previous cycle, that is, $D_l(C_i) - k + 1 \leq D_l(C_{i+1}) \leq D_l(C_i) \quad (1 \leq i < m)$.

Now we describe the behaviour of the right-monotone deterministic RRWW-automaton M'. For an input $y \in \Omega^+$, it simulates the computation of M for the input $w := \pi_1(y)^R$. Each cycle of a computation of M' consists of three phases analogous to the corresponding cycle of M.

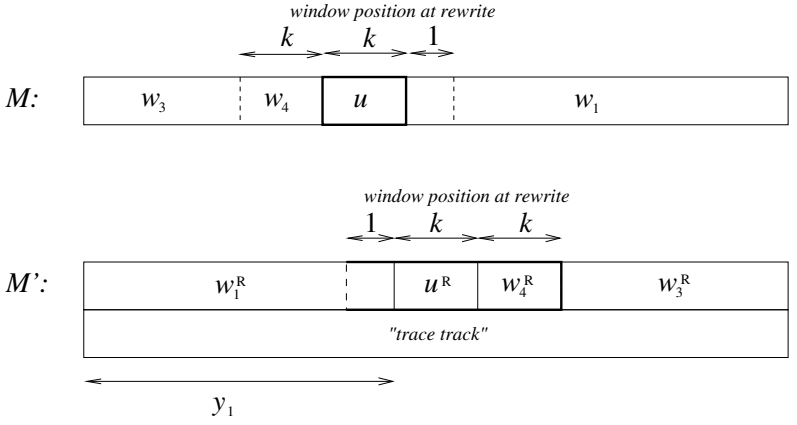
(1) M' first performs a number of MVR steps, scanning a prefix of y. This corresponds to phase (iii) of the corresponding cycle of M's computation on input w, and it continues until the reversal of the left-hand side of a rewrite step of M is found, and the state stored with the leftmost of these letters indicates that M would now execute a rewrite step. While making these MVR steps M' executes the following computations:

(a) As long as the symbols read are input symbols, it checks that the states stored in the second components describe the reverse of a valid sequence of state transitions of M that is compatible with the letters from Σ stored in the first components and that correspond to a third phase of a cycle of M. This checking is abandoned if and when M' encounters a symbol that indicates by a special mark that a rewrite has been performed before (see (2) below).

(b) For the current position of its read/write window, M' determines the following two sets of states of M, where y_1 is the prefix that has been scanned so far (including the first symbol of the current content of the read/write window), and $w_1 := \pi_1(y_1)^R$:

$$Q_+(w_1) := \{\, q \in Q \mid ¢qw_1\$ \vdash_M^{\mathrm{MVR}^*} \mathrm{ACCEPT} \,\},$$
$$Q_{\mathrm{rs}}(w_1) := \{\, q \in Q \mid ¢qw_1\$ \vdash_M^{\mathrm{MVR}^*} \mathrm{RESTART} \,\},$$

where $\vdash_M^{\mathrm{MVR}^*}$ denotes a finite sequence of MVR steps of M.



(2) The automaton M' uses the part from positions 2 to $k+1$ of its read/write window to simulate the read/write window of M. The part from positions $k+2$ to $2k+1$ serves as a kind of look-ahead, in which M' sees those symbols (in reverse) that M saw in the previous k positions (see the diagram above). When M' contains (the reversal of) the lefthand side of the actual rewrite step of M in positions 2 to $k+1$ of its read/write window, then it can determine from the state stored with the $(k+2)$-nd letter in its read/write window the actual state of M, and it can verify that M would now perform a rewrite step. Observe that due to the fact that M is left-monotone, the prefix w_3w_4 to the left of the current position of the read/write window of

M has not yet been rewritten in any way, that is, these letters are still from the original input. Now M' simulates the rewrite step of M, using a special auxiliary symbol for replacing the last letter of w_1^R to indicate for future cycles (see (1)) that a rewrite has been performed. The first position of the read/write window of M' still contains the first letter of the suffix w_1 of the tape content of M. Hence, from the state that M enters after performing the rewrite step and the sets $Q_+(w_1)$ and $Q_{\mathrm{rs}}(w_1)$ associated with this particular letter (see (1)), M' can determine the outcome of phase (iii) of the current cycle of M. Accordingly, M' now knows whether M would accept or reject in this cycle.

(3) Finally M' scans the remaining suffix of its tape content. As long as the symbols read are input symbols, it checks that the states stored in the second components describe the reverse of a valid sequence of state transitions of M that is compatible with the letters from Σ stored in the first components and that correspond to a first phase of a cycle of M. If all these tests are successful, then M' restarts or accepts, if M would restart or accept, respectively. Otherwise, M' rejects.

It follows that M' does indeed accept the language $(G_M(L(M)))^R$, and that M' is right-monotone. □

As a direct consequence of Theorem 2(b) and the above proposition, we see that, for each language $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLWW})$, there exists a deterministic transducer G such that $G(L)^R \in \mathsf{DCFL}$. In order to prove Theorem 3(b), we now derive a technical result which may be seen as the converse of Proposition 1. However, it is stronger than that, as it applies to det-left-mon-RL-automata.

Lemma 2. *Let G be a deterministic transducer with input alphabet Σ, and let $L \subseteq \Sigma^+$. If $G(L)^R \in \mathsf{DCFL}$, then there exists a* det-left-mon-RL*-automaton M that accepts the language L.*

Together with the above proposition this lemma implies Theorem 3(b). Actually we obtain the following characterization.

Corollary 1. *For a language $L \subseteq \Sigma^+$, the following statements are equivalent:*
(a) *There exists a deterministic transducer G such that $G(L)^R \in \mathsf{DCFL}$.*
(b) $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL})$.
(c) $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLW})$.
(d) $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLWW})$.

It remains to prove Lemma 2. Let $G = (Q, \Sigma, \Delta, \delta, q_{start}, F)$ be a deterministic transducer, and let L be a language over Σ such that $G(L)^R \in \mathsf{DCFL}$.

In order to avoid messy technical details, we divide the proof into two parts. In the first part we replace G by a 'uniform' det-GSM $\tilde{G}$ that produces one output letter for each input letter read. This det-GSM will have output alphabet $\tilde{\Delta} := (\Sigma \times Q \times \Delta^{\leq p})$, where p is the maximal length of any output word that G can produce in a single step. For each input symbol read, $\tilde{G}$ will output the input symbol, the state reached by the current transition, and the output produced

by G. Formally, $\tilde{G} := (Q, \Sigma, \tilde{\Delta}, \tilde{\delta}, q_{start}, F)$, where, for each $q \in Q$ and $a \in \Sigma$, if $\delta(q,a) = (q', y) \in Q \times \Delta^*$, then $\tilde{\delta}(q,a) := (q', (a, q', y)) \in Q \times \tilde{\Delta}$.

Let $\pi_1 : \tilde{\Delta}^* \to \Sigma^*$ be the morphism that is induced by mapping each letter $(a, q, y) \in \tilde{\Delta}$ onto its first coordinate a, and let $\pi_3 : \tilde{\Delta}^* \to \Delta^*$ be the morphism that is induced by mapping each letter $(a, q, y) \in \tilde{\Delta}$ onto the word $y \in \Delta^*$. From the above definition it follows immediately that $\pi_1(\tilde{G}(x)) = x$ and $\pi_3(\tilde{G}(x)) = G(x)$ holds for each input x that is accepted by G.

Proposition 2. *If $G(L)$ is in DCFL^R, then so is $\tilde{G}(L)$.*

Proof. Let $(x, q, y) \in \tilde{\Delta}^+$. In order to check whether $(x, q, y) \in \tilde{G}(L)$, we need to verify that $x \in G(L)$ and that q and y are consistent with x and G. To this end we combine a deterministic pushdown automaton that recognizes $G(L)^R$ with a deterministic one-way finite-state acceptor that, given (x^R, q^R, y^R) as input, checks whether $(x[i], q[i], y[i])$ and $(x[i+1], q[i+1], y[i+1])$, $1 \leq i < |x|$, are consistent with the definition of the det-GSM G. Moreover, it must check whether the last letter of (x^R, q^R, y^R) is of the form $(x[1], q[1], y[1])$ such that $\delta(q_{start}, x[1]) = (q[1], y[1])$ holds. □

As $\mathsf{DCFL} = \mathcal{L}(\text{det-mon-R})$ (Theorem 1(b)), there exists a monotone det-R-automaton M_1 that accepts the language $\tilde{G}(L)^R$. From M_1 we immediately obtain a det-RL-automaton M_2 for the language $\tilde{G}(L)$: in each cycle M_2 first moves its window to the right delimiter, checking that the actual tape content is of the form $\tilde{G}(x)$ for some word $x \in \Sigma^+$. In the negative it rejects immediately, while in the affirmative it then simulates the current cycle of M_1, replacing MVR steps by MVL steps. We can even require that M_2 aborts this simulation and rejects when it comes upon a rewrite transition that would transform a word of the form $\tilde{G}(x)$ into a word that does not belong to the set $\tilde{G}(\Sigma^+)$. As M_1 is monotone, it follows that M_2 is left-monotone. Hence, we see that $\tilde{G}(L) \in \mathcal{L}(\text{det-left-mon-RL})$.

Finally we present a left-monotone det-RL-automaton M_3 that, given a word $x \in \Sigma^+$ as input, simulates the computation of M_2 on $\tilde{G}(x)$. Notice that the information that is stored in the second and third coordinates of $\tilde{G}(x)$ is missing when the input x is being processed. Thus, M_3 needs to 'recover' this information from x. For this the following two observations are useful.

First of all, it is rather obvious that there exists a deterministic finite-state acceptor A_G that simulates the deterministic transducer G in such a way that, after processing the i-th symbol of an input word $x \in \Sigma^+$, where $1 \leq i \leq |x|$, the state of A_G contains the i-th symbol of the word $\tilde{G}(x)$. Accordingly, when scanning the given input x from left to right, the det-RL-automaton M_3 can easily recover that part of $\tilde{G}(x)$ that corresponds to the part of x that is currently inside the read/write window.

For 'recovering' this information also in the case that the read/write window is moving from right to left, we need the following result, which is taken from [1], pages 212–213 (see also Lemma 3 of [2]). In fact, this lemma is crucial for our construction.

Lemma 3. *Let A be a deterministic finite-state acceptor. For each word x and each integer i, $1 \leq i \leq |x|$, let $q_A(x,i)$ be the state of A after processing the prefix of length i of x. Then there exists a deterministic two-way finite-state acceptor A' such that, for each input x and each $i \in \{2,3,\ldots,|x|\}$, the following condition is satisfied:*

- *if A' starts its computation on x in state $q_A(x,i)$ with its head on $x[i]$, then A' finishes its computation in state $q_A(x,i-1)$ with its head on $x[i-1]$.*

Hence, there exists a deterministic two-way finite-state acceptor A'_G that, starting in state $q_{A_G}(x,i)$ with tape content x and its head on the symbol $x[i]$, finishes its computation in state $q_{A_G}(x,i-1)$ with its head on $x[i-1]$. Using A'_G the det-RL-automaton M_3 is able to recover the information on $\tilde{G}(x)$ from x while moving its read/write window from right to left.

Now we are prepared for describing the restarting automaton M_3. Each cycle of M_2 on input $\tilde{G}(x)$ is simulated by a cycle of M_3 on input x as follows:

(1) First M_3 simulates the computation of the deterministic finite-state acceptor A_G until it reaches the right delimiter. In this way it computes the k rightmost positions of $\tilde{G}(x)$.
(2) Next M_3 simulates the behaviour of M_2 step by step until it reaches the configuration in which M_2 applies the rewrite step of the current cycle. Each time it must simulate a MVL step, it runs the automaton A'_G in order to recover the value of $\tilde{G}(x)$ at the new leftmost position in the read/write window. Note that at the beginning of the simulation M_3 knows the part of $\tilde{G}(x)$ that is contained in the read/write window (see (1)). Hence, by using A'_G it is able to satisfy this condition for each successive step.
(3) Finally, M_3 simulates the current rewrite step of M_2 (that is, it removes the appropriate symbols).

Recall that M_2 checks in each cycle that its current tape content is indeed of the form $\tilde{G}(w)$. Thus, given an input $x \in \Sigma^+$, M_3 simulates the computation of M_2 on $\tilde{G}(x)$ cycle by cycle. Further, M_3 accepts the given input x if and only if M_2 accepts the input $\tilde{G}(x)$. Hence, it follows that $L(M_3) = L$. As M_2 is left-monotone, M_3 is left-monotone, too. This completes the proof of Lemma 2.

4 Conclusions

For modelling the analysis of natural languages those variants of restarting automata are preferable that do not use auxiliary symbols, as their operation can be interpreted as reducing a given sentence to a 'simple' form within the same language. Here we have seen that at least for deterministic two-way restarting automata that are left- or right-monotone, the variant without auxiliary symbols is as powerful as the variant that is allowed to use auxiliary letters. In fact, each det-left-mon-RLWW-automaton can be transformed into an equivalent det-left-mon-RL-automaton, as each step of this transformation process is effective, and analogously for right-monotonicity. Actually, our main result carries

over to the class of deterministic two-way restarting automata that are simultaneously left- and right-monotone. However, the proof for this case given in [8] is completely different from the proof for the left-monotone case presented here as it is based on pumping techniques. Finally, we should remark that the results of this paper do not carry over to deterministic two-way restarting automata in general.

Acknowledgement. The authors thank M. Chytil and V. Jákl for pointing them to the work of Aho, Hopcroft, and Ullman on the reversibility of deterministic two-way transducers.

References

1. A. V. Aho, J. E. Hopcroft, J. D. Ullman. A General Theory of Translation. *Math. Systems Theory* 3 (1969) 193-221.
2. J. E. Hopcroft and J. D. Ullman. An approach to a unified theory of automata. *Bell System Tech. J.* 46 (1967) 1793-1829.
3. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.
4. P. Jančar, F. Mráz, M. Plátek. Monotonicity of restarting automata. *J. Autom. Lang. Comb.*, to appear.
5. P. Jančar, F. Mráz, M. Plátek, J. Vogel. Restarting automata. In H. Reichel (ed.), *FCT'95, Proc.*, *LNCS* 965, pages 283–292. Springer, Berlin, 1995.
6. P. Jančar, F. Mráz, M. Plátek, J. Vogel. On Monotonic Automata with a Restart Operation. *J. Autom. Lang. Comb.* 4 (1999) 283-292.
7. T. Jurdziński, F. Otto, F. Mráz, and M. Plátek. On left-monotone deterministic restarting automata. In C.S. Calude, E. Calude, and M.J. Dinneen (eds.), *DLT'2004, Proc.*, *LNCS* 3340, Springer, Berlin, 2004, 249–260.
8. T. Jurdziński, F. Otto, F. Mráz, M. Plátek. Deterministic two-way restarting automata and Marcus contextual grammars. *Fund. Inform.* 64 (2005) 217–228.
9. K. Oliva, P. Květoň, R. Ondruška. The computational complexity of rule-based part-of-speech tagging. In V. Matoušek and P. Mautner (eds.), *TSD 2003, Proc.*, *LNCS* 2807, pages 82–89. Springer, Berlin, 2003.
10. F. Otto. Restarting Automata - Notes for a Course at the 3rd International PhD School in Formal Languages and Applications. *Mathematische Schriften Kassel* 6/04, Universität Kassel, 2004.
11. F. Otto and T. Jurdziński. On Left-monotone Restarting Automata, *Mathematische Schriften Kassel* 17/03, Universität Kassel, 2003.
12. M. Plátek. Two-way restarting automata and j-monotonicity. In L. Pacholski P. and Ružička (eds.), *SOFSEM'2001, Proc.*, *LNCS* 2234, pages 316–325. Springer, Berlin, 2001.
13. M. Plátek, M. Lopatková, K. Oliva. Restarting automata: motivations and applications. In M. Holzer (ed.), *Workshop 'Petrinetze' and 13. Theorietag 'Formale Sprachen und Automaten', Proc.*, Institut für Informatik, Technische Universität München, 2003, 90-96.
14. M. Plátek, F. Otto, and F. Mráz. Restarting automata and variants of j-monotonicity. In E. Csuhaj-Varjú, C. Kintala, D. Wotschke, and G. Vaszil (eds.), *Descriptional Complexity of Formal Systems, Proc. DCFS 2003* , MTA SZTAKI, Budapest, 2003, 303–312.

On Hairpin-Free Words and Languages

Lila Kari[1], Stavros Konstantinidis[2], Petr Sosík[3,⋆], and Gabriel Thierrin[4]

[1] Department of Computer Science, The University of Western Ontario, London, ON, N6A 5B7, Canada
lila@csd.uwo.ca
[2] Dept. of Mathematics and Computing Science, Saint Mary's University, Halifax, Nova Scotia, B3H 3C3, Canada
s.konstantinidis@smu.ca
[3] Institute of Computer Science, Silesian University, 74601 Opava, Czech Republic
petr.sosik@fpf.slu.cz
[4] Department of Mathematics, The University of Western Ontario, London, ON, N6A 5B7, Canada
thierrin@uwo.ca

Abstract. The paper examines the concept of hairpin-free words motivated from the biocomputing and bioinformatics fields. Hairpin (-free) DNA structures have numerous applications to DNA computing and molecular genetics in general. A word is called hairpin-free if it cannot be written in the form $xvy\theta(v)z$, with certain additional conditions, for an involution θ (a function θ with the property that θ^2 equals the identity function).
We consider three involutions relevant to DNA computing: a) the mirror image function, b) the DNA complementarity function over the DNA alphabet $\{A, C, G, T\}$ which associates A with T and C with G, and c) the Watson-Crick involution which is the composition of the previous two. We study elementary properties and finiteness of hairpin (-free) languages w.r.t. the involutions a) and c). Maximal length of hairpin-free words is also examined. Finally, descriptional complexity of maximal hairpin-free languages is determined.

Keywords: DNA computing, DNA hairpin, involution, finite automaton

1 Introduction

The primary motivation for study of hairpin-free structures in this paper arises from the areas of DNA computing and bioinformatics, where such structures are important for the design of information-encoding DNA molecules. A single strand DNA molecule can be formally described as a string over the DNA alphabet $\Delta = \{A, C, T, G\}$. These four symbols correspond to nucleotides attached to a sugar-phosphate backbone. Two single strands can bind (anneal) to each other if they have opposite polarity (the strand's orientation in space) and

⋆ Corresponding author

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 296–307, 2005.
© Springer-Verlag Berlin Heidelberg 2005

are pairwise Watson-Crick complementary: A is complementary to T, and C to G. The ability of DNA strands to anneal to each other allows for creation of various secondary structures. A *DNA hairpin* is a particular type of secondary structure important in many applications. An example of a hairpin structure is shown in Figure 1. The figure characterizes the case when θ is the Watson-Crick antimorphic involution (see the next section for exact definition).

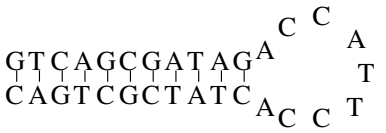


Fig. 1. A single-stranded DNA molecule forming a hairpin loop

Hairpin-like secondary structures play an important role in insertion/deletion operations with DNA. Hairpin-freedom is crucial in the design of primers for the PCR reaction [4]. Hairpins are the main tool used in the Whiplash PCR computing techniques [17]. In [19] hairpins serve as a binary information medium for DNA RAM. Last, but not least, hairpins are basic components of recently investigated "smart drugs" [1]. Therefore, in the above mentioned applications, one needs to construct (sets of) hairpin(-free) DNA molecules, or to test existing sets of DNA molecules for hairpin-freedom and study their properties. We refer e.g. to [16] for an overview of design of DNA languages without hairpins and other undesired bonds. Coding properties of hairpin-free languages have been studied in [11, 12]. Hairpins have also been studied in the context of bio-operations occurring in single-celled organisms (see the hairpin inversion operation defined as one of the three molecular operations that accomplish gene assembly in ciliates [6, 8]).

In addition, the presented results also contribute to mathematical characterization of regularities in formal words and languages. In this sense the definition of hairpin-free words can be understood as a generalization of repetition-freedom. A word u is called *hairpin-k-free* if $u = xvy\theta(v)z$ implies $|v| < k$, for a chosen involution θ. Considering the special case when $k = 1$, θ is the identity involution and y is the empty word, we obtain the square-freedom (see below).

For a general overview and fundamental results in combinatorics on words, the reader is referred to [5, 13]. If w is an nonempty word, then ww is called a square and www is called a cube. Important questions about avoiding squares and cubes in infinite words have been answered in [7]. See [14] for combinatorics on finite words. Words of the form $uvyvz$ with a bounded length of y have been studied e.g. in [3]. Unfortunately, many techniques and results known in combinatorics on words are non-applicable in the case of hairpin-free words. One of the main reasons is that in the case of an antimorphic involution, analogies of the famous defect theorem and its consequences are no longer valid.

The paper is organized as follows. Section 2 introduces basic formal concepts and definitions. In Section 3 we present the concept of hairpin-free words and languages and study their elementary properties. Problems related to the finite-

ness of hairpin-free languages are addressed in Section 4. Finally, in Section 5 we study descriptional complexity of hairpin (-free) languages with regards to possible applications.

2 Formal Language Prerequisites

We will use X to denote a finite alphabet and X^* its corresponding free monoid. The cardinality of the alphabet X is denoted by $|X|$. The empty word is denoted by 1, and $X^+ = X^* - \{1\}$. A *language* is an arbitrary subset of X^*. For a word $w \in X^*$ and $k \geq 0$, we denote by w^k the word obtained as catenation of k copies of w. Similarly, X^k is the set of all words from X^* of length k. By convention, $w^0 = 1$ and $X^0 = \{1\}$. We also denote $X^{\leq k} = X^0 \cup X^1 \cup \ldots \cup X^k$. By convention, $X^{\leq k} = \emptyset$ for $k < 0$.

A mapping $\psi : X^* \to X^*$ is called a *morphism* (*anti-morphism)* of X^* if $\psi(uv) = \psi(u)\psi(v)$ (respectively $\psi(uv) = \psi(v)\psi(u)$) for all $u, v \in X^*$, and $\phi(1) = 1$. See [9] for a general overview of morphisms. An involution $\theta : X \longrightarrow X$ is defined as a map such that θ^2 is the identity function. An involution θ can be extended to a morphism or an antimorphism over X^*. In both cases θ^2 is the identity over X^* and $\theta^{-1} = \theta$. If not stated otherwise, θ refers to an arbitrary morphic or antimorphic involution in this paper.

In our examples we shall refer to the DNA alphabet $\Delta = \{A, C, T, G\}$. By convention, DNA strands are described by strings over this alphabet in orientation from 5' to 3' end. On this alphabet several involutions of interest are defined. The simplest involution is the identity function ϵ. An antimorphic involution which maps each letter of the alphabet to itself is called a *mirror involution* and it is denoted by μ. The DNA *complementarity involution* γ is a morphism given by $\gamma(A) = T$, $\gamma(T) = A$, $\gamma(C) = G$, $\gamma(G) = C$. For example, $\epsilon(ACGCTG) = ACGCTG = \mu(GTCGCA) = \gamma(TGCGAC)$.

Finally, the antimorphic involution $\tau = \mu\gamma$ (the composite function of μ and γ, which is also equal to $\gamma\mu$), called the *Watson-Crick involution*, corresponds to the DNA bond formation of two single strands. If for two strings $u, v \in \Delta^*$ it is the case that $\tau(u)v$, then the two DNA strands represented by u, v anneal as Watson-Crick complementary sequences.

A nondeterministic finite automaton (*NFA*) is a quintuple $A = (S, X, s_0, F, P)$, where S is the finite and nonempty set of states, s_0 is the start state, F is the set of final states, and P is the set of productions of the form $sx \to t$, for $s, t \in S$, $x \in X$. If for every two productions $sx_1 \to t_1$ and $sx_2 \to t_2$ of an NFA we have that $x_1 \neq x_2$ then the automaton is called a *DFA* (deterministic finite automaton). The language accepted by the automaton A is denoted by $L(A)$. The *size* $|A|$ of the automaton A is the number $|S| + |P|$. We refer to [18] for further definitions and elements of formal language theory.

3 Involutions and Hairpins

Definition 1. *If θ is a morphic or antimorphic involution of X^* and k is a positive integer, then a word $u \in X^*$ is said to be θ-k-hairpin-free or simply hp(θ,k)-free if $u = xvy\theta(v)z$ for some $x, v, y, z \in X^*$ implies $|v| < k$.*

Notice that the words 1 and $a \in X$ are hp(θ,1)-free. More generally, words of length less than $2k$ are hp(θ,k)-free. If we interpret this definition for the DNA alphabet Δ and the Watson-Crick involution τ, then a hairpin structure with the length of bond greater than or equal to k is a word that is not hp(τ,k)-free.

Definition 2. *Denote by $hpf(\theta, k)$ the set of all hp(θ,k)-free words in X^*. The complement of $hpf(\theta, k)$ is $hp(\theta, k) = X^* - hpf(\theta, k)$.*

Notice that $hp(\theta, k)$ is the set of words in X^* which are hairpins of the form $xvy\theta(v)z$ where the length of v is at least k. It is also the case that $hp(\theta, k+1) \subseteq hp(\theta, k)$ for all $k > 0$.

Definition 3. *A language L is called θ-k-hairpin-free or simply $hp(\theta, k)$-free if $L \subseteq hpf(\theta, k)$.*

It is easy to see from the definition that a language L is hp(θ, k)-free if and only if $X^*vX^*\theta(v)X^* \cap L = \emptyset$ for all $|v| \geq k$. An analogous definition was given in [11] where a θ-k-hairpin-free language is called θ-subword-k-code. The authors focused on their coding properties and relations to other types of codes. Restrictions on the length of a hairpin were also considered, namely that $1 \leq |y| \leq m$ for some $m \geq 1$. The reader can verify that our Proposition 3 remains valid and the results in Section 5 change only slightly if we apply this additional restriction.

Example.

1. Let $X = \{a, b\}$ with $\theta(a) = b, \theta(b) = a$. Then $hpf(\theta, 1) = a^* \cup b^*$.
   This example shows that in general the product of hp$(\theta, 1)$-free words is not an hp$(\theta, 1)$-free word. Indeed, a and b are hp$(\theta, 1)$-free, but the product ab is not.
2. If $\theta = \gamma$ is the DNA complementary involution over Δ^*, then:
$$hpf(\theta, 1) = \{A, C\}^* \cup \{A, G\}^* \cup \{T, C\}^* \cup \{T, G\}^*$$
3. Let $\theta = \mu$ be the mirror involution and let $u \in hpf(\theta, 1)$. Since $\theta(a) = a$ for all $a \in X$, u cannot contain two occurrences of the same letter a. This implies that $hpf(\theta, 1)$ is finite. For example, if $X = \{a, b\}$, then:
$$hpf(\theta, 1) = \{1, a, b, ab, ba\}$$

We focus first on the important special case when $k = 1$. Observe that $hp(\theta, 1) = \bigcup_{a \in X} X^*aX^*\theta(a)X^*$. Recall also the definition of an embedding order: $u \leq_e w$ if and only if

$$u = u_1u_2 \cdots u_n,\ w = v_1u_1v_2u_2 \cdots v_nu_nv_{n+1}$$

for some integer n with $u_i, v_j \in X^*$.

A language L is called *right $\leq_e$-convex* [20] if $u \leq_e w,\ u \in L$ implies $w \in L$. The following result is well known: *All languages (over a finite alphabet) that are right $\leq_e$-convex are regular.*

Proposition 1. *The language $hp(\theta, 1)$ is right $\leq_e$-convex.*

Proof. If $u = u_1u_2 \in hp(\theta, 1)$ and $v_1, v_2, v_3 \in X^*$, then $w = v_1u_1v_2u_2v_3 \in hp(\theta, 1)$. Therefore, if $u \in hp(\theta, 1)$ and $u \leq_e w$, then w can be constructed from u by a sequence of insertions, and hence $w \in hp(\theta, 1)$. □

Let $L \subseteq X^*$ be a nonempty language and let:

$$S(L) = \{w \in X^* | u \leq_e w, u \in L\}.$$

Hence $S(L)$ is the set of all the words $w \in X^*$ that can be expressed in the form $w = x_1u_1x_2u_2 \cdots x_nu_nx_{n+1}$ with $u = u_1u_2 \cdots u_n \in L$ and $x_i \in X^*$.

Recall further that a set H with $\emptyset \neq H \subseteq X^+$ is called a *hypercode* over X^* *iff* $x \leq_e y$ and $x, y \in H$ imply $x = y$. That is, a hypercode is an independent set with respect to the embedding order.

Proposition 2. *Let θ be a morphic or antimorphic involution. Then there exists a unique hypercode H such that $hp(\theta, 1) = S(H)$.*

Proof. Let $H = \bigcup_{a \in X} a\theta(a)$, then $S(H) = \bigcup_{a \in X} X^*aX^*\theta(a)X^* = hp(\theta, 1)$. The uniqueness of H is immediate. □

Example. Consider the hypercodes for the earlier three examples.

1. For $X = \{a, b\}$ and the involution (morphic or antimorphic) $\theta(a) = b$, $\theta(b) = a$, the hypercode is $H = \{ab, ba\}$.
2. For the DNA complementarity involution γ we have $H = \{AT, TA, CG, GC\}$.
3. The mirror involution over $\{a, b\}^*$ gives the hypercode $H = \{aa, bb\}$.

Proposition 1, true for the case $k = 1$, cannot in general be extended to the case $k > 1$ as the language $hp(\theta, 2)$ is not $\leq_e$-convex. However, the weaker regularity property remains valid. Note that $hp(\theta, k) = \bigcup_{|w| \geq k} X^*wX^*\theta(w)X^*$.

Proposition 3. *The languages $hp(\theta, k)$ and $hpf(\theta, k)$, $k \geq 1$, are regular.*

Proof. One can easily derive $hp(\theta, k) \bigcup_{|w|=k} X^*wX^*\theta(w)X^*$. Every language $X^*wX^*\theta(w)X^*$with $|w| = k$ is regular, hence $hp(\theta, k)$ is a union of a finite number of regular languages. Therefore both $hp(\theta, k)$ and its complement $hpf(\theta, k)$ are regular. □

4 Finiteness of Hairpin-Free Languages

In this section we give the necessary and sufficient conditions under which the language $hpf(\theta, k)$ is finite, for a chosen $k \geq 1$. We study first the interesting special case of μ, the mirror involution, over a binary alphabet X.

Recall that $hp(\mu, k)$ is the set of all words containing two non-overlapping mirror parts of length at least k. In the next proposition we show that the longest hp$(\mu, 4)$-free word is of length 31. This also implies that the language $hpf(\mu, 4)$ is finite. The proof requires several technical lemmata whose proofs are omitted due to page limitations and can be found in [15]. In these lemmata we assume that $|X| = 2$.

Definition 4. *A* run *in a word w is a subword of w of the form c^k, with $c \in X$ and $k \geq 1$, such that $w = uc^kv$ for some word u that does not end with c, and some word v that does not start with c.*

Lemma 1. *Suppose that w is a word in $hpf(\mu, 4)$. The following statements hold true.*

1. *If a^i is any run in w then $i \leq 7$. If the run is internal then $i \leq 5$.*
2. *The word w cannot contain three different runs $a^{i_1}, a^{i_2}, a^{i_3}$ with $i_1, i_2, i_3 \geq 3$. If w contains two runs a^j and a^i with $i, j \geq 3$ then w starts with a^jba^i, or w ends with a^iba^j. Moreover not both i and j can be greater than 3.*
3. *The word w cannot contain three different internal runs a^2. If w contains two internal runs a^2 then they occur as in $\cdots ba^2ba^2b\cdots$.*
4. *The above statements also hold if we replace a with b and vice-versa.*

Lemma 2. *Suppose that a word in $hpf(\mu, 4)$ contains a subword w of the form*

$$ab^{x_1}a^{y_1}\cdots b^{x_n}a^{y_n}b,$$

with $n \geq 3$ and $x_i, y_i \geq 1$ for each i. Then there are at most three indices i such that $x_i = y_i = 1$.

Lemma 3. *Suppose that a word w is in $hpf(\mu, 4)$ and contains two runs c^j and c^i with $i, j \geq 3$ and $c \in X$. Then $|w| \leq 31$.*

Lemma 4. *Suppose that a word w is in $hpf(\mu, 4)$ and contains no two runs c^j and c^i with $i, j \geq 3$ and contains two internal runs b^2 and one internal run b^y with $y \geq 3$ and w is of the following form*

$$a^{y_0}b^{x_1}a^{y_1}\cdots b^{x_n}a^{y_n}(b^{x_{n+1}}a^{y_{n+1}}),$$

where all y_i's and x_j's are positive except possibly for y_{n+1}. Then $|w| \leq 31$.

Lemma 5. *If a word w is in $hpf(\mu, 4)$ and of the form*

$$a^{y_0}b^{x_1}a^{y_1}\cdots b^{x_n}a^{y_n}(b^{x_{n+1}}a^{y_{n+1}}),$$

such that $y_0, x_{n+1} \geq 3$, and $2 \geq y_{n+1} \geq 0$, and $2 \geq x_i, y_i > 0$ for all $i = 1, \ldots, n$, then $|w| \leq 30$. Moreover, the following word of length 30 satisfies the above premises:

$$a^7b^2ab^2ababaa^2ba^2b^7.$$

Proposition 4. *Let X be a binary alphabet. For every word $w \in X^*$ in $hpf(\mu, 4)$ we have that $|w| \leq 31$. Moreover the following word of length 31 is in $hpf(\mu, 4)$*

$$a^7ba^3bababab^2ab^2a^2b^7.$$

Proof. Without loss of generality we can assume that w starts with a. Then w would be of the form

$$a^{y_0}b^{x_1}a^{y_1}\cdots b^{x_n}a^{y_n}(b^{x_{n+1}}a^{y_{n+1}}),$$

where all y_i's and x_j's are positive except possibly for y_{n+1}. We distinguish the following cases.

Case 1: There are two runs c^i and c^j in w with $i, j \geq 3$. By Lemma 3, $|w| \leq 31$ as required.

In the next 7 cases, we assume that the first case does not hold and that there is exactly one run a^δ in w with $\delta \geq 3$.

Case 2: The run a^δ is a^{y_0} and there is a run b^i with $i \geq 3$. If $x_{n+1} \geq 3$ then Lemma 5 implies that $|w| \leq 30$. So assume that $x_{n+1} \leq 2$. If there are two internal runs b^2 in w then Lemma 4 implies that $|w| \leq 31$. So assume further that there is at most one internal run b^2. Note that if $x_{n+1} = 2$ and $y_{n+1} > 0$ then $b^{x_{n+1}}$ is the run b^2. Let g be the quantity $e|b^2a| + x_{n+1} + y_{n+1}$, where $e = 0$ if $x_{n+1} = 2$ and $y_{n+1} > 0$, and $e = 1$ if $x_{n+1} = 1$ or $y_{n+1} = 0$. Hence, $g \leq 6$. Moreover, $|w| \leq 7 + 3|ba| + 2|ba^2| + |b^5a| + g \leq 31$.

Case 3: The run a^δ is a^{y_0} and there is no run b^i with $i \geq 3$. Using again the quantity g of Case 2, we have that $|w| \leq 7 + 3|ba| + 2|ba^2| + |b^2a| + g \leq 28$.

Case 2': The run a^δ is $a^{y_{n+1}}$ and there is a run b^i with $i \geq 3$. Then the word $\mu(w)$ is of the same form as the word w is and the run b^i occurs in $\mu(w)$. Hence, Case 2 applies to $\mu(w)$ and, therefore, both $\mu(w)$ and w are of length at most 31.

Case 3': The run a^δ is $a^{y_{n+1}}$ and there is no run b^i with $i \geq 3$. Then the word $\mu(w)$ is of the same form as the word w is and no run b^i, with $i \geq 3$, occurs in $\mu(w)$. Hence, Case 3 applies to $\mu(w)$ and, therefore, both $\mu(w)$ and w are of length at most 28.

Case 4: The run a^δ is internal and there is one internal run b^j with $j \geq 3$. Then $j, \delta \leq 5$. If w contains two internal runs b^2 then Lemma 4 implies that $|w| \leq 31$. Next assume that w contains at most one internal run b^2 and consider the quantity $g = e|b^2a| + x_{n+1} + y_{n+1}$ as in Case 2. If w contains at most one internal run a^2 then

$$|w| \leq y_0 + 3|ba| + |ba^2| + |ba^5| + |b^5a| + g \leq 2 + 6 + 3 + 6 + 6 + 6 = 29.$$

Next assume further that w contains two internal runs a^2. Then Lemma 1 implies that w contains ba^2ba^2b. Also,

$$|w| \leq 2 + 6 + 2|ba^2| + 6 + 6 + g \leq 26 + g.$$

If $x_{n+1} = 2$ and $y_{n+1} > 0$ then $e = 0$ and $|w| \leq 30$. If $y_{n+1} = 0$ then $e = 1$ and $|w| \leq 31$. If $x_{n+1} = 1$ and $y_{n+1} = 1$ then $|w| \leq 31$. Finally, if $x_{n+1} = 1$ and $y_{n+1} = 2$ then w ends with aba^2, which contradicts the fact that w contains ba^2ba^2b.

Case 4': The run a^δ is internal and there is one external run b^j with $j \geq 3$. Then $y_{n+1} = 0$ and the run b^j is $b^{x_{n+1}}$, as $y_0 > 0$. Let w' be the word resulting by exchanging the letters a and b in w. Then the word $\mu(w')$ satisfies the premises of Case 2, which implies that w is of length at most 31.

Case 5: The run a^δ is internal and there is no run b^j with $j \geq 3$. Using again the quantity g of Case 2, we have that $|w| \leq y_0+3|ba|+|ba^5|+2|ba^2|+|b^2a|+g \leq 29$.
Case 6: Here the first case does not hold and there is no run a^δ with $\delta \geq 3$. If there is an internal run b^j with $j \geq 3$ then $|w| \leq y_0+3|ba|+|b^5a|+2|ba^2|+|b^2a|+g \leq 29$. If there is an external run b^j with $j \geq 3$ then $b^j = b^{x_{n+1}}$ and $y_{n+1} = 0$, and one can verify that $|w| \leq 27$. If there is no run b^j with $j \geq 3$ then one can verify that $|w| \leq 23$.

Finally, by inspection one verifies that $a^7ba^3bababab^2ab^2a^2b^7$ is indeed in $hpf(\mu, 4)$. □

Corollary 1. *Consider a binary alphabet X. Then $hpf(\mu, k)$ is finite if and only if $k \leq 4$.*

Proof. Denote $X = \{a, b\}$. By Proposition 4, the set $hpf(\mu, 4)$ is finite. Now consider the language $L_5 = (aabbab)^+$. The set of its subwords of length 5 is $Sub_5(L_5) = \{aabba, abbab, bbaba, babaa, abaab, baabb\}$. For its mirror image $\mu(L_5)$ we obtain $Sub_5(\mu(L_5)) = \{abbaa, babba, ababb, aabab, baaba, bbaab\}$. As these two sets are mutually disjoint, $L_5 \subseteq hpf(\mu, 5)$.

Finally, notice that for $k > 1$, finiteness of $hpf(\mu, k)$ implies also finiteness of $hpf(\mu, k-1)$. Hence the facts that $hpf(\mu, 4)$ is finite and $hpf(\mu, 5)$ is infinite conclude the proof. □

Proposition 5. *Let θ be a morphic or antimorphic involution. The language $hpf(\theta, k)$ over a non-singleton alphabet X is finite if and only if one of the following holds:*
(a) $\theta = \epsilon$, the identity involution;
(b) $\theta = \mu$, the mirror involution, and either $k = 1$ or $|X| = 2$ and $k \leq 4$.

Proof. (a) Let θ be a morphism. Assume first that $\theta \neq \epsilon$. Then there are $a, b \in X$, $a \neq b$, such that $\theta(a) = b$. Then $a^+ \subseteq hpf(\theta, k)$ for any $k \geq 1$, hence $hpf(\theta, k)$ is infinite.
Assume now that $\theta = \epsilon$ and let w be any word of length $\geq k|X|^k + k$. Since there exist $|X|^k$ distinct words of length k, there are at least two non-overlapping subwords of length k in w which are identical. Hence $w = xvyvz$ for some $v \in X^k$ and $x, y, z \in X^*$. Therefore $hpf(\epsilon, k)$ is finite since it cannot contain any word longer than $k|X|^k + k$.
(b) Let θ be an anti-morphism. Assuming that $\theta \neq \mu$, the same arguments as above show that $hpf(\theta, k)$ is infinite.
Assume now that $\theta = \mu$. Apparently $hpf(\mu, 1)$ is finite as shown in the examples above. For $|X| = 2$ we know that $hpf(\mu, k)$ is finite *iff* $k \leq 4$ by Corollary 1. Finally, for $|X| > 2$ and $k > 1$ the language $hpf(\mu, k)$ is infinite as it always contains the $\mathrm{hp}(\mu, 2)$-free set $(abc)^+$ (regardless to renaming the symbols). □

5 Descriptional Complexity of Hairpin(-Free) Languages

The regularity of the languages $hp(\theta, k)$ and $hpf(\theta, k)$ shown in Section 3 indicates an existence of fast algorithms deciding problems related to hairpin-

freedom. For such algorithms, a construction of automata (NFA or DFA) accepting the languages $hp(\theta, k)$ and $hpf(\theta, k)$ would be important. Therefore we investigate minimal size of these automata. We recall the following technical tools from [2], see also [10].

Definition 5. *A set of pairs of strings* $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ *is called a* fooling set *for a language* L *if for any* i, j *in* $\{1, 2, \ldots, n\}$,
(1) $x_i y_i \in L$, *and*
(2) if $i \neq j$ *then* $x_i y_j \notin L$ *or* $x_j y_i \notin L$.

Lemma 6. *Let* $\mathcal{F}$ *be a fooling set of a cardinality* n *for a regular language* L. *Then any NFA accepting* L *needs at least* n *states.*

Now we can characterize the minimal size of automata accepting languages $hp(\theta, k)$ and $hpf(\theta, k)$. We use the operator $\prod$ for catenation.

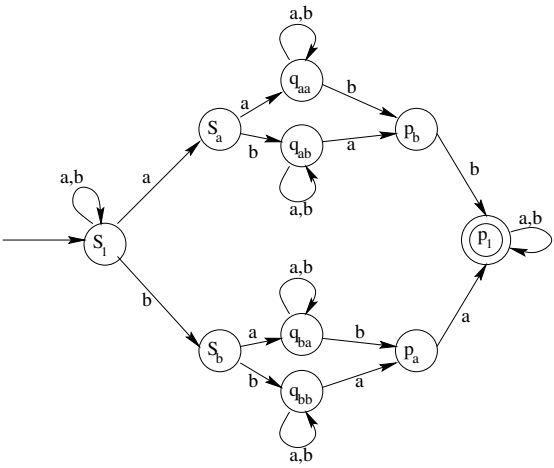


Fig. 2. An example of an NFA accepting the language $hp(\theta, 2)$

Proposition 6. *The number of states of a minimal NFA accepting the language* $hp(\theta, k)$, $k \geq 1$, *over an alphabet* X *of the cardinality* $\ell > 1$, *is between* ℓ^k *and* $3\ell^k$, *its size is at most* $3(\ell^k + \ell^{k+1})$.

Proof. Let $M_k = (S, X, s_1, F, P)$ be an NFA accepting $hp(\theta, k)$.

(i) The reader can easily verify that the set $\mathcal{F} = \{(w, \theta(w)) | w \in X^k\}$ is a fooling set for $hp(\theta, k)$. Therefore $|S| \geq \ell^k$.
(ii) Let

$$S = \{s_w, p_w \mid w \in X^{\leq k-1}\} \cup \{q_w \mid w \in X^k\}.$$

Let further $F = \{p_1\}$. The set of productions P is defined as follows:

$$\begin{array}{ll}
s_v a \to s_w \text{ if and only if } va = w, & \text{for each } v \in X^{\leq k-2},\ a \in X;\\
s_v a \to q_w \text{ if and only if } va = w, & \text{for each } v \in X^{k-1},\ a \in X;\\
q_w a \to q_w & \text{for all } w \in X^k,\ a \in X;\\
q_w a \to p_v \text{ if and only if } \theta(av) = w, & \text{for each } v \in X^{k-1},\ a \in X;\\
p_w a \to p_v \text{ if and only if } av = w, & \text{for each } v \in X^{\leq k-2},\ a \in X.
\end{array}$$

Finally, let $s_1 a \to s_1$ and $p_1 a \to p_1$ for all $a \in X$. An example of the automaton M_k for the case $X = \{a, b\}$, $k = 2$ and θ being an antimorphism, $\theta(a) = b$, $\theta(b) = a$, is at Fig. 2. The reader can verify that $L(M_k) = hp(\theta, k)$, and that $|S| \leq 3\ell^k$, $|P| \leq 3\ell^{k+1}$, therefore $|M_k| \leq 3(\ell^k + \ell^{k+1})$. □

Note that for $\ell = 1$ we have $hp(\theta, k) = X^{2k}X^*$, therefore the size of the minimal automaton accepting $hp(\theta, k)$ is $|M_k| = 4k + 2$.

Proposition 7. *Assume that there are distinct letters $a, b \in X$ such that $a = \theta(b)$. Then the number of states of a minimal NFA accepting $hpf(\theta, k)$, $k \geq 1$, over an alphabet X with the cardinality ℓ, is at least $2^{(\ell-2)^k/2}$.*

Proof. We take into the account only the cases $\ell \geq 3$, the case $\ell = 2$ is trivial. Denote $X_1 = X \setminus \{a, b\}$. We can factorize the set $X_1^k = C_1 \cup C_2 \cup C_3$, where C_1, C_2, C_3 are mutually disjoint sets such that $\theta(C_1) = C_2$ and $\theta(x) = x$ for all $x \in C_3$. Obviously $|C_1| = |C_2|$.

Denote $m = |C_1 \cup C_3|$, then $m \geq (\ell - 2)^k/2$. Consider the set of pairs of strings

$$\mathcal{F} = \left\{ \left(\prod_{w \in D} aw, \prod_{w \in (C_2 \cup C_3) \setminus \theta(D)} aw \right) \;\middle|\; D \subseteq (C_1 \cup C_3) \right\}. \tag{1}$$

We show that $\mathcal{F}$ is a fooling set for $hpf(\theta, k)$.

(i) Consider an arbitrary pair $(x, y) \in \mathcal{F}$. Let $z \in X^k$ be a substring of xy. If z contains a, then $\theta(z)$ cannot be in xy as $\theta(a) = b$ and b is not in xy. If z does not contain a, then $z \in X_1^k$ and z is a subword of either x or y. Assume that z is a part of x. Then, by definition of C_1 and C_3, there is no occurrence of $\theta(z)$ in x which would not overlap z. Also, $\theta(z)$ is not a subword of y as $z \in D$ and hence $\theta(z) \notin (C_2 \cup C_3) \setminus \theta(D)$. If z is a subword of y, the situation is analogous. Therefore, $xy \in hpf(\theta, k)$.

(ii) Let (x, y), (x', y') be two distinct elements of $\mathcal{F}$, associated with the sets $D, D' \subseteq (C_1 \cup C_3)$ in the sense of (1). Let us assume without loss of generality that there is a $z \in D \setminus D'$. Then $\theta(z) \in (C_2 \cup C_3)$ and $\theta(z) \notin \theta(D')$, hence $\theta(z)$ is a subword of y'. Simultaneously z is a subword of x, therefore $xy' \notin hpf(\theta, k)$.

We can conclude that $|\mathcal{F}| = 2^m \geq 2^{(\ell-2)^k/2}$, and hence the statement follows by Lemma 6. □

Corollary 2. *Let X be an alphabet such that $|X| = \ell$, $\ell \geq 2$. Let there be distinct letters $a, b \in X$ such that $a = \theta(b)$. Then the number of states of a minimal DFA over the alphabet X, accepting either $hp(\theta, k)$ or $hpf(\theta, k)$, $k \geq 1$, is between $2^{(\ell-2)^k/2}$ and $2^{3\ell^k}$.*

Proof. Observe that the numbers of states of minimal DFA's accepting $hp(\theta, k)$ and $hpf(\theta, k)$ are the same since these languages are mutual complements. Then the lower bound follows by Proposition 7. The upper bound follows by Proposition 6 and by the subset construction of a DFA equivalent to the NFA M_k mentioned there. □

Corollary 3. *Consider the DNA alphabet $\Delta = \{A, C, T, G\}$ and the Watson-Crick involution τ.*

(i) The size of a minimal NFA accepting $hp(\tau, k)$ is at most $15 \cdot 4^k$. The number of its states is between 4^k and $3 \cdot 4^k$.
(ii) The number of states of either a minimal DFA or an NFA accepting $hpf(\tau, k)$ is between $2^{2^{k-1}}$ and $2^{3 \cdot 2^{2k}}$.

Note: after careful inspection of the automaton in the proof of Proposition 6, one can derive that the actual size is at most $\frac{25}{3} \cdot 4^k + \frac{14}{3}$ and the number of states do not exceed $\frac{5}{3} \cdot 4^k - \frac{2}{3}$.

The above results indicate that the size of a minimal NFA for $hp(\tau, k)$ grows exponentially with k. However, one should recall that k is the *minimal* length of bond allowing for a stable hairpin. Therefore k is rather low in practical applications and the construction of the mentioned automaton can remain computationally tractable.

Acknowledgements

Research was partially supported by the Canada Research Chair Grant to L.K., NSERC Discovery Grants R2824A01 to L.K. and R220259 to S.K., and by the Grant Agency of Czech Republic, Grant 201/02/P079 to P.S. We are indebted to Elena Losseva for drawing the figures and for valuable comments to the paper.

References

1. Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, E. Shapiro, An autonomous molecular computer for logical control of gene expression. *Nature* 429 (2004), 423–429.
2. J.C. Birget, Intersection and union of regular languages and state complexity. *Information Processing Letters* 43 (1992), 185–190.
3. G.S. Brodal, R.B. Lyngsø, C.N.S. Pedersen, J. Stoye, Finding maximal pairs with bounded gap. In *Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching (CPM)*, M. Crochemore and M. Paterson, Eds., LNCS 1645 (1999), 134–149.

4. C.R. Calladine, H.R. Drew, *Understanding DNA: The Molecule and How It Works.* 2nd edition, Academic Press, San Diego, 1999.
5. C. Choffrut, J. Karhumäki, Combinatorics of words. In [18], 329–438.
6. M. Daley, L. Kari, Some properties of ciliate bio-operations. In *Procs. of DLT 2002*, M. Ito, M. Toyama, Eds., LNCS 2450 (2003), 116–127.
7. F.M. Dekking, On repetitions of blocks in binary sequences. *J. Combin. Theory Ser. A* 20 (1976), 292–299.
8. A. Ehrenfeucht, T. Harju, I. Petre, D. Prescott, G. Rozenberg, *Computation in Living Cells: Gene Assembly in Ciliates.* Springer-Verlag, Berlin, 2004.
9. T. Harju, J. Karhumäki, Morphisms. In [18], 439–510.
10. J. Jirásek, G. Jirásková, A. Szabari, State complexity of concatenation and complementation of regular languages. In *CIAA 2004, Ninth International Conference on Implementation and Application of Automata*, M. Domaratzki, A. Okhotin, K. Salomaa, S. Yu, Eds., Queen's University, Kingston, 2004, 132–142.
11. N. Jonoska, D. Kephart, K. Mahalingam, Generating DNA code words. *Congressus Numerantium* 156 (2002), 99–110.
12. N. Jonoska, K. Mahalingam, Languages of DNA based code words. In *DNA Computing, 9th International Workshop on DNA Based Computers*, J. Chen and J.H. Reif, Eds., LNCS 2943 (2004), 61–73.
13. M. Lothaire, *Algebraic Combinatorics on Words.* Cambridge University Press, 2002.
14. A. de Luca, On the combinatorics of finite words. *Theoretical Computer Science* 218 (1999), 13–39.
15. L. Kari, S. Konstantinidis, P. Sosík, G. Thierrin, *On Hairpin-Free Words and Languages.* TR 639, Dept. of Computer Science, University of Western Ontario, 2005, `http://www.csd.uwo.ca/ lila/involuti.ps`.
16. G. Mauri, C. Ferretti, Word Design for Molecular Computing: A Survey. In *DNA Computing, 9th International Workshop on DNA Based Computers*, J. Chen and J.H. Reif, Eds., LNCS 2943 (2004), 37–46.
17. J. A. Rose, R. J. Deaton, M. Hagiya, A. Suyama, PNA-mediated Whiplash PCR. In *DNA Computing, 7th International Workshop on DNA Based Computers*, N. Jonoska and N. C. Seeman, Eds., LNCS 2340 (2002), 104–116.
18. G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, vol. 1, Springer Verlag, Berlin, 1997.
19. N. Takahashi, A. Kameda, M. Yamamoto, A. Ohuchi, Aqueous computing with DNA hairpin-based RAM. In *DNA 10, Tenth International Meeting on DNA Computing*, G. Mauri, C. Ferretti, Eds., University of Milano-Bicocca, 2004, 50–59.
20. G. Thierrin, Convex languages. In *Proc. IRIA Symp. North Holland* 1972, 481–492.

Adding Monotonic Counters to Automata and Transition Graphs

Wong Karianto

Lehrstuhl für Informatik VII, RWTH Aachen, Germany
karianto@i7.informatik.rwth-aachen.de

Abstract. We analyze models of infinite-state automata extended by monotonic counting mechanisms, starting from the (finite-state) Parikh automata studied by Klaedtke and Rueß. We show that, for linear-bounded automata, this extension does not increase the language recognition power. In the framework of infinite transition systems developed by Caucal and others, we show that adding monotonic counters to synchronized rational graphs still results in synchronized rational graphs, in contrast to the case of pushdown graphs or prefix-recognizable graphs. For prefix-recognizable graphs, however, we show that the extension by monotonic counters retains the decidability of the reachability problem.

1 Introduction

The idea of counting devices is a classical one in automata theory. Counters, which are a special case of pushdown stacks – namely such with only one stack symbol – represent the basic model of such devices. Since 2-counter machines are as powerful as Turing machines [13], regarding algorithmic applications, restrictions on the counters under consideration are necessary in order to keep decidability results. Examples of such restrictions are the so-called blind counters [6], which can be found, for example, in Petri nets. Another kind of restrictions lies in the so-called reversal-bounded counters [7]; for these counters, the number of alternations between increments and decrements in any computation is bounded. A special case of reversal-bounded counters is the class of monotonic counters, which is the subject of the present paper.

Current applications of these models can be found in the field of algorithmic verification and also in database theory; counters allow to capture aspects of infinite-state system modeling. Examples of such infinite-state systems are discrete-timed automata [4], reversal-bounded counter machines [8], and some classes of semi-linear systems [1]. In addition, counters provide the possibility to capture some fragments of arithmetic in a computation. For instance, automata on unranked trees (in particular automata on XML documents) can be equipped with arithmetical conditions referring to the sequences of sibling nodes (of unbounded length). Such tree automata have been suggested by Dal Zilio and Lugiez [3] and Seidl et al. [15, 16].

Another example of the application of arithmetic to automata theory, which also serves as our starting point in this paper, are found in the model of Parikh

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 308–319, 2005.
© Springer-Verlag Berlin Heidelberg 2005

automata of Klaedtke and Rueß [10, 11]. In this context, a finite automaton is equipped with some monotonic counters, and at the end of a computation the values of these counters are checked against a constraint of Presburger arithmetic, which is actually given as a semi-linear set.

The purpose of this paper is to consider automata with two infinitary components: an infinite transition graph and a counting mechanism. More precisely, we extend the model of (finite-state) Parikh automata mentioned above using more general automaton models instead of finite automata (for example, pushdown automata and linear-bounded automata[1]). We show results on the expressiveness of the resulting automaton models and on decision problems, in particular with respect to the reachability problem.

Regarding the first aspect, recall that adding counting mechanism to pushdown automata in the form of considering 'Parikh pushdown automata,' as observed in [10], enhances expressive power with respect to language recognition. As first result, we show that for linear-bounded automata this extension does not alter the language recognition power.

Our second result concerning 'expressiveness' is concerned with the classification of infinite transition graphs suggested by Caucal and others (see the survey [18]). In such a graph, the infinite set of vertices is captured by a regular language over an auxiliary alphabet Γ whereas the transition relations are represented by automaton-definable relations over Γ^* (that is, subsets of $\Gamma^* \times \Gamma^*$). Some important classes of such graphs are the class of pushdown graphs, of prefix-recognizable graphs, of synchronized rational graphs, and of rational graphs. We introduce monotonic counters in this context, by attaching vectors of natural numbers to vertices, and observe that, for prefix-recognizable graphs, the resulting graphs, in general, are not prefix-recognizable anymore. As second result, we show that adding monotonic counters to a synchronized rational graph yields again a synchronized rational graph.

Regarding decidability, we observe that there is a prefix-recognizable graph (even a finite graph) such that the extension of this graph by monotonic counters yields a graph with undecidable monadic second-order theory. Nevertheless, we are able to show that the reachability problem for the extension of any prefix-recognizable graph by monotonic counters remains decidable, which might be of interest in the verification of infinite-state systems.

The outline of this paper is as follows. We fix our notations in Sect. 2 and recall the definition of Parikh automata as introduced in [10]. In Sect. 3 we extend the idea of Parikh automata to the automaton classes in the Chomsky hierarchy and show our first result concerning the language recognition power of the resulting automata. In Sect. 4 we introduce the notion of the monotonic-counter extension of transition graphs, starting from the idea of Parikh automata, and show our second result regarding this extension. Section 5 is addressed to the decidability of the reachability problem mentioned above. We conclude with Sect. 6 by giving some final remarks and further perspectives, in particular regarding the more general case of reversal-bounded counters.

[1] Linear-bounded Turing machines

Acknowledgements. I would like to thank Wolfgang Thomas, for supervising my work on this paper, and Philipp Rohde, for proofreading the first draft of this paper. I would like to thank the anonymous referees for some useful suggestions.

2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$ and the set of vectors of natural numbers of dimension $n \geq 1$ by $\mathbb{N}^n$. For a vector $\bar{x} \in \mathbb{N}^n$, we denote its i-th component by $(\bar{x})_i$. For $A \subseteq \mathbb{N}^n$, we define the set $(A)_i := \{(\bar{x})_i \mid \bar{x} \in A\}$. The i-th unit vector in $\mathbb{N}^n$, denoted by $\bar{e}_i$, is the one with the i-th component equal to 1 and all the other components equal to 0. The zero vector of dimension n is denoted by $\bar{0}$.

Let $\bar{x}$ and $\bar{y}$ be vectors of dimension $n \geq 1$, and let k be a natural number. We define the vector addition $\bar{x} + \bar{y}$ and the scalar multiplication $k\bar{x}$ componentwise. For $A, B \subseteq \mathbb{N}^n$, we define $A + B := \{\bar{x} + \bar{y} \mid \bar{x} \in A \text{ and } \bar{y} \in B\}$.

Recall that a set $A \subseteq \mathbb{N}^n$, $n \geq 1$, is said to be *linear* if there are vectors $\bar{x}_0 \in \mathbb{N}^n$ (the *constant vector*) and $\bar{x}_1, \ldots, \bar{x}_m \in \mathbb{N}^n$ (the *periods*), for some $m \geq 0$, such that

$$A = \{\bar{x}_0 + k_1\bar{x}_1 + \ldots + k_m\bar{x}_m \mid k_1, \ldots, k_m \in \mathbb{N}\} \ .$$

The set $A \subseteq \mathbb{N}^n$ is said to be *semi-linear* if it is a finite union of linear sets.

Let $\Sigma = \{a_1, \ldots, a_n\}$, $n \geq 1$, be an alphabet. The *Parikh mapping* $\Phi\colon \Sigma^* \to \mathbb{N}^n$ is defined by $\Phi(w) := (|w|_{a_1}, \ldots, |w|_{a_n})$, for each $w \in \Sigma^*$, where $|w|_a$ denotes the number of occurrences of $a \in \Sigma$ in w. Parikh's theorem [14] asserts that the *Parikh image* (that is, the image under the Parikh mapping) of any context-free language is semi-linear linear and effectively constructible (or *effectively semi-linear*, for short).

The idea underlying a Parikh automaton is the following. We assign a vector of natural numbers, say, of (fixed) dimension $n \geq 1$ to each transition of a finite automaton. Then, at the end of a computation (run) the sum of the vectors associated with the transitions occurring in the computation is checked against a constraint given as a semi-linear set. In this view, the underlying finite automaton is equipped with n-many monotonic counters, which may be incremented each time an input symbol is read, and with the possibility to check the final values of these counters against a semi-linear constraint.

Formally, a *Parikh automaton* of dimension $n \geq 1$ over Σ is a system $(\mathfrak{A}, C)$ where $\mathfrak{A}$ is a finite automaton over an extended alphabet of the form $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$ (the *auxiliary set*), and $C \subseteq \mathbb{N}^n$ (the *constraint set*) is a semi-linear set. We define the *Σ-projection* $\Psi\colon (\Sigma \times D)^* \to \Sigma^*$, mapping $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$ to $u_1 \cdots u_m \in \Sigma^*$, and the *extended Parikh mapping* $\widetilde{\Phi}\colon (\Sigma \times D)^* \to \mathbb{N}^n$, mapping $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$ to $\bar{d}_1 + \cdots + \bar{d}_m \in \mathbb{N}^n$. Now, the acceptance of a word $u_1 \cdots u_m \in \Sigma^*$ by the Parikh automaton requires two conditions. First, there must be some vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ such that the word $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$ is accepted by $\mathfrak{A}$. Second, the sum of these vectors must belong to C. To sum up, we define the language recognized by $(\mathfrak{A}, C)$ as

$$L(\mathfrak{A}, C) := \{\Psi(w) \mid w \in L(\mathfrak{A}) \text{ and } \widetilde{\Phi}(w) \in C\} \ .$$

The following proposition is an extension of Parikh's theorem.

Proposition 1 (Klaedtke and Rueß [10]). *Let Σ be an alphabet and let D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. If the Parikh image of a language $L \subseteq (\Sigma \times D)^*$ is (effectively) semi-linear, then so is its extended Parikh image.*

3 Parikh Automata and the Chomsky Hierarchy

The definition of a Parikh automaton in the preceding section allows us to use any automaton model for the underlying automaton instead of a finite automaton. Using the well-known automaton models of the Chomsky hierarchy, we obtain *Parikh pushdown automata* (*Parikh-PDA*), *Parikh linear-bounded automata* (*Parikh-LBA*), and *Parikh Turing machines* (*Parikh-TM*). To avoid confusion, we also refer to the Parikh automata of Klaedtke and Rueß (with finite automata as the underlying automata) as *Parikh finite automata* (*Parikh-FA*). Obviously, a finite automaton is equivalent to a Parikh-FA; the corresponding Parikh-FA just does not make use of its counters. Similar fact holds for PDA's, LBA's, and TM's as well.

It is not difficult to construct a Parikh-FA that recognizes the language $\{a^k b^k c^k \mid k \geq 0\}$, which is not context-free. Consequently, Parikh-FA's and Parikh-PDA's are more powerful than finite automata and pushdown automata, respectively. In contrast, we show that any Parikh-LBA can be simulated by an LBA. Furthermore, we show that the class of Parikh-PDA-recognizable languages is properly contained in the class of context-sensitive languages.

Theorem 2. *Each Parikh-LBA is equivalent to an LBA.*

Proof. Let $(\mathfrak{M}, C)$ be a Parikh-LBA of dimension $n \geq 1$ over Σ with the auxiliary set $D \subseteq \mathbb{N}^n$. We construct a TM $\mathfrak{M}'$ over Σ that recognizes $L(\mathfrak{M}, C)$ and show that the amount of space used by $\mathfrak{M}'$ in any computation is linear in the length of its input, which implies that $\mathfrak{M}'$ is indeed an LBA.

The TM $\mathfrak{M}'$ works as follows. Given an input word $u := u_1 \cdots u_m \in \Sigma$, $m \geq 0$, we first guess the vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ nondeterministically and maintain the sum $\bar{x}$ of these vectors somewhere in the working tape (with a unary representation described below). Then, we simulate the computation of $\mathfrak{M}$ on $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$. If $\mathfrak{M}$ accepts, then it remains to check whether the vector $\bar{x}$ belongs to C.

Without loss of generality, we assume that C is linear since, otherwise, we can nondeterministically choose one of the finitely many linear sets that constitute C to work with. Let $\bar{x}_0$ be the constant vector and $\bar{x}_1, \ldots, \bar{x}_r$ be the periods of C. First, we subtract $\bar{x}_0$ from $\bar{x}$. Then, one by one, we nondeterministically choose a period $\bar{x}_j$ and subtract it from $\bar{x}$. If $\bar{x}$ belongs to C, then it will eventually become the zero vector, whereupon we go to an accepting state.

We turn to analyzing the amount of space used by $\mathfrak{M}'$ to accept an input word $u \in L(\mathfrak{M}, C)$. Simulating $\mathfrak{M}$ requires only linear space since $\mathfrak{M}$ is an LBA.

Thus, the critical point in the construction above lies in how much space is needed to maintain the vector $\bar{x}$ and to check whether $\bar{x}$ belongs to C. For the former task, we will use a unary representation of the vector $\bar{x}$. Then, the latter task does not need extra space since we only subtract some vectors from $\bar{x}$ and do not add some to $\bar{x}$. For the unary representation of $\bar{x}$, we use the word

$$\underbrace{|_1 \cdots |_1}_{(\bar{x})_1\text{-times}} \quad \$ \quad \cdots \quad \$ \quad \underbrace{|_n \cdots |_n}_{(\bar{x})_n\text{-times}} \quad ,$$

where $|_1, \ldots, |_n$, and \$ are new symbols. Let t be the greatest natural number occurring in D, that is, $t := \max\{(\bar{d})_i \mid \bar{d} \in D \text{ and } 1 \leq i \leq n\}$. Since each input symbol contributes at most t to each component of $\bar{x}$, it is not difficult to see that the length of the unary representation of $\bar{x}$ can be bounded by $n \cdot t \cdot |u| + n - 1$, which is linear in $|u|$ since n and t are fixed (the term $n-1$ represents the number of \$'s). Hence, we conclude that $\mathfrak{M}'$ is indeed an LBA. □

Note that the idea of the proof of Theorem 2 above applies to the case of Parikh-TM as well.

Corollary 3. *Each Parikh-TM is equivalent to a TM.*

Theorem 2 and Corollary 3 imply that we do not need to introduce new language classes in order to capture Parikh-LBA's and Parikh-TM's. In the following, we classify the Parikh-FA-recognizable and Parikh-PDA-recognizable languages in the Chomsky hierarchy. The results are summarized in Fig. 1, where each solid edge pointing to the right indicates a strict inclusion. For simplicity, we denote the language classes by the corresponding automaton classes.

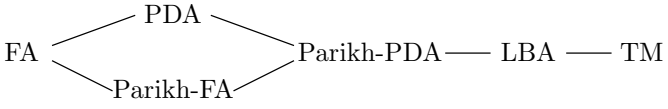


Fig. 1. Hierarchy of language classes

Clearly, any Parikh-FA can be simulated by a Parikh-PDA, which in turn can be simulated by a Parikh-LBA (and thus, by Theorem 2, also by an LBA). These facts justify the inclusion relations shown in Fig. 1. Moreover, these inclusions are strict. As noted before, the non-context-free language $\{a^k b^k c^k \mid k \geq 0\}$ is Parikh-FA-recognizable. On the other hand, one can show that the context-free language $\{ww^{\mathrm{R}} \mid w \in \{a, b\}^*\}$ is not Parikh-FA-recognizable[2]. The strictness of the inclusion between Parikh-PDA's and LBA's follows from Proposition 4

[2] Actually, the proof of this fact is merely a slight modification of the proof that the language $\{ww \mid w \in \{a, b\}^*\}$ is not Parikh-FA-recognizable, as presented in [10]. For further details, the reader is referred to [9, pages 31–33]

below, which states that the Parikh image of any Parikh-PDA-recognizable language is semi-linear, and from the fact that there is a context-sensitive language whose Parikh image is not semi-linear; for instance, it is straightforward to show that the language $\{a^k b^{k^2} \mid k \geq 0\}$ is context sensitive, but its Parikh image is not semi-linear (see [9, pages 56 and 66–67]).

Proposition 4. *The Parikh image of any Parikh-PDA-recognizable language is effectively semi-linear.*

Proof (sketch). We use the extended Parikh mapping to 'simulate' the Parikh mapping by means of some additional dimensions.

Given a Parikh-PDA $(\mathfrak{A}, C)$ of dimension n over $\Sigma := \{a_1, \ldots, a_m\}$, we extend it to a new Parikh-PDA $(\mathfrak{B}, C')$ of dimension $n + m$ that recognizes the same language as $(\mathfrak{A}, C)$ and counts the number of occurrences of the input symbols by using the additional dimensions. Roughly speaking, the idea is to associate each transition in which a_i is involved with the unit vector $\bar{e}_i$. Then, the vector accumulated at the end of a computation contains the number of occurrences of the input symbols in the last m components. The set of such m-dimensional vectors can be extracted from the extended Parikh image of $L(\mathfrak{B})$ by using the projection functions. The semi-linearity of this set then follows from the semi-linearity of $L(\mathfrak{B})$ (Proposition 1) and the closure of semi-linear sets under the Boolean operations and the projection functions [5]. □

Corollary 5. *The class of Parikh-PDA-recognizable languages is properly contained in the class of context-sensitive languages.*

4 Monotonic-Counter Extensions of Transition Graphs

Let Σ be an alphabet. A *Σ-labeled (transition) graph* is a structure $G := (V, (E_a)_{a \in \Sigma})$, where V is the set of *vertices*, and $E_a \subseteq V \times V$, for each $a \in \Sigma$, is the set of *a-labeled edges* of G. The set of all edges of G is denoted by $E := \bigcup_{a \in \Sigma} E_a$.

Although the graph G under consideration might be infinite, we require that it has a finite representation; the set V of vertices is given by a regular language over an auxiliary alphabet, say Γ, and the set E_a of a-labeled edges, for each $a \in \Sigma$, is given by a binary relation over Γ^*, which is, more or less, definable by an automaton. Some well-known classes of such graphs, with respect to how the edge relations are defined, are *pushdown graphs*, *prefix-recognizable graphs*, *synchronized rational graphs*, and *rational graphs*, where the edge relations are defined by ε-free pushdown transitions, prefix rewriting rules, synchronized rational relations, and rational relations, respectively. Moreover, it is known that these classes constitute a strict increasing chain of inclusions. For an introduction to these graph classes, the reader is referred to [18].

Note that in the sequel we do not distinguish Σ-labeled graphs up to isomorphism; we consider two isomorphic Σ-labeled graphs to be the same. For convenience, we denote the class of finite graphs, pushdown graphs, prefix-recognizable

graphs, synchronized rational graphs, and rational graphs by $\mathcal{G}_{\mathrm{Fin}}$, $\mathcal{G}_{\mathrm{PD}}$, $\mathcal{G}_{\mathrm{PR}}$, $\mathcal{G}_{\mathrm{SR}}$, and $\mathcal{G}_{\mathrm{Rat}}$, respectively.

We apply the idea of Parikh automata as follows. First, we use an extended alphabet of the form $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$, $n \geq 1$, for the edge labeling, thereby obtaining a $(\Sigma \times D)$-labeled graph. If we now take into consideration the vectors that are *accumulated along a path* and code them as monotonic counters (represented by vectors of natural numbers) in the vertices, the vector component of any edge label can be reconstructed from the vectors of the vertices that are incident on this edge. Thus, we may omit the vector component of the edge labels. In this way, we get back to a Σ-labeled graph.

Definition 6. *Let Σ be an alphabet, and let D (called the* auxiliary set*) be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $G := (V, (E_{(a,\bar{d})})_{(a,\bar{d}) \in \Sigma \times D})$ be a $(\Sigma \times D)$-labeled graph. The Σ-*labeled monotonic-counter extension *(of dimension n) of G (with respect to D) is the Σ-labeled graph $\widetilde{G} := (\widetilde{V}, (\widetilde{E}_a)_{a \in \Sigma})$ with $\widetilde{V} := V \times \mathbb{N}^n$ and*

$$\widetilde{E}_a := \{((\alpha, \bar{x}), (\beta, \bar{y})) \in \widetilde{V} \times \widetilde{V} \mid \exists \bar{d} \in D : (\alpha, \beta) \in E_{(a,\bar{d})} \text{ and } \bar{y} = \bar{x} + \bar{d}\} \ ,$$

for each $a \in \Sigma$. The set of all edges of $\widetilde{G}$ is denoted by $\widetilde{E} := \bigcup_{a \in \Sigma} \widetilde{E}_a$.

For a class $\mathcal{G}(\Sigma)$ of Σ-labeled graphs, the class $\mathcal{G}_{\mathrm{MC}}(\Sigma)$ precisely contains the Σ-labeled graphs in $\mathcal{G}(\Sigma)$ and their (Σ-labeled) monotonic-counter extensions. We will omit Σ whenever it is clear from the context. Occasionally, we refer to a graph that is obtained by a monotonic-counter extension simply as a monotonic-counter graph*.*

Example 7. The infinite two-dimensional grid (see [18, page 134]) can be captured as the following monotonic-counter graph. Let $\Sigma := \{a, b\}$ and $D := \{(1, 0), (0, 1)\}$. We define the $(\Sigma \times D)$-labeled finite graph G_{grid} depicted in Fig. 2 (left). Then, the monotonic-counter extension of G_{grid} is the graph $\widetilde{G}_{\mathrm{grid}} = (\widetilde{V}, \widetilde{E}_a, \widetilde{E}_b)$ with $\widetilde{V} := \{(\bullet, (i, j)) \mid i, j \in \mathbb{N}\}$, $\widetilde{E}_a := \{((\bullet, (i, j)), (\bullet, (i + 1, j))) \mid i, j \in \mathbb{N}\}$, and $\widetilde{E}_b := \{((\bullet, (i, j)), (\bullet, (i, j + 1))) \mid i, j \in \mathbb{N}\}$. The graph $\widetilde{G}_{\mathrm{grid}}$ is illustrated in Fig. 2 (right), where, for better readability, the symbol $\bullet$ has been omitted.

Starting from the class $\mathcal{G}_{\mathrm{Fin}}$ of finite graphs, we obtain the class $\mathcal{G}_{\mathrm{FMC}}$. Analogously, from the graph classes mentioned above we obtain the classes $\mathcal{G}_{\mathrm{PDMC}}$, $\mathcal{G}_{\mathrm{PRMC}}$, $\mathcal{G}_{\mathrm{SRMC}}$, and $\mathcal{G}_{\mathrm{RMC}}$. Moreover, these classes constitute an increasing chain of inclusions as the underlying graph classes also do.

By definition, we have $\mathcal{G} \subseteq \mathcal{G}_{\mathrm{MC}}$, for any class $\mathcal{G}$ of Σ-labeled graphs. For prefix-recognizable graphs, the converse does not hold; the infinite two-dimensional grid of Example 7 is known to have an undecidable monadic second-order theory and thus is not prefix-recognizable [18, Theorem 8 and Theorem 10]. However, the converse holds for synchronized rational graphs and for rational graphs. Towards showing this result, we briefly recall the definitions of synchronized rational and rational graphs. For further references, the reader is referred to [18, pages 133–135].

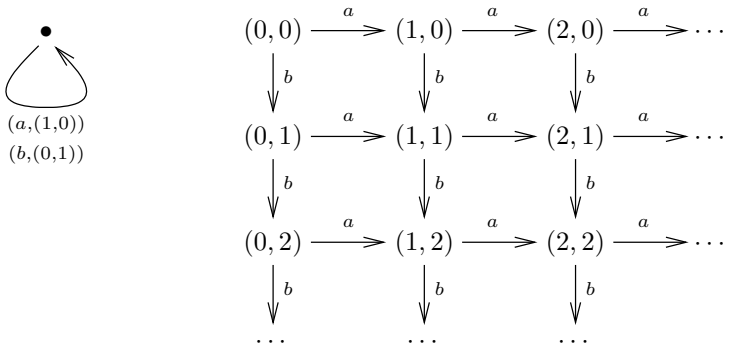


Fig. 2. The graph G_{grid} (left) and $\widetilde{G}_{\text{grid}}$ (right) of Example 7

Let Γ be an alphabet, and let $\diamond$ be a new symbol. For convenience, we write $\Gamma_\diamond$ for $\Gamma \cup \{\diamond\}$. For any words $\alpha = X_1 \cdots X_m$ and $\beta = Y_1 \cdots Y_n$ over Γ, we define $\alpha \hat{} \beta := \left[\begin{smallmatrix} X'_1 \\ Y'_1 \end{smallmatrix}\right] \cdots \left[\begin{smallmatrix} X'_k \\ Y'_k \end{smallmatrix}\right]$, a word of length $k := \max(m, n)$ over $\Gamma_\diamond \times \Gamma_\diamond$, where $X'_i := X_i$, for $i \leq m$, and $X'_i := \diamond$, otherwise; similarly, $Y'_j := Y_j$, for $j \leq n$, and $Y'_j := \diamond$, otherwise. We assign to a relation $R \subseteq \Gamma^* \times \Gamma^*$ the language

$$L_R := \{\alpha \hat{} \beta \mid (\alpha, \beta) \in R\} \subseteq (\Gamma_\diamond \times \Gamma_\diamond)^* \ .$$

The relation R is called *synchronized rational* if L_R is regular. Intuitively, a finite automaton recognizing L_R can be seen as a finite automaton with two one-way input tapes, each with its own input head, which may only be moved simultaneously. The special symbol $\diamond$ is needed to deal with the case where the two words under consideration are of different length.

The automaton characterization of synchronized rational relations carries over to rational relations; a relation $R \subseteq \Gamma^* \times \Gamma^*$ is called *rational* if L_R is recognized by a finite automaton with two one-way input tapes, each with its own input head, which may be moved independently from each other.

Let Σ be an alphabet. A Σ*-labeled* (*synchronized*) *rational graph* is a graph $G = (V, (E_a)_{a \in \Sigma})$ where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and E_a is (synchronized) rational, for each $a \in \Sigma$.

Theorem 8. *The monotonic-counter extension of a synchronized rational graph is a synchronized rational graph.*

Proof. Let Σ be an alphabet and D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $\widetilde{G} = (\widetilde{V}, (\widetilde{E}_a)_{a \in \Sigma})$ be the monotonic-counter extension of the synchronized rational graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d}) \in \Sigma \times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and $E_{(a,\bar{d})}$ is synchronized rational, for each $(a, \bar{d}) \in \Sigma \times D$; that is, each $L_{E_{(a,\bar{d})}}$ is recognizable by a finite automaton with two one-way input tapes and simultaneously moving input heads, say by $\mathfrak{A}_{(a,\bar{d})}$. In order to show that $\widetilde{G}$ is synchronized rational, we need to find a synchronized rational graph that is isomorphic to $\widetilde{G}$.

First of all, we need to code the vertices of $\widetilde{G}$, each of which consists of a word over Γ and a vector of dimension n. Let $\Pi := \Gamma \cup \{|_1, \ldots, |_n\}$, where $|_1, \ldots, |_n$ are new symbols, which will be used to code vectors of dimension n as follows. Each vector $(x_1, \ldots, x_n) \in \mathbb{N}^n$ is represented by the word $f(x_1, \ldots, x_n) := |_1^{x_1} \cdots |_n^{x_n}$. Using this coding, we now define a coding function for the vertices of $\widetilde{G}$. Let $h\colon V \times \mathbb{N}^n \to (|_1^* \cdots |_n^*)V$ be defined by $h(\alpha, \bar{x}) := f(\bar{x})\alpha$, for each $\alpha \in V$ and $\bar{x} \in \mathbb{N}^n$. Clearly, both f and h are bijective, and thus, the graph $G' := (V', (E'_a)_{a\in\Sigma})$ with

$$V' := (|_1^* \cdots |_n^*)V \ ,$$
$$E'_a := \{(h(\alpha, \bar{x}), h(\beta, \bar{y})) \in V' \times V' \mid ((\alpha, \bar{x}), (\beta, \bar{y})) \in \widetilde{E}_a\}$$

is isomorphic to $\widetilde{G}$. The task is now to show that G' is synchronized rational.

Clearly, the set V' of vertices of G' is regular since V is regular. It remains to show that the edge relation E'_a is synchronized rational, for each $a \in \Sigma$. For each edge relation E'_a, we define a finite automaton $\mathfrak{A}'_a$ with two one-way input tapes and simultaneously moving input heads which works as follows. Let two vertices $f(\bar{x})\alpha$ and $f(\bar{y})\beta$ of G' be given.

- First, we guess a vector $\bar{d} \in D$.
- Then, we check whether $\bar{x}+\bar{d} = \bar{y}$. Note that, for this purpose, the automaton must be able to perform a kind of 'shifted reading' since the input heads may only be moved simultaneously on both input tapes. To illustrate this, consider the following example. Let $\alpha := ab$, $\beta := b$, $\bar{x} := (1, 2)$, and $\bar{y} := (3, 3)$. Then, we have $h(\alpha, \bar{x})\hat{}\,h(\beta, \bar{y}) = \left[\begin{smallmatrix}|_1\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}|_2\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}|_2\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}a\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}b\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}\diamond\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}\diamond\\a\end{smallmatrix}\right]$. In this example, while we are still working on the first component of the vectors (represented by sequences of $|_1$'s), we must already deal with the second component of the vectors (represented by sequences of $|_2$'s), and similarly, while we are still working with the second component of the vectors, we must already deal with the Γ-components. Since D and Γ are finite (and n is fixed), we need only use a finite memory. In particular, we must remember the *differences* between $\bar{x}$ and $\bar{y}$ (that is, the vector $\bar{d}$) in each component.
- Finally, after verifying that $\bar{x}+\bar{d} = \bar{y}$, we just simulate the automaton $\mathfrak{A}_{(a,\bar{d})}$, checking whether $(\alpha, \beta) \in E_{(a,\bar{d})}$.

It is now straightforward to see that $\mathfrak{A}'_a$ indeed recognizes $L_{E'_a}$. Thus, E'_a is synchronized rational, for each $a \in \Sigma$, and consequently, G' (and thus also $\widetilde{G}$) is indeed a synchronized rational graph. □

The proof idea above can also be applied to the monotonic-counter extensions of rational graphs; the construction of the automata for the edge relations is even easier since the input heads may be moved independently from each other.

Corollary 9. *The monotonic-counter extension of a rational graph is a rational graph.*

Consequently, the classes $\mathcal{G}_{\mathrm{SR}}$ and $\mathcal{G}_{\mathrm{Rat}}$ coincide with the classes $\mathcal{G}_{\mathrm{SRMC}}$ and $\mathcal{G}_{\mathrm{RMC}}$, respectively. The class $\mathcal{G}_{\mathrm{PRMC}}$ is contained in $\mathcal{G}_{\mathrm{SRMC}}$ (and thus also in

$\mathcal{G}_{\mathrm{SR}}$) since $\mathcal{G}_{\mathrm{PR}}$ is contained in $\mathcal{G}_{\mathrm{SR}}$. Moreover, this inclusion is strict since $\mathcal{G}_{\mathrm{SR}}$ contains graphs for which the reachability problem is undecidable [18, Theorem 9] while $\mathcal{G}_{\mathrm{PRMC}}$ does not (see Sect. 5 below). Obviously, the class $\mathcal{G}_{\mathrm{PRMC}}$ in turn subsumes the classes $\mathcal{G}_{\mathrm{PR}}$ and $\mathcal{G}_{\mathrm{PDMC}}$. These inclusions are also strict; on the one hand, the infinite two-dimensional grid of Example 7 belongs to $\mathcal{G}_{\mathrm{PDMC}}$, but not to $\mathcal{G}_{\mathrm{PR}}$; on the other hand, since pushdown graphs are of bounded degree and since the auxiliary sets under consideration are always finite, the class $\mathcal{G}_{\mathrm{PDMC}}$ only contains graphs of bounded degree, which does not hold for $\mathcal{G}_{\mathrm{PR}}$ (see [18, pages 132–133]).

These (and some other) hierarchy results are summarized in Fig. 3, where each solid edge pointing to the right indicates a strict inclusion. For simplicity, we write, for instance, FMC instead of $\mathcal{G}_{\mathrm{FMC}}$. The proofs are straightforward and can be found in [9, pages 113–118].

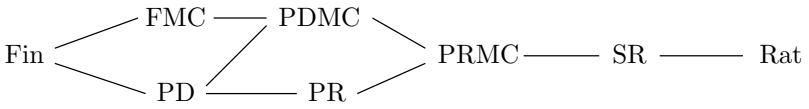


Fig. 3. Hierarchy of graph classes

5 Reachability Problem

A fundamental question in the field of model checking is the reachability problem. For prefix-recognizable graphs, this problem is decidable whereas it is undecidable for synchronized rational graphs. Regarding the former graphs, we show that the monotonic-counter extension preserves the decidability of the reachability problem.

To be precise, we consider the following reachability problem. Let Σ and Γ be alphabets, and let D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let G be a $(\Sigma \times D)$-labeled prefix-recognizable graph with regular set $V \subseteq \Gamma^*$ of vertices. Then, the reachability problem for $\widetilde{G}$, the monotonic-counter extension of G, is the question: "Given two regular sets $U, U' \subseteq V$ of vertices in G, and given two semi-linear sets $C, C' \subseteq \mathbb{N}^n$, are there vertices $(\alpha, \bar{x}) \in U \times C$ and $(\beta, \bar{y}) \in U' \times C'$ in $\widetilde{G}$ such that $(\beta, \bar{y})$ is reachable from $(\alpha, \bar{x})$?"

Towards our result, we will need the following lemma, stating that the traces[3] of prefix-recognizable graphs, with regular sets of initial and final vertices, are context-free. Actually, this lemma is a corollary of the fact that each prefix-recognizable graph is the ε-closure of the configuration graph of a pushdown automaton [17] (a *constructive* proof can be found in [12]).

Lemma 10 (Caucal [2]). *The traces of any Σ-labeled prefix-recognizable graph yield a context-free language over Σ, which is effectively constructible.*

[3] For the definition of the traces of transition graphs, the reader might consult [18]

In order to show the decidability of the reachability problem for the monotonic-counter extensions of prefix-recognizable graphs we reduce this problem to the emptiness problem for semi-linear sets, which is known to be decidable.

Theorem 11. *The reachability problem for the monotonic-counter extension of any prefix-recognizable graph is decidable.*

Proof. Let Σ be an alphabet and D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $\widetilde{G}$ be the monotonic-counter extension of the prefix-recognizable graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ. Now, let $U, U' \subseteq V$ be regular sets of vertices in G, and let $C, C' \subseteq \mathbb{N}^n$ be semi-linear sets. By Lemma 10, the traces of G, with U as the set of initial vertices and U' as the set of final vertices, yield *effectively* a context-free language L over $\Sigma \times D$. Then, by Parikh's theorem and Proposition 1, the extended Parikh image of L (denoted by $\widetilde{\Phi}(L)$) is effectively semi-linear.

It is straightforward to show that the following statements are equivalent:

1. There are some vertices $(\alpha, \bar{x}) \in U \times C$ and $(\beta, \bar{y}) \in U' \times C'$ in $\widetilde{G}$ such that $(\beta, \bar{y})$ is reachable from $(\alpha, \bar{x})$.
2. $(C + \widetilde{\Phi}(L)) \cap C' \neq \emptyset$.

By the effective closure of semi-linear sets under addition (this property is straightforward to verify) and intersection [5], the set $(C + \widetilde{\Phi}(L)) \cap C'$ is effectively semi-linear, and hence, the emptiness problem for this set is decidable. Consequently, it is decidable whether the first statement above holds, which in turn implies the decidability of the reachability problem for $\widetilde{G}$. □

6 Conclusions

We have drawn the boundary where the idea of Parikh automata, which corresponds to the extension of automata by monotonic counters, properly increases the language recognition power of the automaton classes of the Chomsky hierarchy. While this statement is true for finite automata and pushdown automata, it does not hold for linear-bounded automata and Turing machines. Likewise, we showed that adding monotonic counters to synchronized rational graphs does not go beyond the scope of synchronized rational graphs whereas for prefix-recognizable graphs one obtains a new class of transition graphs. Nevertheless, for the latter we showed that the reachability problem remains decidable.

A natural next step towards extending the present results, which is also a subject of our current work, is to consider the case of reversal-bounded counters [7] instead of monotonic counters. For this case, some technical preparations on the definitions are needed, in particular in order to connect the model of Parikh automata with the model of reversal-bounded counter machines carefully. For the former model, for instance, a constraint on the values of the counters (via semi-linear sets) occurs only at the end of computations whereas for the latter model intermediate tests are allowed. A first contribution to this issue has been

done by Klaedtke and Rueß [10], showing that (finite-state) Parikh automata and reversal-bounded counter machines are equivalent. Another direction is to extend the arithmetical conditions under consideration beyond the scope of semi-linear sets.

References

1. A. Bouajjani, P. Habermehl: Constrained properties, semi-linear systems, and Petri nets. In Proc. CONCUR 1996. LNCS 1119. Springer (1996) 481–497
2. D. Caucal: On infinite transition graphs having a decidable monadic theory. Theoretical Computer Science **290** (2003) 79–115
3. S. Dal Zilio, D. Lugiez: XML schema, tree logic and sheaves automata. In Proc. RTA 2003. LNCS 2706. Springer (2003) 246–263
4. Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su: Binary reachability analysis of discrete pushdown timed automata. In Proc. CAV 2000. LNCS 1855. Springer (2000) 69–84
5. S. Ginsburg, E.H. Spanier: Bounded ALGOL-like languages. Transactions of the American Mathematical Society **113** (1964) 333–368
6. S.A. Greibach: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science **7** (1978) 311–324
7. O.H. Ibarra: Reversal-bounded multicounter machines and their decision problems. Journal of the Association for Computing Machinery **25** (1978) 116–133
8. O.H. Ibarra, T. Bultan, J. Su: Reachability analysis for some models of infinite-state transition systems. In Proc. CONCUR 2000. LNCS 1877. Springer (2000) 183–198
9. W. Karianto: Parikh automata with pushdown stack. Diploma thesis, RWTH Aachen, Germany (2004)
10. F. Klaedtke, H. Rueß: Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science, Freiburg University, Germany (2002)
11. F. Klaedtke, H. Rueß: Monadic second-order logics with cardinalities. In Proc. ICALP 2003. LNCS 2719. Springer (2003) 681–696
12. M. Leucker: Prefix-recognizable graphs and monadic logic. In Automata, Logics, and Infinite Games. LNCS 2500. Springer (2002) 263–283
13. M.L. Minsky: Recursive unsolvability of Post's problem of 'tag' and other topics in theory of Turing machines. Annals of Mathematics **74** (1961) 437–455
14. R.J. Parikh: On context-free languages. Journal of the Association for Computing Machinery **13** (1966) 570–581
15. H. Seidl, T. Schwentick, A. Muscholl: Numerical document queries. In Proc. PODS 2003. ACM Press (2003) 155–166
16. H. Seidl, T. Schwentick, A. Muscholl, P. Habermehl: Counting in trees for free. In Proc. ICALP 2004. LNCS 3142. Springer (2004) 1136–1149
17. C. Stirling: Decidability of bisimulation equivalence for pushdown processes. Technical Report EDI-INF-RR-0005, School of Informatics, University of Edinburgh, Scotland (2000)
18. W. Thomas: A short introduction to infinite automata. In Proc. DLT 2001. LNCS 2295. Springer (2002) 130–144

Polynomial Generators of Recursively Enumerable Languages

Juha Kortelainen

University of Oulu, Department of Information Processing Science, Finland
jkortela@tols16.oulu.fi

Abstract. For each language L, let $\hat{\mathcal{F}}_\cap(L)$ be the smallest intersection-closed full AFL generated by the language L. Furthermore, for each natural number $k \geq 2$ let $P_k = \{a^{n^k} | n \in \mathbb{N}\}$. By applying certain classical and recent results on Diophantine equations we show that $\mathcal{L}_{RE} = \hat{\mathcal{F}}_\cap(P_k)$, i.e., the family of all recursively enumerable languages coincides with the smallest intersection-closed full AFL generated by the polynomial language P_k for all $k \geq 2$. This allows us to answer to an open problem of S. Ginsburg and J. Goldstine in [2].

1 Introduction

The creators of formal language theory studied intensively simple generators of classical Chomsky hierarchy language families, especially those of recursively enumerable languages. The role of the language $DUP = \{a^n b^n \mid n \in \mathbb{N}\}$ appeared to be crucially important. Certainly DUP is a one-counter context-free language which, in the basic language theory courses serves as the first example of a nonregular set. Anyhow in [3] it was shown that $\hat{\mathcal{F}}_\cap(DUP)$ coincides with the family $\mathcal{L}_{RE}$ of all recursively enumerable languages. Since then, to prove that $\hat{\mathcal{F}}_\cap(L)$ contains all r.e. languages, it became a common practise to show that $DUP \in \hat{\mathcal{F}}_\cap(L)$. Ginsburg and Goldstine studied one-letter generators of $\mathcal{L}_{RE}$ in [2]; they proved, among other deep things that

- $DUP \in \hat{\mathcal{F}}_\cap(L)$ for each infinite language $L = \{a^{n_i} \mid i \in \mathbb{N}\}$ such that

$$lim\ inf_{i\to\infty} \frac{n_{i+1}}{n_i} \ > \ 1 \ ; \text{ and}$$

- $DUP \notin \hat{\mathcal{F}}_\cap(L)$ for each language $L \subseteq a^*$ such that

$$lim_{n\to\infty} \frac{|\{a^i \mid a^i \in L, 0 \leq i < n\}|}{n} \ = \ 1 \ .$$

Recall that all one-letter context-free languages are regular. This means that a one-letter generator of $\mathcal{L}_{RE}$ cannot be context-free. The first of the items above can be interpreted so that if a one-letter language is sparse enough, it generates all r.e. languages. For instance each exponential language $EXP_k = \{a^{k^n} \mid n \in \mathbb{N}\}$ with integer base $k \geq 2$ satisfies its condition. On the other hand, the latter result

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 320–326, 2005.
© Springer-Verlag Berlin Heidelberg 2005

says that if $L \subseteq a^*$ is appropriately thick, then $\mathcal{L}_{RE}$ is not a subset of $\hat{\mathcal{F}}_\cap(L)$. The complement $a^* \setminus P_k$ of the polynomial language P_k (for each $k \in \{2, 3, ...\}$) is then certainly not a generator of r.e. languages with respect to the full AFL operations and intersection. Ginsburg and Goldstine left open the question whether or not

$$\mathcal{L}_{RE} \;=\; \hat{\mathcal{F}}_\cap(P_k) \tag{1}$$

is true when $k \in \{2, 3, ...\}$. We attack the problem with a simple reduction of language-theoretic problems to (systems of) Diophantine equations.

Andrew Wiles's celebrated proof of Fermat's Last Theorem in 1995 [7] brought on solutions to other difficult Diophantine equations resembling $x^n + y^n = z^n$. H. Darmon H. and L. Merel showed in [1] that the $x^n + y^n = 2\,z^n$ has no solutions for integers n, x, y, z such that $x \neq y$ and $n \geq 3$. A straightforward application of this fact shows that (1) holds true for integers $k \geq 3$. To prove that (1) is valid also for $k = 2$, we make use of parametric solutions to Diophantine equations of the type $a\,x^2 + b\,y^2 = c\,y^2$ presented in [5].

In [6] some more sophisticated generators of $\mathcal{L}_{RE}$ are considered. Turakainen proves that $\hat{\mathcal{C}}_\cap(PROD)$, the smallest intersection closed full trio generated by $PROD = \{a^n b^n c^{n\,m} | n, m \in \mathbb{N}\}$, contains all recursively enumerable one-letter languages. For the sets $REP = \{(a^n b)^m |\ n, m \in \mathbb{N}\}$ and $BREP = \{(a^n b)^n |\ n \in \mathbb{N}\}$ the relations

$$\hat{\mathcal{C}}_\cap(DUP) \subsetneq \hat{\mathcal{C}}_\cap(PROD) \subsetneq \hat{\mathcal{C}}_\cap(BREP) \subseteq \hat{\mathcal{C}}_\cap(REP) \tag{2}$$

are as well verified in [6]. Whether the last inclusion is an equality remained open. We prove that REP is in $\hat{\mathcal{C}}_\cap(BREP)$, thus verifying that $\hat{\mathcal{C}}_\cap(BREP) = \hat{\mathcal{C}}_\cap(REP)$.

2 Basic Definitions

Denote by $\mathbb{N}$ the set of all natural numbers, by $\mathbb{Z}$ the set of all integers and by $\mathbb{Q}$ the set of all rational numbers.

For each finite set S, let $|S|$ be the number of elements in S.

Let X be a finite alphabet and let X^* be the free monoid generated by X. As usual, the elements of X^* are *words*, the subsets of X^* are *languages*. Let $w \in X^*$ be such that $w = x_1 x_2 \cdots x_n$ where n is a nonnegative integer and $x_i \in X$ for $i = 1, 2, \cdots, n$. The number n is the *length* of x, denoted by $|x|$. If $n = 0$, then w is the *empty word* e. For each $y \in X^*$ and $a \in X$, let $|w|_a$ be the number of occurrences of the symbol a in w. A language L is e-free if it does not contain the empty word. The language operation *shuffle*, denoted by $shuf$, between two languages L_1 and L_2 over the alphabet X is defined by

$$shuf(L_1, L_2) = \{u_1 v_1 u_2 v_2 \cdots u_n v_n |\ \ n \in \mathbb{N},\ u_i, v_i \in X^* \text{ for } i = 1, 2, \ldots n,\\ u_1 u_2 \cdots u_n \in L_1,\ v_1 v_2 \cdots v_n \in L_2\}\ .$$

A *family of languages* is any set of languages containing at least one nonempty element. A (*full*) *trio* is a family of languages closed under nonerasing morphism (arbitrary morphism), inverse morphism and intersection with regular

sets. A (full) trio closed under union, concatenation and Kleene+ is a (*full*) *AFL* (acronym for Abstract Family of Languages).

For each language L, let

- $\mathcal{C}_\cap(L)$ be the smallest intersection-closed trio
- $\mathcal{F}_\cap(L)$ be the smallest intersection-closed AFL
- $\hat{\mathcal{C}}_\cap(L)$ be the smallest intersection-closed full trio
- $\hat{\mathcal{F}}_\cap(L)$ be the smallest intersection-closed full AFL

containing (or generated by) the language L.

It is easy to see that each

- trio is closed under union with an e-free regular sets; and
- intersection closed trio is also closed under shuffle-operation.

3 $\mathcal{L}_{RE} = \hat{\mathcal{F}}_\cap(P_k)$ for Each $k \geq 3$

In [4] we showed that DUP is in $\mathcal{C}_\cap(EXP_2)$. The purpose of this and the following section is to establish that the previous relation remains true if the exponential language is replaced with any P_k such that $k \in \{2, 3, \ldots\}$. The following lemma straightforwardly implies the main result of this section.

Lemma 1. *For each integer $k \geq 3$, the language DUP is in $\mathcal{C}_\cap(P_k)$.*

Proof. Let $k \geq 3$ be an integer and $T_k = \{a^{2^{n^k}} \mid n \in \mathbb{N}\}$. Clearly $T_k \in \mathcal{C}_\cap(P_k)$. Moreover, let h_1, h_2 and h_3 be three morphisms from $\{a, b\}^*$ into $\{a\}^*$ defined by

$$\begin{aligned} h_1(a) &= a,\ h_1(b) = e \\ h_2(a) &= e,\ h_2(b) = a \\ h_3(a) &= a,\ h_3(b) = a\ . \end{aligned}$$

The language $Q_k = h_1^{-1}(P_k) \cap h_2^{-1}(P_k) \cap h_3^{-1}(T_k)$ is certainly in $\mathcal{C}_\cap(P_k)$. Let $w \in Q_k$, $i = |w|_a$ and $j = |w|_b$. By the above construction, there exist $n, m, r \in \mathbb{N}$ such that $i = n^k$, $j = m^k$ and $i + j = 2\,r^k$. We thus have $n^k + m^k = 2\,r^k$. By the result of Darmon and Merel in [1], the equalities $n = m = r$ hold. Obviously

$$Q_k \ = \ \{w \in \{a, b\}^* \mid |w|_a = |w|_b = n^k \text{ for some } n \in \mathbb{N}\}\ .$$

Let g and h be morphisms: $\{a, b, c\}^* \to \{a, b\}^*$ satisfying

$$\begin{aligned} g(c) &= aab,\ g(a) = a,\ g(b) = b \\ h(a) &= a, \quad\ h(c) = a,\ h(b) = b\ . \end{aligned}$$

It is quite straightforward to see that the language $S_k = h(g^{-1}(Q_k)) \cap a^*b^*$ is a subset of DUP such that $DUP \setminus S_k$ is finite. Since certainly S_k is in $\mathcal{C}_\cap(P_k)$, we deduce that also DUP is in $\mathcal{C}_\cap(P_k)$. □

By the previous lemma, $DUP \in \hat{\mathcal{F}}_\cap(P_k)$ for each $k \geq 3$. The result of Hartmanis and Hopcroft in [3] immediately implies

Theorem 1. $\mathcal{L}_{RE} = \hat{\mathcal{F}}_{\cap}(P_k)$ *for each* $k \geq 3$.

Remark 1. The case $k = 2$ was not considered in Lemma 1. The reason is that the Diophantine equation $x^2 + y^2 = 2\,z^2$ has also other integer solutions than those with the property $|x| = |y| = |z|$.

4 $\mathcal{L}_{RE} = \hat{\mathcal{F}}_{\cap}(P_2)$

Consider the system of Diophantine equations

$$\begin{cases} x^2 + y^2 = 2\,z^2 \\ x^2 + 2\,y^2 = 3\,u^2 \end{cases} \tag{3}$$

over the field of rationals. We wish to determine the integer solutions of (3), i.e., quadruples $(x_0, y_0, z_0, u_0) \in \mathbb{Z}^4$ such that $x_0^2 + y_0^2 = 2\,z_0^2$ and $x_0^2 + 2\,y_0^2 = 3\,u_0^2$. Call a solution (x_0, y_0, z_0, u_0) *proper* if $(x_0, y_0, z_0, u_0) \neq (0, 0, 0, 0)$ and the integers x_0, y_0 have no common divisor greater than 1. Clearly $(1, 1, 1, 1)$ is a proper solution of (3). When studying the formula (20) in the book of Nagell [5], page 225 we notice that each proper solution (x_0, y_0, z_0, u_0) of (3) satisfies the following system of equations

$$\begin{cases} \Delta_1\, x_0 = -u_1^2 - 2\,u_1 v_1 + v_1^2 \\ \Delta_1\, y_0 = u_1^2 - 2\,u_1 v_1 - v_1^2 \\ \Delta_2\, x_0 = -u_2^2 - 4\,u_2 v_2 + 2\,v_2^2 \\ \Delta_2\, y_0 = u_2^2 - 2\,u_2 v_2 - 2\,v_2^2 \end{cases} \tag{4}$$

where u_i, v_i are relatively prime integers and Δ_i denotes the greatest common divisor of the right hand sides of the two equations containing u_i and v_i, $i = 1, 2$. The equations of (4) imply straightforwardly

$$\begin{cases} 2\,\Delta_2 u_1 v_1 = 3\,\Delta_1 u_2 v_2 \\ \Delta_2\,(u_1^2 - v_1^2) = \Delta_1\,(u_2^2 - 2 \cdot v_2^2)\ . \end{cases} \tag{5}$$

Solving x with respect to u_2, v_2 from the first equation of (4) and comparing it to the value given in the third equation of the same system allows us to deduce that either u_2 or v_2 is equal to zero. This implies that $|x_0| = |y_0| = |z_0| = |u_0|$. Since each solution of (4) is a integer multiple of a proper one, the following holds true.

Theorem 2. *The quadruple* $(x_0, y_0, z_0, u_0) \in \mathbb{Z}^4$ *is an integer solution of the system of Diophantine equations* (3) *if and only if* $|x_0| = |y_0| = |z_0| = |u_0|$.

The remaining part of our considerations proceeds as in the previous section.

Lemma 2. *The language DUP is in* $\mathcal{C}_{\cap}(P_2)$.

Proof. Suppose that $a, b, , b_1, b_2$ are distinct symbols. The languages

$$\begin{aligned} L_1 &= \{w \in \{a,b\}^* \mid |w|_a = m^2,\ |w|_b = n^2 \text{ for some } m,n \in \mathbb{N}\} \\ L_2 &= \{a^{2\,r^2} \mid r \in \mathbb{N}\} \\ L_3 &= \{a^{3\,s^2} \mid s \in \mathbb{N}\} \end{aligned}$$

are surely in $\mathcal{C}_\cap(P_2)$. Let $h_1 : \{a,b\}^* \to \{a,b_1,b_2\}^*$ and $g_1, g_2 : \{a,b_1,b_2\}^* \to \{a\}^*$ be the morphisms defined by

$$\begin{aligned} h_1(a) &= a,\ h_1(b) = b_1 b_2^2 \\ g_1(a) &= a,\ g_1(b_1) = a,\ g_1(b_2) = e \\ g_2(a) &= a,\ g_2(b_1) = e,\ g_2(b_2) = a\ . \end{aligned}$$

The language $L_4 = h_1(L_1) \cap g_1^{-1}(L_2) \cap g_2^{-1}(L_2)$ is obviously in $\mathcal{C}_\cap(P_2)$. Let $w \in L_4$. By the above construction, there exist $m, n, r, s \in \mathbb{N}$ such that $|w|_a = m^2$, $|w|_{b_1} = n^2$, $|w|_{b_2} = 2\,n^2$, $m^2 + n^2 = 2\,r^2$ and $m^2 + 2\,n^2 = 3\,s^2$. By Theorem 2, we have $m = n = r = s$. Then certainly

$$L_4 = \{w \in \{a, b_1 b_2^2\}^* \mid |w|_{b_1} = |w|_a = n^k,\ |w|_{b_2} = 2\,|w|_a = 2\,n^k \text{ for some } n \in \mathbb{N}\}.$$

Let $g_3 : \{a,b\}^* \to \{a,b_1,b_2\}^*$ be the morphisms defined by $g_3(a) = a$, $g_3(b) = b_1 b_2^2 a$. Then the language

$$L_5 \ = \ g_3^{-1}(L_4) \ = \ \{w \in \{a,b\}^* \mid |w|_a = |w|_b = n^2 \text{ for some } n \in \mathbb{N}\}$$

is in $\mathcal{C}_\cap(P_2)$. As in the previous section, let g and h be morphisms: $\{a,b,c\}^* \to \{a,b\}^*$ defined by

$$\begin{aligned} g(c) &= aab,\ g(a) = a,\ g(b) = b \\ h(a) &= a,\quad h(c) = a,\ h(b) = b\ . \end{aligned}$$

It is quite easy to see that the language $L_6 = h(g^{-1}(L_5)) \cap a^* b^*$ is equal to DUP. Since L_6 is in $\mathcal{C}_\cap(P_k)$, we are through. □

Theorem 3. $\mathcal{L}_{RE} = \hat{\mathcal{F}}_\cap(P_2)$.

Remark 2. To prove Lemma 1 only one Diophantine equation was needed; the integer solutions of the Diophantine equation $x^k + y^k = 2\,z^k$ satisfy $|x| = |y| = |z|$. For the proof of Lemma 2 we apply two 'independent' Diophantine equations $x^k + y^k = 2\,z^k$ and $x^k + 2\,y^k = 3\,z^k$. Together they guarantee that their integer solutions necessarily satisfies $|x| = |y| = |z|$.

5 $\hat{\mathcal{C}}_\cap(BREP) = \hat{\mathcal{C}}_\cap(REP)$

Let a, b, c, a_1, b_1, c_1, a_2, b_2, c_2 be distinct letters in an alphabet. Define

$$\begin{aligned} REP &= \{(a^n b)^m \mid m,n \in \mathbb{N}\} \\ BREP &= \{(a^n b)^n \mid n \in \mathbb{N}\} \quad \text{and} \\ BREP_i &= \{(a_i^n b_i)^m \mid n \in \mathbb{N}\} \quad \text{for } i = 1,2,3. \end{aligned}$$

We have the following

Theorem 4. $\hat{\mathcal{C}}_{\cap}(BREP) = \hat{\mathcal{C}}_{\cap}(REP)$.

Proof. By Theorem 4 of [6] the inclusion $\hat{\mathcal{C}}_{\cap}(BREP) \subseteq \hat{\mathcal{C}}_{\cap}(REP)$ holds. We now show that $REP \in \hat{\mathcal{C}}_{\cap}(BREP)$. Since $\hat{\mathcal{C}}_{\cap}(BREP)$ is closed under intersection, it is closed under shuffle-operation. Thus the languages L_1 and L_2 below are in $\hat{\mathcal{C}}_{\cap}(BREP)$.

$$\begin{aligned} L_1 &= BREP_1 \cdot BREP_2 = \{(a_1^n b_1)^n (a_2^m b_2)^m \mid n, m \in \mathbb{N}\}\ ; \quad \text{and} \\ L_2 &= shuf(L_1, BREP_3) \cap ((a_1 a_3)^* a_3^* b_1 b_2)^* ((a_2 a_3)^* a_3^* b_2 b_3)^* \\ &= \{((a_1 a_3)^n a_3^m b_1 b_3)^n ((a_2 a_3)^m a_3^n b_2 b_3)^m \mid n, m \in \mathbb{N}\} \end{aligned}$$

Let h be the morphism on $\{a_1, b_1, c_1, a_2, b_2, c_2\}^*$ defined by $h(a_1) = a_3$, $h(b_1) = b_1 b_3$, $h(c_1) = a_1 a_3$, $h(a_2) = a_3$, $h(b_2) = b_2 b_3$, $h(c_2) = a_2 a_3$. Then the language

$$\begin{aligned} L_3 &= h^{-1}(L_2) \cap \{a_1, b_1, c_1\}^* \{a_2, b_2, c_2\}^* \\ &= \{(c_1^n a_1^m b_1)^n (c_2^m a_2^n b_2)^m \mid n, m \in \mathbb{N}\} \end{aligned}$$

is in $\hat{\mathcal{C}}_{\cap}(BREP)$. Let g be the morphism on $\{a_1, b_1, c_1, a_2, b_2, c_2\}^*$ defined by $g(b_1) = b$, $g(c_1) = g(a_2) = g(b_2) = g(c_2) = e$. The language

$$g(L_3) = \{(a^m b)^n \mid m, n \in \mathbb{N}\}$$

is equal to REP and is certainly in $\hat{\mathcal{C}}_{\cap}(BREP)$. This completes the proof. □

6 Some Open Problems

We are confined to list some questions that still remain without answer.

Even if $\mathcal{L}_{RE} = \hat{\mathcal{F}}_{\cap}(P_k)$, we state the following

Conjecture 1. For each $k \geq 2$ the language family $\hat{\mathcal{C}}_{\cap}(P_k)$ does not contain all recursively enumerable one-letter languages.

This is equivalent to

Conjecture 2. The language $PROD$ is not in $\hat{\mathcal{C}}_{\cap}(P_k)$ for any integer $k \geq 2$.

A sharper version of the previous conjecture can be so stated:

Conjecture 3. For each pair of integers $j, k \geq 2$ such that $j \neq k$, the language families $\mathcal{C}_{\cap}(P_j)$ and $\mathcal{C}_{\cap}(P_k)$ are incomparable, i.e., neither $\mathcal{C}_{\cap}(P_j) \subseteq \mathcal{C}_{\cap}(P_k)$ nor $\mathcal{C}_{\cap}(P_k) \subseteq \mathcal{C}_{\cap}(P_k)$ holds.

We believe that even a stronger claim is true:

Conjecture 4. For each pair of integers $j, k \geq 2$ such that $j \neq k$, we have $\mathcal{C}_{\cap}(P_j) \cap \mathcal{C}_{\cap}(P_k) = \mathcal{C}_{\cap}(DUP)$.

Finally an interesting research topic

Open Problem Is the language family $\hat{\mathcal{C}}_{\cap}(BREP) = \hat{\mathcal{C}}_{\cap}(REP)$ a proper subset of $\mathcal{L}_{RE}$?

The previous question is equivalent to whether or not the set $\hat{\mathcal{C}}_{\cap}(BREP) = \hat{\mathcal{C}}_{\cap}(REP)$ contains the catenation closure DUP^* of DUP.

References

1. Darmon H., Merel L., Winding quotients and some variants of Fermat's 7Last Theorem, *Journal für die reine und angewandte Mathematik* 490 (1997), 81-100.
2. Ginsburg S., Goldstine J., Intersection-closed full AFL and the recursively enumerable languages, *Information and Control* 22 (1973), 201-231.
3. Hartmanis J., Hopcrroft J., What makes some language theory problems undecidable?, *Journal of Computer and System Sciences* 4 (1970), 368-376.
4. Kortelainen J., On Language Families Generated by Commutative Languages, Annales Academiae Scientiarum Fennicae Series A I. Mathematica Dissertationes 44.
5. Nagell T., *Introduction to Number Theory*, Almqvist & Wiksell, Stockholm, John Wiley & Sons, New York (1951).
6. Turakainen P., On some bounded semiAFLs and AFLs, *Information Sciences* 23 (1981), 31-48.
7. Wiles A.,Modular elliptic-curves and Fermat's Last Theorem, *Annals of Mathematics* 141 (1995), 443-551.

On Language Inequalities $XK \subseteq LX$

Michal Kunc[1,2,⋆]

[1] Department of Mathematics, University of Turku
FIN-20014 Turku, Finland
[2] Turku Centre for Computer Science, FIN-20014 Turku, Finland
kunc@math.muni.cz
http://www.math.muni.cz/~kunc/

Abstract. It is known that for a regular language L and an arbitrary language K the largest solution of the inequality $XK \subseteq LX$ is regular. Here we show that there exist finite languages K and P and star-free languages L, M and R such that the largest solutions of the systems $\{XK \subseteq LX,\ X \subseteq M\}$ and $\{XK \subseteq LX,\ XP \subseteq RX\}$ are not recursively enumerable.

1 Introduction

This paper is a continuation of recent investigations of simple systems of implicit language equations and inequalities where constants are regular languages. These investigations aim to discover which types of systems have largest solutions regular no matter what regular constants they contain and which systems can have largest solutions non-regular (or non-recursive) even though all constants are regular (or even finite or star-free) languages. Known results on this topic are surveyed in [10].

Systems of language equations and inequalities were studied mainly in connection with context-free languages since these languages can be described as components of smallest solutions of systems of explicit polynomial equations, i.e. equations with the operations of union and concatenation. Much less attention was devoted to implicit language equations. Such equations were first considered by Conway [5], who observed that inequalities of the form $E \subseteq L$, where E is a regular function of variables and L is a regular language, possess only finitely many maximal solutions, all of them are regular and computable.

It is well known that regular languages can be characterized as components of smallest or largest solutions of systems of explicit right-linear equations. Regular solutions of more general systems were studied for example by Leiss [13]. For systems of implicit right-linear inequalities, i.e. inequalities of the form

$$K \cup K_1X_1 \cup \cdots \cup K_nX_n \subseteq L \cup L_1X_1 \cup \cdots \cup L_nX_n\,,$$

where K, K_1, …, K_n and L, L_1, …, L_n are constant languages, it is known that their largest solutions are always regular provided all constant languages on

⋆ Supported by the Academy of Finland under grant 208414

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 327–337, 2005.
© Springer-Verlag Berlin Heidelberg 2005

right-hand sides are regular [12]. Moreover, if all constant languages occurring in the system are regular, then computability of the largest solution follows from Rabin's results on MSO logic over infinite trees [16]; actually, the computation of the solution is an ExpTime-complete problem [1–3].

Another result on regularity of solutions was obtained in [9], where well quasi-orders of free monoids were used to prove that all maximal solutions are regular for a large class of systems of inequalities where all constants are languages recognizable by finite simple semigroups.

A common property of the positive results mentioned above is that they can be formulated not only for single equations, but also for arbitrary finite systems of such equations. The aim of this paper is to show that this is not the case for inequalities of the form $XK \subseteq LX$, where K and L are regular languages. As demonstrated by the author [9], if L is a regular language and K is an arbitrary language, the largest solution of the inequality $XK \subseteq LX$ is regular. A related problem of regularity of largest solutions of equations of the form $XL = LX$, which was formulated by Conway [5] in 1971, has recently attracted some attention [4, 6, 7, 17] (see [8] for a survey), and it turned out [11] that largest solutions of such equations are non-recursive even for some finite languages L. This means that the positive result is not preserved if we combine an inequality $XK \subseteq LX$ with a similar inequality where the order of concatenation is reversed. In this paper we show that the same situation as for equations $XL = LX$ arises when we either consider a system of two inequalities of the form $XK \subseteq LX$ or add to such an inequality an additional restriction on the solution provided by a regular language. Moreover, to obtain this result it is sufficient to consider only finite languages K and star-free languages L. On the other hand, the complements of largest solutions of such systems of inequalities are always recursively enumerable [15]. As in our proofs we encode the complement of an arbitrary language computed by a Minsky machine into a largest solution, we actually obtain for these systems the most negative possible result.

Basic notions employed in our considerations are recalled in the following section. For a more comprehensive introduction to formal languages the reader is referred to [18].

2 Preliminaries

We denote the sets of positive and non-negative integers by $\mathbb{N}$ and $\mathbb{N}_0$, respectively. Throughout the paper we consider a finite alphabet A. As usual, we write A^+ for the set of all non-empty finite words over A, and A^* for the set obtained from A^+ by adding the empty word ε. If $u, v, w \in A^*$ are words such that $w = uv$, then u and v are called a *prefix* and a *suffix* of w, respectively.

Languages over A are arbitrary subsets of A^*. The basic operation on languages is concatenation defined by the rule $K \cdot L = \{uv \mid u \in K,\ v \in L\}$, and we use the standard notation $L^+ = \bigcup_{m\in\mathbb{N}} L^m$ and $L^* = L^+ \cup \{\varepsilon\}$. Further, we write KL^{-1} for the language $\{u \in A^* \mid \exists v \in L\colon uv \in K\}$. *Regular* languages are languages definable by finite automata, or equivalently, by rational expres-

sions. The basic tool for proving non-regularity of languages is the well-known pumping lemma (see e.g. [18]). A language $L \subseteq A^*$ is called *star-free* if it can be obtained from finite languages using the operations of union, complementation and concatenation; in particular, for every $B \subseteq A$, the languages B^+ and B^* are star-free, and if $B, C \subseteq A$ are disjoint subalphabets of A, then $(BC)^+$ is a star-free language as well.

3 Systems with Inequalities $XK \subseteq LX$

The aim of this section is to prove the results of this paper. First, we construct a system consisting of one inequality $XK \subseteq LX$ and an additional constraint $X \subseteq M$ whose largest solution is not recursive and then we encode this system into a system of two inequalities of the form $XK \subseteq LX$. Notice that for any constant languages K, L and M, an arbitrary system consisting of such inequalities possesses the largest solution which can be obtained as the union of all its solutions (because concatenation distributes over infinite union and therefore the union of arbitrarily many solutions is again a solution).

Theorem 1. *There exists a finite language K and star-free languages L, M such that the largest solution of the system*

$$XK \subseteq LX, \quad X \subseteq M \tag{1}$$

is not recursively enumerable.

Before presenting the proof of this theorem, let us demonstrate its basic idea by encoding testing of equality of two counters into the largest solution of a system of the form (1), which is enough to conclude that the solution is non-regular.

Example 1. Consider the alphabet $B = \{a, b, c\}$ together with its disjoint copy $\hat{B} = \{\hat{a}, \hat{b}, \hat{c}\}$ and let $A = B \cup \hat{B}$. Then define languages $K = \{a\hat{a}, b\hat{b}, c\hat{c}\}$,

$$L = \{a\hat{a}, b\hat{b}a\hat{a}, c\hat{c}a\hat{a}, b\hat{b}c\hat{c}b\} \cup \hat{B}B \cup cA^*a \cup bA^*a \cup A^+bA^*b \cup A^+cA^*c$$

and $M = (B\hat{B})^+ \cup \hat{B}$ over A and let us denote by S the largest solution of the system (1) obtained in this way. We are going to prove that the language S is not regular by means of the pumping lemma.

First we show that for every $n \in \mathbb{N}_0$ the word $b\hat{b}(a\hat{a})^n c\hat{c}(a\hat{a})^n$ belongs to the language S by constructing a solution N of (1) which contains all these words. The required solution is defined as

$$N = \hat{B} \cup \{u_{n,m}, v_{n,m} \mid n \in \mathbb{N}_0,\ 0 \leq m \leq n\},$$

where $u_{n,m} = (a\hat{a})^m b\hat{b}(a\hat{a})^n c\hat{c}(a\hat{a})^{n-m}$ and $v_{n,m} = (a\hat{a})^m c\hat{c}(a\hat{a})^{n+1} b\hat{b}(a\hat{a})^{n-m}$.

Let us verify that $NK \subseteq LN$. Clearly $\hat{B}K \subseteq (\hat{B}B) \cdot \hat{B} \subseteq LN$. Now let us show that $u_{n,m}K \subseteq LN$. First, we have

$$u_{n,0} \cdot a\hat{a} \in bA^*a \cdot \hat{B} \subseteq LN\,,$$
$$u_{n,m+1} \cdot a\hat{a} = a\hat{a} \cdot u_{n,m} \in LN\,.$$

For the word $b\hat{b}$ we distinguish three cases:

$$u_{0,0} \cdot b\hat{b} = b\hat{b}c\hat{c}b \cdot \hat{b} \in LN\,,$$
$$u_{n+1,0} \cdot b\hat{b} = b\hat{b}a\hat{a} \cdot v_{n,n} \in LN\,,$$
$$u_{n,m+1} \cdot b\hat{b} \in A^+bA^*b \cdot \hat{B} \subseteq LN\,.$$

And finally, for the word $c\hat{c} \in K$ we directly obtain $u_{n,m} \cdot c\hat{c} \in A^+cA^*c \cdot \hat{B} \subseteq LN$.

For every word $v_{n,m} \in N$ the verification is similar. And because the fact $N \subseteq M$ is trivial, we have actually shown that N is a solution of (1) and therefore the word $b\hat{b}(a\hat{a})^nc\hat{c}(a\hat{a})^n$ belongs to the largest solution S for every $n \in \mathbb{N}_0$.

Now we are going to show that $b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \notin S$ for every $m, n \in \mathbb{N}_0$ satisfying $m < n$. The proof proceeds by induction with respect to m.

First, let $m = 0$ and assume for contradiction that $b\hat{b}c\hat{c}(a\hat{a})^n \in S$. Then we obtain

$$b\hat{b}c\hat{c}(a\hat{a})^n \cdot b\hat{b} \in SK \subseteq LS \subseteq LM\,,$$

which is not true because $n \geq 1$. Therefore $b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \notin S$ holds for $m = 0$.

Now consider the case when $m \geq 1$. The induction hypothesis states that $b\hat{b}(a\hat{a})^{m-1}c\hat{c}(a\hat{a})^{n-1} \notin S$ and we prove the required fact $b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \notin S$ by contradiction. From $b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \in S$ it follows that

$$b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \cdot b\hat{b}(a\hat{a})^{m-1} \in SK^m \subseteq L^mS \subseteq L^mM\,.$$

One can see that every prefix of this word belonging to L^m is either $b\hat{b}(a\hat{a})^m$ or an element of $bA^*a \cdot (\hat{B}B)^{m-1}$. Because in the latter case, for none of such prefixes the corresponding suffix belongs to the language M, we deduce that $c\hat{c}(a\hat{a})^nb\hat{b}(a\hat{a})^{m-1} \in S$. Now we repeat the previous argument using the word $c\hat{c}(a\hat{a})^{n-1}$ from K^n instead of $b\hat{b}(a\hat{a})^{m-1}$ and obtain $b\hat{b}(a\hat{a})^{m-1}c\hat{c}(a\hat{a})^{n-1} \in S$, which contradicts the induction hypothesis.

Hence we have demonstrated both $b\hat{b}(a\hat{a})^nc\hat{c}(a\hat{a})^n \in S$ for every $n \in \mathbb{N}_0$ and $b\hat{b}(a\hat{a})^mc\hat{c}(a\hat{a})^n \notin S$ for $m < n$, and so the largest solution of (1) is not regular due to the pumping lemma.

To prove the theorem we use essentially the same method as in the above example, but this time we encode into the system an arbitrary Minsky machine.

Proof (of Theorem 1). Let $\mathcal{M}$ be a Minsky machine [14] which computes a non-recursive set of non-negative integers. The machine consists of two counters and a finite set of states Q, which is a disjoint union

$$Q = T_1 \cup T_2 \cup I_1 \cup I_2 \cup D_1 \cup D_2 \cup \{1\}\,,$$

where 1 is the terminal state. We assume that the initial state 0 of $\mathcal{M}$ belongs to I_1. A configuration of the machine is a triple (i, m, n), where $i \in Q$ is a current state and $m, n \in \mathbb{N}_0$ are values stored in the counters. The step performed by the machine in a given state is determined by the instruction associated with this state:

- From the state $i \in T_s$, $s \in \{1, 2\}$, the machine goes to the state $\tau_0(i)$ if counter s is empty and to the state $\tau_1(i)$ otherwise, where $\tau_0(i) \neq i$ and $\tau_1(i) \neq i$ are distinct states.
- When the machine is in the state $i \in I_s$ (or $i \in D_s$), it increments (decrements, respectively) counter s and goes to the state $\tau(i) \neq i$.
- When the machine reaches the state 1, the computation stops.

We write $(i, m, n) \to (j, k, l)$ whenever the step of $\mathcal{M}$ performed in the configuration (i, m, n) produces the configuration (j, k, l).

The machine computes the set $\mathcal{L}(\mathcal{M}) \subseteq \mathbb{N}_0$ of all numbers n such that the computation of $\mathcal{M}$ on the initial configuration $(0, 0, n)$ eventually reaches the terminal state. Since we have chosen $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M})$ is not recursive, its complement $\mathbb{N}_0 \setminus \mathcal{L}(\mathcal{M})$ is not recursively enumerable.

We assume that every decrement instruction of the machine $\mathcal{M}$ is preceded by the appropriate zero-test instruction, and from now on we consider only configurations (i, m, n) satisfying $m \geq 1$ if $i \in D_1$ and $n \geq 1$ if $i \in D_2$. Then the computation of $\mathcal{M}$ on every such configuration never reaches a decrement instruction when the corresponding counter is empty, and so every computation either stops in a terminal configuration or continues forever.

In addition, let us assume that if the machine can get to a certain state in one step from two different states i and j, then both states i and j belong to D_1; this can be easily achieved for instance by inserting an increment instruction followed by a decrement instruction after every instruction.

Consider the alphabet $B = \{a\} \cup \{b_i, c_i \mid i \in Q\}$ together with its disjoint copy $\hat{B} = \{\hat{a}\} \cup \{\hat{b}_i, \hat{c}_i \mid i \in Q\}$ and let $A = B \cup \hat{B}$. Every configuration (i, m, n) of the machine $\mathcal{M}$ will be represented by the word $b_i \hat{b}_i (a\hat{a})^m c_i \hat{c}_i (a\hat{a})^n$. Such a word will belong to the largest solution of system (1) if and only if the computation of $\mathcal{M}$ on (i, m, n) does not stop.

Now we define the languages K, L, M over A. First, let

$$\begin{aligned} K = {} & \{a\hat{a}\} \cup \{b_{\tau_0(i)} \hat{b}_{\tau_0(i)}, b_{\tau_1(i)} \hat{b}_{\tau_1(i)} \mid i \in T_1 \cup T_2\} \\ & \cup \{b_{\tau(i)} \hat{b}_{\tau(i)} a\hat{a} \mid i \in I_1\} \\ & \cup \{b_{\tau(i)} \hat{b}_{\tau(i)} \mid i \in I_2 \cup D_1 \cup D_2\} \\ & \cup \{c_{\tau_0(i)} \hat{c}_{\tau_0(i)}, c_{\tau_1(i)} \hat{c}_{\tau_1(i)} \mid i \in T_1 \cup T_2\} \\ & \cup \{c_{\tau(i)} \hat{c}_{\tau(i)} a\hat{a} \mid i \in I_2\} \\ & \cup \{c_{\tau(i)} \hat{c}_{\tau(i)} \mid i \in I_1 \cup D_1 \cup D_2\}. \end{aligned}$$

The language L is defined as the union of the following star-free languages L_0 through L_{12}:

$$
\begin{aligned}
L_0 &= \{a\hat{a}\} \cup \hat{B}K\hat{B}^{-1}\,,\\
L_1 &= \{b_i\hat{b}_i \mid i \in T_1 \cup T_2 \cup I_1 \cup I_2 \cup D_2\}\,,\\
L_2 &= \{b_i\hat{b}_i a\hat{a} \mid i \in D_1\}\,,\\
L_3 &= \{c_i\hat{c}_i \mid i \in T_1 \cup T_2 \cup I_1 \cup I_2 \cup D_1\}\,,\\
L_4 &= \{c_i\hat{c}_i a\hat{a} \mid i \in D_2\}\,,\\
L_5 &= \{b_i \mid i \in Q \setminus \{1\}\} \cdot (A \setminus \{b_i \mid i \in Q\})^* \cdot a\,,\\
L_6 &= \{c_i \mid i \in Q\} \cdot (A \setminus \{c_i \mid i \in Q\})^* \cdot a\,,\\
L_7 &= \bigcup\{A^*c_iA^*b_jA^* \mid i,j \in Q,\ j \notin \{\tau(i), \tau_0(i), \tau_1(i)\}\} \cap A^*B\,,\\
L_8 &= \bigcup\{A^*b_iA^*c_jA^* \mid i,j \in Q,\ i \neq j\} \cap A^*B\,,\\
L_9 &= A^+ \cdot \{b_i \mid i \in Q\} \cdot A^* \cdot \{b_i \mid i \in Q\} \cdot A^* \cap A^*B\,,\\
L_{10} &= A^+ \cdot \{c_i \mid i \in Q\} \cdot A^* \cdot \{c_i \mid i \in Q\} \cdot A^* \cap A^*B\,,\\
L_{11} &= \bigcup\{b_i\hat{b}_iA^+c_iA^*b_{\tau_0(i)} \cup b_i\hat{b}_ic_iA^*b_{\tau_1(i)} \mid i \in T_1\}\,,\\
L_{12} &= \bigcup\{A^*c_i\hat{c}_iA^+b_{\tau_0(i)} \cup A^*c_i\hat{c}_ib_{\tau_1(i)} \mid i \in T_2\}\,.
\end{aligned}
$$

Finally, let $M = (B\hat{B})^+ \cup \hat{B}$ and denote by S the largest solution of system (1).

Let us briefly sketch out how computations of the Minsky machine are simulated by manipulations of words from A^+. The computation basically proceeds by removing pairs of letters from the beginning of a word $b_i\hat{b}_i(a\hat{a})^mc_i\hat{c}_i(a\hat{a})^n$ and appending them to its end. In addition, letters b_i, $\hat{b}_i$, c_i and $\hat{c}_i$ are replaced by the letters corresponding to the next state of the machine, and when for instance the first counter should be incremented, then instead of the pair $b_{\tau(i)}\hat{b}_{\tau(i)}$ the word $b_{\tau(i)}\hat{b}_{\tau(i)}a\hat{a}$ is appended, so the number of pairs $a\hat{a}$ in the resulting word increases.

This process is controlled by system (1) as follows. If some word $u \in S$ obtained during our manipulations is concatenated with a word $v \in K$, the resulting word uv should belong to the language LS. In fact, most of the words uv lie in $L\hat{B}$, and because all elements of the alphabet $\hat{B}$ belong to S thanks to the language L_0, such words uv lie also in LS. For every u there is just one exceptional word v corresponding to the correct computation of the machine. And if u does not to belong to S, the only reason for this could be that uv is not in LS for the exceptional word v. This word uv then possesses only one decomposition wu' into a word w from L and a word u' from M. In this way we achieve that u lies in S if and only if u' lies in S. Therefore every computation of the machine $\mathcal{M}$ preserves the properties that the word corresponding to a configuration belongs, or does not belong, to the solution S.

Incorrect computations of the machine are described by the languages L_5 through L_{12}. The language M is used to ensure that any word from one of these languages can be removed from uv only if its removal produces a one-letter word; otherwise it would produce a word from $\hat{B}A^+$, which does not belong to M and consequently also to S. This allows us to describe certain properties

of the word uv as a whole, not only properties of its prefixes. For instance the language L_5 serves for dealing with words uv starting with a certain letter b_i and not containing other occurrences of such letters; these words arise if one appends some v not containing any letter b_q, $q \in Q$, although b_i is just going to be removed from the beginning of u. Notice that the terminal state 1 is treated differently from other states in the definition of L_5, namely, also incorrect computations are considered as correct here to ensure that words corresponding to terminal configurations do not belong to S.

Let us now turn to the formal proof. Because the complement of $\mathcal{L}(\mathcal{M})$ is not recursively enumerable, to show that the solution S is not recursively enumerable, it is enough to prove that the equivalence

$$n \notin \mathcal{L}(\mathcal{M}) \iff b_0\hat{b}_0c_0\hat{c}_0(a\hat{a})^n \in S \tag{2}$$

is true for every $n \in \mathbb{N}_0$.

Let us consider the set C of all configurations (i, m, n) of $\mathcal{M}$ such that the computation of $\mathcal{M}$ starting in (i, m, n) never stops. The rest of the proof is devoted to verifying that for every configuration (i, m, n) of $\mathcal{M}$:

$$(i, m, n) \in C \iff b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \in S\,. \tag{3}$$

Then we obtain (2) by taking $i = 0$ and $m = 0$, because $(0, 0, n) \in C$ is equivalent to $n \notin \mathcal{L}(\mathcal{M})$.

In order to verify the direct implication of (3), we construct a language N which satisfies (1) and contains the word $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n$ for all $(i, m, n) \in C$:

$$\begin{aligned} N = \hat{B} \cup \{&b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n, (a\hat{a})^p c_i\hat{c}_i(a\hat{a})^n b_j\hat{b}_j(a\hat{a})^{k-p}, \\ &(a\hat{a})^r b_j\hat{b}_j(a\hat{a})^k c_j\hat{c}_j(a\hat{a})^{l-r} \mid (i, m, n) \in C,\ (i, m, n) \to (j, k, l), \\ &0 \le p \le \min(m, k),\ 0 \le r \le \min(n, l)\}\,. \end{aligned}$$

Let us verify that $NK \subseteq LN$. Clearly $\hat{B}K \subseteq L_0\hat{B} \subseteq LN$. Now take any configuration $(i, m, n) \in C$, and assume that $(i, m, n) \to (j, k, l)$.

First, consider the word $u = b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \in N$ and let us show that $uK \subseteq LN$. We have $ua\hat{a} \in L_5\hat{B} \subseteq LN$ and if we take an arbitrary state $q \in Q \setminus \{\tau(i), \tau_0(i), \tau_1(i)\}$, then it is clear that both words $ub_q\hat{b}_q$ and $ub_q\hat{b}_qa\hat{a}$ belong to $L_7\hat{B} \subseteq LN$.

Further, notice that our additional assumption about the machine $\mathcal{M}$ ensures that for every $q \in Q$ only one of the words $b_q\hat{b}_q$ and $b_q\hat{b}_qa\hat{a}$ lies in K (and similarly for the words $c_q\hat{c}_q$ and $c_q\hat{c}_qa\hat{a}$). Therefore if we take q equal to one of the states $\tau(i)$, $\tau_0(i)$ and $\tau_1(i)$, there is always only one word from K to deal with. For such states q we have to distinguish several cases according to the instruction associated with the state i.

Let us start with a state $i \in T_1$. Here we directly obtain $ub_{\tau_0(i)}\hat{b}_{\tau_0(i)} \in L_{11}\hat{B}$ if $m \neq 0$ and $ub_{\tau_1(i)}\hat{b}_{\tau_1(i)} \in L_{11}\hat{B}$ if $m = 0$. Further, we have $k = m$ and $l = n$. For $m \neq 0$ we get $j = \tau_1(i)$, which implies $ub_{\tau_1(i)}\hat{b}_{\tau_1(i)} \in L_1N$ since the

word $(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n b_{\tau_1(i)}\hat{b}_{\tau_1(i)}$ can be proved to belong to N by taking $p = m$. Similarly, for $m = 0$ we get $j = \tau_0(i)$ and therefore $ub_{\tau_0(i)}\hat{b}_{\tau_0(i)} \in L_1N$.

The case of $i \in T_2$ is analogous; one has to employ the language L_{12} in place of L_{11}.

If $i \in I_1$ then $ub_{\tau(i)}\hat{b}_{\tau(i)}a\hat{a} \in L_1N$, and if $i \in I_2 \cup D_2$ then $ub_{\tau(i)}\hat{b}_{\tau(i)} \in L_1N$; both facts can be verified by setting $p = m$. And for $i \in D_1$ we have to use the value $p = m - 1$ to obtain $ub_{\tau(i)}\hat{b}_{\tau(i)} \in L_2N$.

Finally, for an arbitrary state $q \in Q$, both words $uc_q\hat{c}_q$ and $uc_q\hat{c}_qa\hat{a}$ belong to $L_{10}\hat{B}$.

Now consider a word $u = (a\hat{a})^p c_i\hat{c}_i(a\hat{a})^n b_j\hat{b}_j(a\hat{a})^{k-p} \in N$ where $p \neq 0$ or $u = (a\hat{a})^r b_j\hat{b}_j(a\hat{a})^k c_j\hat{c}_j(a\hat{a})^{l-r} \in N$ where $r \neq 0$. Then the fact $ua\hat{a} \in L_0N$ follows by taking $p - 1$ in place of p and $r - 1$ in place of r, respectively. And if $q \in Q$ is an arbitrary state, then both words $ub_q\hat{b}_q$ and $ub_q\hat{b}_qa\hat{a}$ belong to $L_9\hat{B}$ and both words $uc_q\hat{c}_q$ and $uc_q\hat{c}_qa\hat{a}$ belong to $L_{10}\hat{B}$.

Next, let $p = 0$ and take the corresponding word $u = c_i\hat{c}_i(a\hat{a})^n b_j\hat{b}_j(a\hat{a})^k \in N$. Clearly $ua\hat{a} \in L_6\hat{B}$ and $ub_q\hat{b}_q, ub_q\hat{b}_qa\hat{a} \in L_9\hat{B}$ for any $q \in Q$. Further, for $q \in Q \setminus \{j\}$, both $uc_q\hat{c}_q$ and $uc_q\hat{c}_qa\hat{a}$ lie in $L_8\hat{B}$. If $i \in T_1 \cup T_2 \cup I_1 \cup D_1$ then $uc_j\hat{c}_j \in L_3N$, because we can put $r = n$. In the case $i \in I_2$, we have $uc_j\hat{c}_ja\hat{a} \in L_3N$ by the same argument. And for $i \in D_2$ we can verify that $uc_j\hat{c}_j \in L_4N$ by setting $r = n - 1$. This concludes the verification for words of the second form from the definition of N.

Finally, notice that for $r = 0$ every word of the third form becomes a word of the first form, since $(j, k, l) \in C$.

Hence, our claim $NK \subseteq LN$ is proved. Because the fact $N \subseteq M$ is trivial, we have actually verified that N is a solution of (1). Therefore $N \subseteq S$, and so $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \in S$ holds for every configuration $(i, m, n) \in C$. This shows that the direct implication of (3) is true.

To complete the proof, we have to verify the converse implication of (3). We are going to prove $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \notin S$ for every configuration $(i, m, n) \notin C$ by induction with respect to the length of a run of $\mathcal{M}$ reaching a terminal configuration from (i, m, n).

First, let $(1, m, n)$ be an arbitrary terminal configuration. Assuming that $u = b_1\hat{b}_1(a\hat{a})^m c_1\hat{c}_1(a\hat{a})^n \in S$, we obtain a contradiction by considering the word $u \cdot a\hat{a} \in SK$ since $ua\hat{a}$ has no prefix belonging to L and therefore $ua\hat{a} \notin LS$.

Now let $(i, 0, n) \notin C$ be a configuration of $\mathcal{M}$ where $i \in T_1$. The step of $\mathcal{M}$ performed in this configuration produces the configuration $(\tau_0(i), 0, n) \notin C$. By the induction hypothesis, we have $b_{\tau_0(i)}\hat{b}_{\tau_0(i)}c_{\tau_0(i)}\hat{c}_{\tau_0(i)}(a\hat{a})^n \notin S$. We prove $b_i\hat{b}_ic_i\hat{c}_i(a\hat{a})^n \notin S$ by contradiction. If $b_i\hat{b}_ic_i\hat{c}_i(a\hat{a})^n \in S$ then

$$b_i\hat{b}_ic_i\hat{c}_i(a\hat{a})^n \cdot b_{\tau_0(i)}\hat{b}_{\tau_0(i)} \in SK \subseteq LS\,, \tag{4}$$

and therefore $c_i\hat{c}_i(a\hat{a})^n b_{\tau_0(i)}\hat{b}_{\tau_0(i)} \in S$ because $b_i\hat{b}_i \cdot c_i\hat{c}_i(a\hat{a})^n b_{\tau_0(i)}\hat{b}_{\tau_0(i)}$ is the only decomposition of the word (4) into a word from L and a word from M. Now we have

$$c_i\hat{c}_i(a\hat{a})^n b_{\tau_0(i)}\hat{b}_{\tau_0(i)} \cdot c_{\tau_0(i)}\hat{c}_{\tau_0(i)} \in SK \subseteq LS\,,$$

and repeating the previous argument, we obtain $(a\hat{a})^n b_{\tau_0(i)}\hat{b}_{\tau_0(i)}c_{\tau_0(i)}\hat{c}_{\tau_0(i)} \in S$. Consequently,

$$(a\hat{a})^n b_{\tau_0(i)}\hat{b}_{\tau_0(i)}c_{\tau_0(i)}\hat{c}_{\tau_0(i)} \cdot (a\hat{a})^n \in SK^n \subseteq L^n S\,,$$

and because this word possesses only one prefix belonging to L^n, namely $(a\hat{a})^n$, one can deduce that $b_{\tau_0(i)}\hat{b}_{\tau_0(i)}c_{\tau_0(i)}\hat{c}_{\tau_0(i)}(a\hat{a})^n \in S$, which is a contradiction.

Analogously, one can deal with every configuration $(i, m, n) \notin C$ where $i \in T_1$ and $m \geq 1$.

Now take a state $i \in I_1$ and consider any configuration $(i, m, n) \notin C$. As this configuration is followed by the configuration $(\tau(i), m+1, n)$ in the computation of $\mathcal{M}$, the induction hypothesis states that $b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^{m+1}c_{\tau(i)}\hat{c}_{\tau(i)}(a\hat{a})^n \notin S$. Assuming $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \in S$, one gets

$$b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \cdot b_{\tau(i)}\hat{b}_{\tau(i)}a\hat{a} \in SK \subseteq LS\,,$$

and since there is only one decomposition of this word into a word from L and a word from M, one directly verifies that $(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)}a\hat{a} \in S$. Therefore

$$(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)}a\hat{a} \cdot (a\hat{a})^m \in SK^m \subseteq L^m S\,,$$

which implies $c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^{m+1} \in S$ because $(a\hat{a})^m$ is the only prefix belonging to L^m. Continuing this way, using c's instead of b's, we obtain $b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^{m+1}c_{\tau(i)}\hat{c}_{\tau(i)}(a\hat{a})^n \in S$. This is a contradiction, and so we have proved $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \notin S$.

Further, let us show how to deal with a state $i \in D_2$. In this case, a configuration $(i, m, n) \notin C$, where $n \geq 1$, is followed by the configuration $(\tau(i), m, n-1)$. The induction hypothesis now states $b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m c_{\tau(i)}\hat{c}_{\tau(i)}(a\hat{a})^{n-1} \notin S$ and we assume that $b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \in S$. Then

$$b_i\hat{b}_i(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n \cdot b_{\tau(i)}\hat{b}_{\tau(i)} \in SK \subseteq LS\,,$$

and the uniqueness of the decomposition of this word into words from L and M shows that $(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)} \in S$. Consequently,

$$(a\hat{a})^m c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)} \cdot (a\hat{a})^m \in SK^m \subseteq L^m S\,,$$

and as this word has only one prefix belonging to the language L^m, we obtain $c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m \in S$. This implies

$$c_i\hat{c}_i(a\hat{a})^n b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m \cdot c_{\tau(i)}\hat{c}_{\tau(i)} \in SK \subseteq LS\,.$$

Again, there is only one prefix from L such that the corresponding suffix belongs to M, namely $c_i\hat{c}_i a\hat{a}$, which gives $(a\hat{a})^{n-1}b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m c_{\tau(i)}\hat{c}_{\tau(i)} \in S$. Finally, we have

$$(a\hat{a})^{n-1}b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m c_{\tau(i)}\hat{c}_{\tau(i)} \cdot (a\hat{a})^{n-1} \in SK^{n-1} \subseteq L^{n-1}S\,,$$

and by removing the only prefix of this word belonging to L^{n-1} we obtain $b_{\tau(i)}\hat{b}_{\tau(i)}(a\hat{a})^m c_{\tau(i)}\hat{c}_{\tau(i)}(a\hat{a})^{n-1} \in S$. Thus we have reached a contradiction.

The remaining cases of configurations (i, m, n), where the state i belongs to one of the sets T_2, I_2 and D_1, can be handled similarly. □

Theorem 2. *There exist finite languages K, P and star-free languages L, R such that the largest solution of the system*

$$XK \subseteq LX, \quad XP \subseteq RX \tag{5}$$

is not recursively enumerable.

Proof. Let us consider the languages K, L, M over the alphabet A defined in the proof of Theorem 1. We enrich the alphabet with a new letter by setting $\tilde{A} = A \cup \{d\}$ and define the languages $P = \{d\hat{a}\}$ and $R = Md$ over $\tilde{A}$. We are going to prove that any language containing the word $\hat{a}$ is a solution of system (5) if and only if it is a solution of (1). This in particular means that the largest solutions of these systems are equal, which proves that the largest solution of (5) is not recursively enumerable.

It is clear that every solution N of (1) satisfying $\hat{a} \in N$ is a solution of (5) too. Conversely, assume that a language N is a solution of (5). First notice that the inequality $XK \subseteq LX$ implies $NK^n \subseteq L^n N$ for arbitrarily large $n \in \mathbb{N}$, and since $\varepsilon \notin L$, we have $N \subseteq A^*$. Therefore the letter d occurs only once in every word of the language $NP = Nd\hat{a}$, and from the definition of R one can immediately see that the inclusion $NP \subseteq RN$ implies $N \subseteq M$. □

References

1. Aiken, A., Kozen, D., Vardi, M., Wimmers, E.: The complexity of set constraints. In *Proc. CSL '93*, LNCS 832, Springer (1994) 1–17.
2. Baader, F., Küsters, R.: Unification in a description logic with transitive closure of roles. In *Proc. LPAR 2001*, LNCS 2250, Springer (2001) 217–232.
3. Charatonik, W., Podelski, A.: Co-definite set constraints. In *Proc. RTA-98*, LNCS 1379, Springer (1998) 211–225.
4. Choffrut, C., Karhumäki, J., Ollinger, N.: The commutation of finite sets: A challenging problem. *Theoret. Comput. Sci.* **273** (2002) 69–79.
5. Conway, J.H.: *Regular Algebra and Finite Machines*. Chapman and Hall (1971).
6. Karhumäki J., Latteux M., Petre, I.: Commutation with codes. *Theoret. Comput. Sci.*, to appear.
7. Karhumäki J., Latteux M., Petre, I.: Commutation with ternary sets of words. *Theory Comput. Syst.* **38**(2) (2005) 161–169.
8. Karhumäki, J., Petre, I.: Two problems on commutation of languages. In *Current Trends in Theoretical Computer Science, The Challenge of the New Century, vol. 2*, World Scientific (2004) 477–493.
9. Kunc, M.: Regular solutions of language inequalities and well quasi-orders. *Theoret. Comput. Sci.*, to appear. Extended abstract in *Proc. ICALP 2004*, LNCS 3142, Springer (2004) 870–881.

10. Kunc, M.: Simple language equations. *Bull. EATCS* **85** (2005) 81–102.
11. Kunc, M.: The power of commuting with finite sets of words. Manuscript (2004), available at `http://www.math.muni.cz/~kunc/`. Extended abstract in *Proc. STACS 2005*, LNCS 3404, Springer (2005).
12. Kunc, M.: Largest solutions of left-linear language inequalities. Manuscript (2005), available at `http://www.math.muni.cz/~kunc/`
13. Leiss, E.L.: *Language Equations*. Springer (1999).
14. Minsky, M.L.: *Computation: Finite and Infinite Machines*. Prentice-Hall (1967).
15. Okhotin, A.: Decision problems for language equations. Submitted for publication, available at `http://www.cs.queensu.ca/home/okhotin/`. Preliminary version in *Proc. ICALP 2003*, LNCS 2719, Springer (2003) 239–251.
16. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.* **141** (1969) 1–35.
17. Ratoandromanana, B.: Codes et motifs. *RAIRO Inform. Théor. Appl.* **23**(4) (1989) 425–444.
18. Rozenberg, G., Salomaa, A. (eds.): *Handbook of Formal Languages*. Springer (1997).

The Power of Tree Series Transducers of Type I and II

Andreas Maletti*

Technische Universität Dresden, Fakultät Informatik
D-01062 Dresden, Germany
maletti@tcs.inf.tu-dresden.de

Abstract. The power of tree series transducers of type I and II is studied for IO as well as OI tree series substitution. More precisely, it is shown that the IO tree series transformations of type I (respectively, type II) are characterized by the composition of homomorphism top-down IO tree series transformations with bottom-up (respectively, linear bottom-up) IO tree series transformations. On the other hand, polynomial OI tree series transducers of type I and II and top-down OI tree series transducers are equally powerful.

1 Introduction

In [1] (restricted) top-down tree transducers were generalized to tree series transducers [2, 3], in which each transition carries a weight taken from a semiring. It was shown in Corollary 14 of [1] that nondeleting and linear top-down tree series transformations preserve recognizable tree series [4–6]. In a sequel [7], Kuich also showed that nondeleting, linear top-down tree series transformations are closed under composition (see Theorem 2.4 in [7]). He built on those two properties the theory of full abstract families of tree series [7]. These results leave an unexplained gap because nondeletion is not required for these results in tree transducer theory; *i. e.*, linear top-down tree transformations with regular look-ahead [8] preserve recognizable tree languages and are closed under composition. Consequently, the survey [9] poses Question 2, which asks for the power of tree series transducers which allow look-ahead and copying of output trees [2, 3].

In the unweighted case, linear top-down tree transducers with regular look-ahead are as powerful as linear bottom-up tree transducers, which was shown in Theorem 5.13 of [8]. Moreover, the power of generalized finite-state tree transducers (respectively, top-down tree transducers with regular look-ahead) is characterized by the composition of a homomorphism and a bottom-up (respectively, linear bottom-up) tree transformation (see Theorems 5.10 and 5.15 of [8]). In this paper we show that these results generalize nicely to tree series transducers. In particular, we show that the linear tree series transducers of type II, which

* Financially supported by the German Research Foundation (DFG, GK 334/3)

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 338–349, 2005.
© Springer-Verlag Berlin Heidelberg 2005

are the canonical generalization of top-down tree transducers with regular look-ahead, compute exactly the class of linear bottom-up tree series transformations (see Theorem 4). Similarly, we study the canonical extension of generalized finite-state tree transducers, which are called tree series transducers of type I in the sequel. We show that the class of tree series transformations of type I (respectively, of type II) coincides with the composition of the class of homomorphism top-down tree series transformations with the class of bottom-up (respectively, linear bottom-up) tree series transformations (see Theorem 3). Altogether we obtain the analogue of the diagram presented on page 228 of [8] for IO tree series transformations over commutative and $\aleph_0$-complete semirings.

Finally, we investigate tree series transducers of type I and II using OI-substitution and thereby address Question 2 as originally posed in [9]. It turns out that polynomial tree series transducers of type I and II and top-down tree series transducers are equally powerful.

2 Preliminaries

We use $\mathbb{N}$ to represent the nonnegative integers $\{0, 1, 2, \dots\}$ and $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. In the sequel, let $k, n \in \mathbb{N}$ and $[k]$ be an abbreviation for $\{\, i \in \mathbb{N} \mid 1 \leqslant i \leqslant k \,\}$. A set Σ which is nonempty and finite is also called an *alphabet*, and the elements thereof are called *symbols*. As usual, Σ^* denotes the set of all finite sequences of symbols of Σ (also called Σ-words). Given $w \in \Sigma^*$, the *length of* w is denoted by $|w|$, and for every $1 \leqslant i \leqslant |w|$ the i-th symbol in w is denoted by w_i (*i. e.*, $w = w_1 \cdots w_{|w|}$).

A *ranked alphabet* is an alphabet Σ together with a mapping $\mathrm{rk}_\Sigma \colon \Sigma \longrightarrow \mathbb{N}$, which associates to each symbol a *rank*. We use the denotation Σ_k to represent the set of symbols (of Σ) which have rank k. Furthermore, we use the set $\mathrm{X} = \{\, \mathrm{x}_i \mid i \in \mathbb{N}_+ \,\}$ of *(formal) variables* and the finite subset $\mathrm{X}_k = \{\, \mathrm{x}_i \mid i \in [k] \,\}$. Given a ranked alphabet Σ and $V \subseteq \mathrm{X}$, the set of *Σ-trees indexed by V*, denoted by $T_\Sigma(V)$, is inductively defined to be the smallest set T such that (i) $V \subseteq T$ and (ii) for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \dots, t_k \in T$ also $\sigma(t_1, \dots, t_k) \in T$. Since we generally assume that $\Sigma \cap \mathrm{X} = \emptyset$, we write α instead of $\alpha()$ whenever $\alpha \in \Sigma_0$. Moreover, we also write T_Σ to denote $T_\Sigma(\emptyset)$.

For every $t \in T_\Sigma(\mathrm{X})$, we denote by $|t|_x$ the number of occurrences of $x \in \mathrm{X}$ in t. Given a finite $I \subseteq \mathbb{N}_+$ and family $(\, t_i \,)_{i \in I}$ of $t_i \in T_\Sigma(\mathrm{X})$, the expression $t[t_i]_{i \in I}$ denotes the result of substituting in t every x_i by t_i for every $i \in I$. If $I = [n]$, then we simply write $t[t_1, \dots, t_n]$. Let $V \subseteq \mathrm{X}$ be finite. We say that $t \in T_\Sigma(\mathrm{X})$ is *linear in V* (respectively, *nondeleting in V*), if every $x \in V$ occurs at most once (respectively, at least once) in t. The set of all Σ-trees, which are linear and nondeleting in V, is denoted by $\widehat{T_\Sigma}(V)$.

A *semiring* is an algebraic structure $\mathcal{A} = (A, +, \cdot, 0, 1)$ consisting of a commutative monoid $(A, +, 0)$ and a monoid $(A, \cdot, 1)$ such that $\cdot$ distributes over $+$ and 0 is absorbing with respect to $\cdot$. The semiring is called commutative, if $\cdot$ is commutative. As usual we use $\sum_{i \in I} a_i$ (respectively, $\prod_{i \in I} a_i$ for $I \subseteq \mathbb{N}$) for sums (respectively, products) of families $(\, a_i \,)_{i \in I}$ of $a_i \in A$ where for only finitely

many $i \in I$ we have $a_i \neq 0$ (respectively, $a_i \neq 1$). For products the order of the factors is given by the order $0 \leqslant 1 \leqslant \cdots$ on the index set I. We say that $\mathcal{A}$ is $\aleph_0$-*complete*, whenever it is possible to define an infinitary sum operation $\sum_I$ for each countable index set I (*i. e.*, $\operatorname{card}(I) \leqslant \aleph_0$) such that for every family $(a_i)_{i\in I}$ of $a_i \in A$

(i) $\sum_I (a_i)_{i\in I} = a_{j_1} + a_{j_2}$, if $I = \{j_1, j_2\}$ with $j_1 \neq j_2$,
(ii) $\sum_I (a_i)_{i\in I} = \sum_J (\sum_{I_j} (a_i)_{i\in I_j})_{j\in J}$, whenever $I = \bigcup_{j\in J} I_j$ for some countable J and $I_{j_1} \cap I_{j_2} = \emptyset$ for all $j_1 \neq j_2$, and
(iii) $\big(\sum_I (a_i)_{i\in I}\big)\big(\sum_J (b_j)_{j\in J}\big) = \sum_{I\times J} (a_i b_j)_{(i,j)\in I\times J}$ for all countable J and families $(b_j)_{j\in J}$ of $b_j \in A$.

In the sequel, we simply write the accustomed $\sum_{i\in I} a_i$ instead of the cumbersome $\sum_I (a_i)_{i\in I}$, and we implicitly assume $\sum_I$ to be given whenever we speak about an $\aleph_0$-complete semiring.

Let S be a set and $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a semiring. A *(formal) power series* φ is a mapping $\varphi\colon S \longrightarrow A$. Given $s \in S$, we denote $\varphi(s)$ also by (φ, s) and write the series as $\sum_{s\in S} (\varphi, s)\, s$. The *support* of φ is $\operatorname{supp}(\varphi) = \{\, s \in S \mid (\varphi, s) \neq 0 \,\}$. Power series with finite support are called *polynomials*, and power series with at most one support element are also called *singletons*. We denote the set of all power series by $A\langle\!\langle S \rangle\!\rangle$ and the set of polynomials by $A\langle S\rangle$. We call $\varphi \in A\langle\!\langle S\rangle\!\rangle$ *boolean*, if $(\varphi, s) = 1$ for every $s \in \operatorname{supp}(\varphi)$. The boolean singleton with empty support is denoted by $\widetilde{0}$. Power series $\varphi, \varphi' \in A\langle\!\langle S\rangle\!\rangle$ are added componentwise; *i. e.*, $(\varphi + \varphi', s) = (\varphi, s) + (\varphi', s)$ for every $s \in S$, and the power series φ is multiplied with a coefficient $a \in A$ componentwise; *i. e.*, $(a \cdot \varphi, s) = a \cdot (\varphi, s)$ for every $s \in S$.

In this paper, we consider only power series in which the set S is a set of trees. Such power series are also called *tree series*. Let Δ be a ranked alphabet. A tree series $\varphi \in A\langle\!\langle T_\Delta(\mathrm{X})\rangle\!\rangle$ is said to be *linear* (respectively, *nondeleting*) *in* $V \subseteq \mathrm{X}$, if every $t \in \operatorname{supp}(\varphi)$ is linear (respectively, nondeleting) in V. Let $\mathcal{A}$ be an $\aleph_0$-complete semiring, $\varphi \in A\langle\!\langle T_\Delta(\mathrm{X})\rangle\!\rangle$, $I \subseteq \mathbb{N}_+$ be finite, and $(\psi_i)_{i\in I}$ be a family of $\psi_i \in A\langle\!\langle T_\Delta(\mathrm{X})\rangle\!\rangle$. The *pure IO tree series substitution* (for short: IO-substitution) *(of* $(\psi_i)_{i\in I}$ *into* φ*)* [2, 10], denoted by $\varphi \longleftarrow (\psi_i)_{i\in I}$, is defined by

$$\varphi \longleftarrow (\psi_i)_{i\in I} = \sum_{\substack{t\in T_\Delta(\mathrm{X}),\\ (\forall i\in I)\colon t_i \in T_\Delta(\mathrm{X})}} (\varphi, t) \cdot \prod_{i\in I} (\psi_i, t_i)\, t[t_i]_{i\in I} \quad .$$

Let Q be an alphabet and $V \subseteq \mathrm{X}$. We write $Q(V)$ for $\{\, q(v) \mid q \in Q, v \in V \,\}$. We use the notation $|w|_x$ and the notions of linearity and nondeletion in V accordingly also for $w \in Q(\mathrm{X})^*$. Let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a semiring and Σ and Δ be ranked alphabets. A *(type I) tree representation* μ *(over* Q*,* Σ*,* Δ*, and* $\mathcal{A}$*)* [2, 9] is a family $(\mu_k(\sigma))_{k\in\mathbb{N}, \sigma\in\Sigma_k}$ of matrices $\mu_k(\sigma) \in A\langle\!\langle T_\Delta(\mathrm{X})\rangle\!\rangle^{Q\times Q(\mathrm{X}_k)^*}$ such that for every $(q, w) \in Q\times Q(\mathrm{X}_k)^*$ it holds that $\mu_k(\sigma)_{q,w} \in A\langle\!\langle T_\Delta(\mathrm{X}_{|w|})\rangle\!\rangle$, and we have $\mu_k(\sigma)_{q,w} \neq \widetilde{0}$ for only finitely many $(q, w) \in Q\times Q(\mathrm{X}_k)^*$. A tree representation μ is said to be

- *polynomial* (respectively, *boolean*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q(\mathrm{X}_k)^*$ the tree series $\mu_k(\sigma)_{q,w}$ is polynomial (respectively, boolean),
- *of type II* (respectively, *top-down*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q(\mathrm{X}_k)^*$ the tree series $\mu_k(\sigma)_{q,w}$ is linear (respectively, linear and nondeleting) in $\mathrm{X}_{|w|}$,
- *linear* (respectively, *nondeleting*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q(\mathrm{X}_k)^*$ such that $\mu_k(\sigma)_{q,w} \neq \widetilde{0}$ both $\mu_k(\sigma)_{q,w}$ is linear (respectively, nondeleting) in $\mathrm{X}_{|w|}$, and w is linear (respectively, nondeleting) in X_k,
- *bottom-up*, if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $(q, w) \in Q \times Q(\mathrm{X}_k)^*$ such that $\mu_k(\sigma)_{q,w} \neq \widetilde{0}$ we have that $w = q_1(\mathrm{x}_1) \cdots q_k(\mathrm{x}_k)$ for some $q_1, \ldots, q_k \in Q$,
- *td-deterministic*, if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $q \in Q$ there exists at most one $(w, t) \in Q(\mathrm{X}_k)^* \times T_\Delta(\mathrm{X})$ such that $t \in \mathrm{supp}(\mu_k(\sigma)_{q,w})$, and
- *bu-deterministic*, if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $w \in Q(\mathrm{X}_k)^*$ there exists at most one $(q, t) \in Q \times T_\Delta(\mathrm{X})$ such that $t \in \mathrm{supp}(\mu_k(\sigma)_{q,w})$.

Usually when we specify a tree representation μ, we just specify some entries of $\mu_k(\sigma)$ and implicitly assume the remaining entries to be $\widetilde{0}$. A *tree series transducer* [2, 9] is a sextuple $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ consisting of

- an alphabet Q of *states*,
- ranked alphabets Σ and Δ, also called *input* and *output ranked alphabet*,
- a semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$,
- a vector $F \in A\langle\!\langle \widehat{T_\Delta}(\mathrm{X}_1) \rangle\!\rangle^Q$ of *final outputs*, and
- a tree representation μ over Q, Σ, Δ, and $\mathcal{A}$.

Tree series transducers inherit the properties from their tree representation; *e. g.*, a tree series transducer with a polynomial bottom-up tree representation is called a polynomial bottom-up tree series transducer. Additionally, we say that M is a *td-homomorphism* (respectively, *bu-homomorphism*), if $Q = \{\star\}$, $F_\star = 1\,\mathrm{x}_1$, and μ is td-deterministic (respectively, bu-deterministic).

For the definition of the IO tree series transformation induced by M we need IO-substitution, and consequently, $\mathcal{A}$ should be $\aleph_0$-complete. Hence let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a tree series transducer over the $\aleph_0$-complete semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$. Then M induces a mapping $\|M\| \colon A\langle\!\langle T_\Sigma \rangle\!\rangle \longrightarrow A\langle\!\langle T_\Delta \rangle\!\rangle$ as follows. For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma$ we define the mapping $h_\mu \colon T_\Sigma \longrightarrow A\langle\!\langle T_\Delta \rangle\!\rangle^Q$ componentwise for every $q \in Q$ by

$$h_\mu\big(\sigma(t_1, \ldots, t_k)\big)_q = \sum_{\substack{w \in Q(\mathrm{X}_k)^*, \\ w = q_1(\mathrm{x}_{i_1}) \cdots q_n(\mathrm{x}_{i_n})}} \mu_k(\sigma)_{q,w} \longleftarrow \big(h_\mu(t_{i_j})_{q_j} \big)_{j \in [n]} \ .$$

Then for every $\varphi \in A\langle\!\langle T_\Sigma \rangle\!\rangle$ the *(IO) tree series transformation computed by* M is

$$\|M\|(\varphi) = \sum_{t \in T_\Sigma} (\varphi, t) \cdot \sum_{q \in Q} F_q \longleftarrow \big(h_\mu(t)_q\big) \ .$$

By $\mathrm{TOP}(\mathcal{A})$ we denote the class of tree series transformations computable by top-down tree series transducers over the semiring $\mathcal{A}$. Similarly, we use $\mathrm{p\text{–}TOP}(\mathcal{A})$

[respectively, b–TOP($\mathcal{A}$), l–TOP($\mathcal{A}$), n–TOP($\mathcal{A}$), d–TOP($\mathcal{A}$), and h–TOP($\mathcal{A}$)] for the classes of tree series transformations computable by polynomial (respectively, boolean, linear, nondeleting, td-deterministic, and td-homomorphism) top-down tree series transducers over the semiring $\mathcal{A}$. Combinations of restrictions are handled in the usual manner; *i. e.*, let x–TOP($\mathcal{A}$) and y–TOP($\mathcal{A}$) be two classes of top-down tree series transformations, then

$$xy\text{–TOP}(\mathcal{A}) = x\text{–TOP}(\mathcal{A}) \cap y\text{–TOP}(\mathcal{A}) \ .$$

The same nomenclature using the stem TOP$^{\mathrm{R}}$ (respectively, GST and BOT) is applied to type II (respectively, type I and bottom-up) tree series transducers, where for bottom-up tree series transducers the properties beginning with "td" are replaced by the corresponding ones starting with "bu". For example, hn–BOT($\mathcal{A}$) denotes the class of tree series transformations computable by nondeleting bu-homomorphism bottom-up tree series transducers over the semiring $\mathcal{A}$.

We write $\circ$ for function composition; so if $\tau_1 \colon A\langle\langle T_\Sigma \rangle\rangle \longrightarrow A\langle\langle T_\Delta \rangle\rangle$ and $\tau_2 \colon A\langle\langle T_\Delta \rangle\rangle \longrightarrow A\langle\langle T_\Gamma \rangle\rangle$ then $(\tau_1 \circ \tau_2)(\varphi) = \tau_2\big(\tau_1(\varphi)\big)$ for every $\varphi \in A\langle\langle T_\Sigma \rangle\rangle$. This composition is extended to classes of functions in the usual manner.

3 IO Tree Series Substitution

In this section we first show how to simulate a tree series transducer M of type I or II by means of the composition of a td-homomorphism top-down tree series transducer M_1 and a bottom-up tree series transducer M_2. Thereby we obtain a limitation of the power of tree series transducers of types I and II. The idea of the construction is to simply create sufficiently many copies of subtrees of the input tree by M_1. Then multiple visits of M to one input subtree such as, for example, $q(\mathrm{x}_1)$ and $p(\mathrm{x}_1)$ can be simulated by $q(\mathrm{x}_1)$ and $p(\mathrm{x}_6)$ where x_6 refers to a copy of x_1 created by M_1. More precisely, we first compute the maximal number of visits to one subtree spawned by one rule application. Let mx be that number. We create a new output alphabet from the input alphabet Σ of M by keeping the symbols of Σ but changing their rank to mx-times their rank in Σ. Reading $\sigma(t_1, \dots, t_k)$ in the input, M_1 simply outputs $\sigma(u_1, \dots, u_1, \dots, u_k, \dots, u_k)$ where u_i is the translation of t_i for every $i \in [k]$. Then we can simulate M without visiting input subtrees twice because enough copies are available. Altogether this yields that at each node of the output tree of M_1 each direct input subtree is visited at most once and such a tree series transducer can be simulated by a bottom-up tree series transducer M_2.

In the sequel, we use the notation $[y]$ where y is one of the abbreviations of restrictions (*i. e.*, $y \in \{\mathrm{p}, \mathrm{b}, \mathrm{l}, \mathrm{n}, \mathrm{d}, \mathrm{h}\}$) in equalities and inequalities to mean that this restriction is optional; *i. e.*, throughout the statement $[y]$ can be substituted by the empty word or by y. For example, [d]–TOP($\mathcal{A}$) $\subseteq$ [d]–TOP$^{\mathrm{R}}$($\mathcal{A}$) states that each tree series transformation computable by top-down (respectively, td-deterministic top-down) tree series transducers is also computable by tree series

transducers of type II (respectively, td-deterministic tree series transducers of type II).

Lemma 1 (Decomposition). *Let $\mathcal{A}$ be a commutative, $\aleph_0$-complete semiring.*

$$\text{[p][b][l]–GST}(\mathcal{A}) \subseteq \text{[l]bhn–TOP}(\mathcal{A}) \circ \text{[p][b][l]–BOT}(\mathcal{A}) \tag{1}$$

$$\text{[p][b][l]–TOP}^{\mathrm{R}}(\mathcal{A}) \subseteq \text{[l]bhn–TOP}(\mathcal{A}) \circ \text{[p][b]l–BOT}(\mathcal{A}) \tag{2}$$

Proof. Let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a tree series transducer. We construct a td-homomorphism top-down tree series transducer M_1 and a bottom-up tree series transducer M_2 such that $\|M\| = \|M_1\| \circ \|M_2\|$. Let $w \in Q(\mathrm{X})^*$. Recall that by $|w|_x$ we denote the number of occurrences of $x \in \mathrm{X}$ in w. Let

$$\mathrm{mx} = \max\bigl(\{1\} \cup \{\, |w|_{\mathrm{x}_j} \mid k, j \in \mathbb{N}, \sigma \in \Sigma_k, (q, w) \in Q \times Q(\mathrm{X})^*, \mu_k(\sigma)_{q,w} \neq \widetilde{0} \,\}\bigr)$$

and for every $k \in \mathbb{N}$ we let $\Gamma_{k\cdot\mathrm{mx}} = \Sigma_k$ and $\Gamma_n = \emptyset$ for every $n \in \mathbb{N}$ that is not a multiple of mx. We note that $\mathrm{mx} = 1$ if M is linear. We construct $M_1 = (\{\star\}, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ with $(F_1)_\star = 1\,\mathrm{x}_1$ and for every $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$

$$(\mu_1)_k(\sigma)_{\star,\underbrace{\star(\mathrm{x}_1)\cdots\star(\mathrm{x}_1)}_{\text{mx times}}\cdots\underbrace{\star(\mathrm{x}_k)\cdots\star(\mathrm{x}_k)}_{\text{mx times}}} = 1\,\sigma(\mathrm{x}_1, \ldots, \mathrm{x}_{k\cdot\mathrm{mx}}) \ .$$

Clearly, M_1 is a boolean, nondeleting, homomorphism tree series transducer, which is linear whenever M is so. In this case M_1 just computes the identity.

Let $\perp \notin Q$ be a new state, $Q' = Q \cup \{\perp\}$, and $d \in \mathbb{N}$ be the maximal integer such that $\Sigma_d \neq \emptyset$. For every $n \in [d]$ let $I_n \in \mathbb{N}^{[d]}$, where $\mathbb{N}^{[d]}$ is the set of all mappings from $[d]$ to $\mathbb{N}$ (alternatively a vector with d entries of $\mathbb{N}$), be $I_n(n') = 0$ for every $n' \in [d] \setminus \{n\}$ and $I_n(n) = 1$. Moreover, let $I = \sum_{i\in[d]} I_i$. For every $k \in \mathbb{N}$ we define $\mathrm{ren}_k \colon Q(\mathrm{X})^* \times \mathbb{N}^{[d]} \longrightarrow Q'(\mathrm{X})^*$ for every $f \in \mathbb{N}^{[d]}$ inductively on $Q(\mathrm{X})^*$ by

$$\begin{aligned}\mathrm{ren}_k(\varepsilon, f) = {} & \perp(\mathrm{x}_{f(1)}) \cdots \perp(\mathrm{x}_{\mathrm{mx}})\perp(\mathrm{x}_{\mathrm{mx}+f(2)}) \cdots \perp(\mathrm{x}_{2\cdot\mathrm{mx}}) \\ & \cdots \\ & \perp(\mathrm{x}_{(k-2)\cdot\mathrm{mx}+f(k-1)}) \cdots \perp(\mathrm{x}_{(k-1)\cdot\mathrm{mx}})\perp(\mathrm{x}_{(k-1)\cdot\mathrm{mx}+f(k)}) \cdots \perp(\mathrm{x}_{k\cdot\mathrm{mx}})\end{aligned}$$

and for every $q \in Q$, $i \in [d]$, $w \in Q(\mathrm{X}_d)^*$ by

$$\mathrm{ren}_k(q(\mathrm{x}_i)\cdot w, f) = q(\mathrm{x}_{(i-1)\cdot\mathrm{mx}+f(i)})\cdot \mathrm{ren}_k(w, f + I_i) \ .$$

Secondly, let $M_2' = (Q', \Gamma, \Delta, \mathcal{A}, F_2, \mu_2')$ with $(F_2)_q = F_q$ for every $q \in Q$ and $(F_2)_\perp = \widetilde{0}$ and $(\mu_2')_{k\cdot\mathrm{mx}}(\sigma)_{q,\mathrm{ren}_k(w,I)} = \mu_k(\sigma)_{q,w}$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q(\mathrm{X}_k)^*$. Finally, let $\alpha \in \Sigma_0$ be arbitrary and

$$(\mu_2')_{k\cdot\mathrm{mx}}(\sigma)_{\perp,\perp(\mathrm{x}_1)\cdots\perp(\mathrm{x}_{k\cdot\mathrm{mx}})} = 1\,\alpha \ .$$

Note that M_2' need not be bottom-up because there may be $(\mu_2')_k(\sigma)_{q,w} \neq \widetilde{0}$ where w is of the form $w_1\cdot q_1(\mathrm{x}_{j_1})q_2(\mathrm{x}_{j_2})\cdot w_2$ with $j_1 > j_2$; *i. e.*, the variables

in w do not occur in the order $\mathrm{x}_1, \dots, \mathrm{x}_k$. By a straightforward reordering of the symbols $q_i(\mathrm{x}_j)$ in w and a corresponding substitution of variables in $(\mu_2')_k(\sigma)_{q,w}$, we can, however, turn M_2' into a bottom-up tree series transducer M_2.

We furthermore note that if M is of type II, then M_2' is actually a linear tree series transducer of type II, and consequently, M_2 is linear as well. Finally, it is also obvious that M_2 is polynomial (respectively, boolean, linear), whenever M is so. Clearly, the homomorphism property (of M) is not preserved because we added the extra state $\bot$. □

Now let us investigate the opposite direction; *i. e.*, the composition of a td-homomorphism top-down tree series transducer M_1 and a bottom-up tree series transducer M_2. The idea of the construction is quite straightforward; we translate the output of M_1 with the help of M_2. Therefore, we need to generalize the mapping h_{μ_2} to trees with variables. Roughly speaking, we supply h_{μ_2} with a mapping that assigns a state to each variable. A successful computation may commence at a variable only in the state assigned to the variable. So let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a tree series transducer over an $\aleph_0$-complete semiring $\mathcal{A}$, and let $V \subseteq \mathbb{N}_+$. For every $\bar{q} \in Q^V$ we define the mapping $h_\mu^{\bar{q}} \colon T_\Sigma(\mathrm{X}) \longrightarrow A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle^Q$ as follows.

- For every $j \in \mathbb{N}_+$ and $q \in Q$

$$h_\mu^{\bar{q}}(\mathrm{x}_j)_q = \begin{cases} \widetilde{0} & \text{if } j \in V, \bar{q}_j \neq q \ , \\ 1\,\mathrm{x}_j & \text{otherwise} \ . \end{cases}$$

- For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $t_1, \dots, t_k \in T_\Sigma(\mathrm{X})$, and $q \in Q$

$$h_\mu^{\bar{q}}\big(\sigma(t_1, \dots, t_k)\big)_q = \sum_{\substack{w \in Q(\mathrm{X}_k)^*, \\ w = q_1(\mathrm{x}_{i_1}) \cdots q_n(\mathrm{x}_{i_n})}} \mu_k(\sigma)_{q,w} \longleftarrow \big(h_\mu(t_{i_j})_{q_j} \big)_{j \in [n]} \ .$$

We just write $\bar{q}_1 \cdots \bar{q}_n$ for $\bar{q}$ whenever $V = [n]$ for some $n \in \mathbb{N}$.

Lemma 2 (Composition). *Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring.*

$$\text{[l]h–TOP}(\mathcal{A}) \circ \text{[p][l][h]–BOT}(\mathcal{A}) \subseteq \text{[p][l][h]–GST}(\mathcal{A}) \tag{3}$$

$$\text{[l]h–TOP}(\mathcal{A}) \circ \text{[p][h]l–BOT}(\mathcal{A}) \subseteq \text{[p][l][h]–TOP}^{\mathrm{R}}(\mathcal{A}) \tag{4}$$

$$\text{[l]h–TOP}(\mathcal{A}) \circ \text{[p][h]nl–BOT}(\mathcal{A}) \subseteq \text{[p][l][h]–TOP}(\mathcal{A}) \tag{5}$$

Proof. Let $M_1 = (\{\star\}, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ be a homomorphism top-down tree series transducer and $M_2 = (Q, \Gamma, \Delta, \mathcal{A}, F, \mu_2)$ be a bottom-up tree series transducer. We construct a tree series transducer $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ as follows. For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w = q_1(\mathrm{x}_{i_1}) \cdots q_n(\mathrm{x}_{i_n}) \in Q(\mathrm{X}_k)^*$ let

$$\mu_k(\sigma)_{q,w} = h_{\mu_2}^{q_1 \cdots q_n}\big((\mu_1)_k(\sigma)_{\star, \star(\mathrm{x}_{i_1}) \cdots \star(\mathrm{x}_{i_n})}\big)_q \ .$$

By the definition of top-down tree representations, $(\mu_1)_k(\sigma)_{\star,\star(\mathrm{x}_{i_1})\cdots\star(\mathrm{x}_{i_n})}$ is non-deleting and linear in X_n. Whenever M_2 is linear (respectively, nondeleting and linear), then M will be of type II (respectively, top-down). The proof of preservation of the additional properties is left to the reader. □

Putting Lemmata 1 and 2 together, we obtain the following characterization of the power of tree series transducers of type I and II.

Theorem 3. *Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring.*

$$[\mathrm{p}][\mathrm{l}]\text{–}\mathrm{GST}(\mathcal{A}) = [\mathrm{l}]\mathrm{bh}\text{–}\mathrm{TOP}(\mathcal{A}) \circ [\mathrm{p}][\mathrm{l}]\text{–}\mathrm{BOT}(\mathcal{A}) \tag{6}$$

$$[\mathrm{p}][\mathrm{l}]\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A}) = [\mathrm{l}]\mathrm{bh}\text{–}\mathrm{TOP}(\mathcal{A}) \circ [\mathrm{p}]\mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A}) \tag{7}$$

Proof. The statements follow directly from Lemmata 1 and 2. □

Finally, we turn to the question concerning linear tree series transformations of type II. So far, we have seen that $\mathrm{l}\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A}) = \mathrm{lbh}\text{–}\mathrm{TOP}(\mathcal{A}) \circ \mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A})$ and hence $\mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A}) \subseteq \mathrm{l}\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A})$. The converse [*i. e.*, $\mathrm{l}\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A}) \subseteq \mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A})$] can be seen from our remark in the proof of Lemma 1. There we noted that if the input transducer M is linear, then the first transducer M_1 of the composition just computes the identity; thus $\mathrm{l}\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A}) \subseteq \mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A})$. Hence we derived the following theorem.

Theorem 4. *Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring.*

$$\mathrm{l}\text{–}\mathrm{TOP}^{\mathrm{R}}(\mathcal{A}) = \mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A}) \tag{8}$$

The inclusions are displayed graphically for commutative and $\aleph_0$-complete semirings $\mathcal{A}$ in Fig. 1, where all line segments are directed upwards, so that, *e. g.*, $\mathrm{l}\text{–}\mathrm{TOP}(\mathcal{A}) \subseteq \mathrm{l}\text{–}\mathrm{BOT}(\mathcal{A})$. However, none of the inclusions needs to be strict.

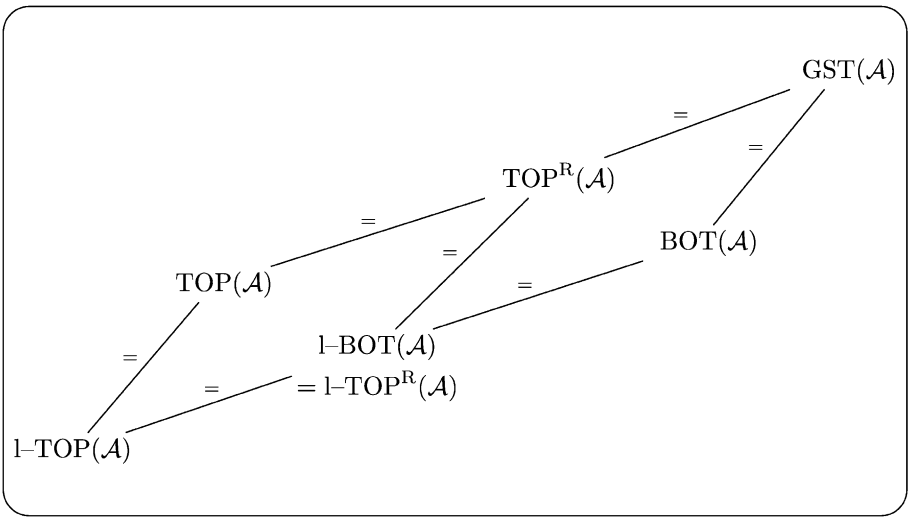


Fig. 1. Hierarchy of IO tree series transformations

4 OI Tree Series Substitution

Throughout the survey [9] the used tree series substitution is OI tree series substitution [1, 11]. It is clear that as long as the tree representation of a tree series transducer is linear and nondeleting (*i. e.*, the tree series transducer is top-down), the use of OI tree series substitution instead of pure IO tree series substitution does not yield different results. If these conditions are, however, not required (*e. g.*, for tree series transducers of types I and II), the results may diverge, and this section examines the ramifications of using OI tree series substitution. Thereby we answer Question 2 as originally posed in [9].

Let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be an $\aleph_0$-complete semiring, $k \in \mathbb{N}$, $\delta \in \Delta_k$ be a symbol of a ranked alphabet Δ, and $\psi_1, \dots, \psi_k \in A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle$. We define

$$\delta(\psi_1, \dots, \psi_k) = \sum_{t_1,\dots,t_k \in T_\Delta(\mathrm{X})} (\psi_1, t_1) \cdots (\psi_k, t_k)\, \delta(t_1, \dots, t_k) \ .$$

Let $I \subseteq \mathbb{N}_+$ be finite, $\varphi \in A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle$, and $(\psi_i)_{i\in I}$ be a family of tree series $\psi_i \in A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle$. The *OI tree series substitution (of* $(\psi_i)_{i\in I}$ *into* φ*)* [1, 11] (for short: OI-substitution), denoted by $\varphi[\psi_i]_{i\in I}$, is inductively defined as follows.

- For every $j \in \mathbb{N}_+$
$$\mathrm{x}_j[\psi_i]_{i\in I} = \begin{cases} \psi_j & \text{if } j \in I \ , \\ 1\,\mathrm{x}_j & \text{otherwise} \ . \end{cases}$$
- For every $k \in \mathbb{N}$, $\delta \in \Delta_k$, and $t_1, \dots, t_k \in T_\Delta(\mathrm{X})$
$$\delta(t_1, \dots, t_k)[\psi_i]_{i\in I} = \delta(t_1[\psi_i]_{i\in I}, \dots, t_k[\psi_i]_{i\in I}) \ .$$

Finally, $\varphi[\psi_i]_{i\in I} = \sum_{t\in T_\Sigma(\mathrm{X})} (\varphi, t) \cdot t[\psi_i]_{i\in I}$. We write $\varphi[\psi_1, \dots, \psi_n]$ for $\varphi[\psi_i]_{i\in[n]}$.

The semantics of tree series transducers using OI-substitution is defined next. Let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a tree series transducer. We define the mapping $h^{\mathrm{OI}}_\mu \colon T_\Sigma \longrightarrow A\langle\langle T_\Delta\rangle\rangle^Q$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \dots, t_k \in T_\Sigma$ componentwise for every $q \in Q$ by

$$h^{\mathrm{OI}}_\mu\big(\sigma(t_1, \dots, t_k)\big)_q = \sum_{\substack{w\in Q(\mathrm{X}_k)^*,\\ w=q_1(\mathrm{x}_{i_1})\cdots q_n(\mathrm{x}_{i_n})}} \mu_k(\sigma)_{q,w} \big[h^{\mathrm{OI}}_\mu(t_{i_j})_{q_j}\big]_{j\in[n]} \ .$$

The *OI tree series transformation computed by* M, denoted by $\|M\|^{\mathrm{OI}}$, is then defined for every $\varphi \in A\langle\langle T_\Sigma\rangle\rangle$ by

$$\|M\|^{\mathrm{OI}}(\varphi) = \sum_{t\in T_\Sigma} (\varphi, t) \cdot \sum_{q\in Q} F_q\big[h^{\mathrm{OI}}_\mu(t)_q\big] \ .$$

We denote the class of OI tree series transformations computable by a class of tree series transducers by the OI-subscripted denotation of the class of tree series transformations computable by the same class of transducers using IO-substitution. For example, $\mathrm{p\text{–}TOP}^{\mathrm{R}}_{\mathrm{OI}}(\mathcal{A})$ denotes the class of all OI tree series

transformations computable by polynomial tree series transducers of type II (over the semiring $\mathcal{A}$).

Firstly, we show that tree series transducers of type II and top-down tree series transducers are equally powerful supposed that OI-substitution is used. Roughly speaking, this is due to the fact that the tree series to be substituted for a deleted variable has absolutely no influence on the result of the substitution; *i. e.*, $\varphi[\psi_i]_{i\in I} = \varphi[\psi_i]_{i\in \mathrm{var}(\varphi)}$ where $\mathrm{var}(\varphi) = \bigcup_{t\in \mathrm{supp}(\varphi)} \mathrm{var}(t)$.

Lemma 5. *For every $\aleph_0$-complete semiring $\mathcal{A}$*

$$\text{[p][b][l][n][d][h]–}\mathrm{TOP}^{\mathrm{R}}_{\mathrm{OI}}(\mathcal{A}) = \text{[p][b][l][n][d][h]–}\mathrm{TOP}_{\mathrm{OI}}(\mathcal{A}) \ . \tag{9}$$

Proof. Clearly, each top-down tree series transducer is also of type II, so it just remains to prove $y\text{–}\mathrm{TOP}^{\mathrm{R}}_{\mathrm{OI}}(\mathcal{A}) \subseteq y\text{–}\mathrm{TOP}_{\mathrm{OI}}(\mathcal{A})$. Let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a tree series transducer and $j \in \mathbb{N}_+$ be the maximal integer such that there exist $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, $w \in Q(\mathrm{X})^*$, and $t \in \mathrm{supp}(\mu_k(\sigma)_{q,w})$ such that $j \leqslant |w|$ and $|t|_{\mathrm{x}_j} = 0$. We construct a tree series transducer $M' = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu')$ such that $\|M\| = \|M'\|$ and for every $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$ all tree series in the range of $\mu'_k(\sigma)$ will be nondeleting in $\mathrm{X}_k \cap (\mathrm{X} \setminus \mathrm{X}_{j-1})$. Clearly, since $\max\{\, |w| \mid k \in \mathbb{N}, \sigma \in \Sigma_k, \mu_k(\sigma)_{q,w} \neq \widetilde{0}\,\}$ is finite, iteration of this construction yields the desired result.

For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w = q_1(\mathrm{x}_{i_1}) \cdots q_n(\mathrm{x}_{i_n}) \in Q(\mathrm{X}_k)^*$, if $j > n$ then we let $\mu'_k(\sigma)_{q,w} = \mu_k(\sigma)_{q,w}$ and otherwise we set

$$\mu'_k(\sigma)_{q,w} = \sum_{\substack{t\in T_\Delta(\mathrm{X}),\\ |t|_{\mathrm{x}_j}\geqslant 1}} (\mu_k(\sigma)_{q,w}, t)\, t + \\ + \sum_{\substack{w'\in Q(\mathrm{X}_k)^{n+1},\\ w=w'_1\cdots w'_{j-1}w'_{j+1}\cdots w'_{n+1},\\ t\in T_\Delta(\mathrm{X}\setminus\{\mathrm{x}_j\})}} (\mu_k(\sigma)_{q,w'}, t)\, t[\mathrm{x}_1, \ldots, \mathrm{x}_j, \mathrm{x}_j, \ldots, \mathrm{x}_n] \ .$$

Clearly, $\mu'_k(\sigma)_{q,w}$ is nondeleting in $X_k \cap (\mathrm{X} \setminus \mathrm{X}_{j-1})$ as is every other tree series in the range of $\mu'_k(\sigma)$ for arbitrary $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$. Moreover, $\|M\| = \|M'\|$ because for every $\varphi, \varphi' \in A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle$ and family $(\psi_i)_{i\in I}$ of $\psi_i \in A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle$ we have that $(\varphi + \varphi')[\psi_i]_{i\in I} = \varphi[\psi_i]_{i\in I} + \varphi'[\psi_i]_{i\in I}$ and $\varphi[\psi_i]_{i\in I} = \varphi[\psi_i]_{i\in I\setminus\{j\}}$, whenever $j \notin \mathrm{var}(\varphi)$. □

Similarly, we can show that nonlinearity can be resolved by naming multiple occurrences of the same variable apart. Let $\mathrm{ren}\colon T_\Delta(\mathrm{X}) \times \mathbb{N}_+ \times \mathbb{N}_+ \longrightarrow T_\Delta(\mathrm{X})$ be the mapping such that $\mathrm{ren}(t, j, n)$ is the tree obtained by renaming the first occurrence (with respect to a depth-first left-to-right traversal of t) of x_j in t to x_j, the second occurrence of x_j to x_n, the third occurrence of x_j to x_{n+1}, and so on. Then roughly speaking, the construction is based on the observation $t[\psi_1, \ldots, \psi_k] = \mathrm{ren}(t, j, k+1)[\psi_1, \ldots, \psi_k, \psi_j, \ldots, \psi_j]$ for every $t \in T_\Delta(\mathrm{X}_k)$ and $j \in [k]$.

Lemma 6. *For every $\aleph_0$-complete semiring $\mathcal{A}$*

$$\text{[b][l][n][d][h]p–TOP}^{\mathrm{R}}_{\mathrm{OI}}(\mathcal{A}) = \text{[b][l][n][d][h]p–GST}_{\mathrm{OI}}(\mathcal{A}) \ . \tag{10}$$

Proof. Let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a polynomial tree series transducer and $j \in \mathbb{N}_+$ be the maximal integer such that there exist $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, $w \in Q(\mathrm{X})^*$, and $t \in \operatorname{supp}(\mu_k(\sigma)_{q,w})$ such that $j \leqslant |w|$ and $|t|_{\mathrm{x}_j} > 1$. We construct a tree series transducer $M' = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu')$ such that $\|M\| = \|M'\|$ and for every $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$ all tree series in the range of $\mu'_k(\sigma)$ will be linear in $\mathrm{X}_k \cap (\mathrm{X} \setminus \mathrm{X}_{j-1})$. Clearly, since $\max\{\, |w| \mid k \in \mathbb{N}, \sigma \in \Sigma_k, \mu_k(\sigma)_{q,w} \neq \widetilde{0} \,\}$ is finite, iteration of this construction yields the desired result.

For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w = q_1(\mathrm{x}_{i_1}) \cdots q_n(\mathrm{x}_{i_n}) \in Q(\mathrm{X}_k)^*$, if $j > n$ then we let $\mu'_k(\sigma)_{q,w} = \mu_k(\sigma)_{q,w}$, and otherwise we set

$$\begin{aligned}\mu'_k(\sigma)_{q,w} = &\sum_{\substack{t \in T_\Delta(\mathrm{X}),\\ |t|_{\mathrm{x}_j} \leqslant 1}} (\mu_k(\sigma)_{q,w}, t)\, t + \\ &+ \sum_{\substack{w' \in Q(\mathrm{X}_k)^*,\\ t \in \operatorname{supp}(\mu_k(\sigma)_{q,w'}), |t|_{\mathrm{x}_j} > 1,\\ w'' = (w'_j)^{|t|_{\mathrm{x}_j} - 1}, w = w'w''}} (\mu_k(\sigma)_{q,w'}, t)\ \operatorname{ren}(t, j, |w'| + 1) \ ,\end{aligned}$$

Due to the fact that M is polynomial, μ' is a tree representation. Moreover, $\mu'_k(\sigma)_{q,w}$ is linear in $X_k \cap (\mathrm{X} \setminus \mathrm{X}_{j-1})$ as is every other tree series in the range of $\mu'_k(\sigma)$ for arbitrary $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$. Moreover, $\|M\| = \|M'\|$ because for every $n \in \mathbb{N}$, $i \in [n]$, $t \in T_\Delta(\mathrm{X}_n)$, and $\psi_1, \ldots, \psi_n \in A\langle T_\Delta(\mathrm{X})\rangle$ we have that

$$t[\psi_1, \ldots, \psi_n] = \operatorname{ren}(t, i, n+1)[\psi_1, \ldots, \psi_n, \underbrace{\psi_i, \ldots, \psi_i}_{|t|_{\mathrm{x}_i} - 1}] \ .$$

The proof proceeds along the lines of the one of Lemma 5 and is therefore omitted. □

The proof of the previous result breaks down whenever M is not polynomial. However, if there exists a constant $n \in \mathbb{N}$ such that for every $j \in \mathbb{N}_+$, $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, $w \in Q(\mathrm{X}_k)^*$, and $t \in \operatorname{supp}(\mu_k(\sigma)_{q,w})$ we have that $|t|_{\mathrm{x}_j} \leqslant n$, then the result holds and can be proved in essentially the same manner.

Theorem 7. *For every $\aleph_0$-complete semiring $\mathcal{A}$*

$$\begin{aligned}&\text{[b][l][n][d][h]p–TOP}(\mathcal{A}) = \text{[b][l][n][d][h]p–TOP}_{\mathrm{OI}}(\mathcal{A}) \\ &= \text{[b][l][n][d][h]p–TOP}^{\mathrm{R}}_{\mathrm{OI}}(\mathcal{A}) = \text{[b][l][n][d][h]p–GST}_{\mathrm{OI}}(\mathcal{A}) \ .\end{aligned} \tag{11}$$

Proof. The theorem is an immediate consequence of Lemmata 5 and 6. □

Hence Question 2 of [9] can be answered by stating that polynomial tree series transducers of type I, polynomial tree series transducers of type II, and polynomial top-down tree series transducers are all equally powerful with respect to OI-substitution.

Acknowledgements

The author greatly appreciates the suggestions of the referees. Their comments enabled the author to correct and clarify important points. One referee raised the question of effectiveness of the constructions when the involved tree representations are not polynomial. Clearly, this question is interesting but the answer would largely depend on the finitary presentation of such tree representations. Thus the author would like to leave this as an open problem.

References

1. Kuich, W.: Tree transducers and formal tree series. Acta Cybernet. **14** (1999) 135–149
2. Engelfriet, J., Fülöp, Z., Vogler, H.: Bottom-up and top-down tree series transformations. J. Autom. Lang. Comb. **7** (2002) 11–70
3. Fülöp, Z., Vogler, H.: Tree series transformations that respect copying. Theory Comput. Syst. **36** (2003) 247–293
4. Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theoret. Comput. Sci. **18** (1982) 115–148
5. Kuich, W.: Formal power series over trees. In Bozapalidis, S., ed.: Proc. 3rd Int. Conf. Developments in Language Theory, Aristotle University of Thessaloniki (1997) 61–101
6. Borchardt, B., Vogler, H.: Determinization of finite state weighted tree automata. J. Autom. Lang. Comb. **8** (2003) 417–463
7. Kuich, W.: Full abstract families of tree series I. In Karhumäki, J., Maurer, H.A., Paun, G., Rozenberg, G., eds.: Jewels Are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa. Springer (1999) 145–156
8. Engelfriet, J.: Bottom-up and top-down tree transformations – a comparison. Math. Systems Theory **9** (1975) 198–231
9. Ésik, Z., Kuich, W.: Formal tree series. J. Autom. Lang. Comb. **8** (2003) 219–285
10. Bozapalidis, S.: Context-free series on trees. Inform. and Comput. **169** (2001) 186–229
11. Bozapalidis, S.: Equational elements in additive algebras. Theory Comput. Syst. **32** (1999) 1–33

The Inclusion Problem for Unambiguous Rational Trace Languages*

Paolo Massazza

Dipartimento di Informatica e Comunicazione
Università degli Studi dell'Insubria,
Via Mazzini 5, 21100 Varese, Italia
paolo.massazza@uninsubria.it

Abstract. Given a class $\mathcal{C}$ of languages, the Inclusion Problem for $\mathcal{C}$ consists of deciding whether for $L_1, L_2 \in \mathcal{C}$ we have $L_1 \subseteq L_2$.
In this work we prove that the Inclusion Problem is decidable for the class of unambiguous rational trace languages that are subsets of the monoid $(((a_1^\star \cdot b_1^\star) \times c_1^\star) \cdot ((a_2^\star \cdot b_2^\star) \times c_2^\star)) \times c_3^\star$.

1 Introduction

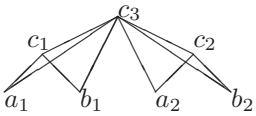
Trace languages have been introduced by Mazurkiewicz [15] as a model of the behaviour of concurrent processes. More precisely, trace languages are subsets of free partially commutative monoids for which several decision problems have been studied. In particular, given a class $\mathcal{C}$ of trace languages the Inclusion Problem for $\mathcal{C}$ consists of deciding whether for $L_1, L_2 \in \mathcal{C}$ we have $L_1 \subseteq L_2$. The Equivalence Problem for $\mathcal{C}$ ($L_1 = L_2$?) can be defined similarly; moreover, it is immediate to see that, for every class $\mathcal{C}$, Equivalence for $\mathcal{C}$ is reducible to Inclusion for $\mathcal{C}$ since $L_1 = L_2$ iff $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$. Therefore, if Inclusion is decidable then Equivalence is decidable, and if a class $\mathcal{C}$ is closed under union (or intersection) then both problems are either decidable or undecidable since $L_1 \subseteq L_2$ if and only if $L_1 \cup L_2 = L_2$ ($L_1 \cap L_2 = L_1$).

In this paper we deal with the class of unambiguous rational trace languages $\mathrm{Rat_U}(\Sigma, C)$, a particular subclass of the class of rational trace languages $\mathrm{Rat}(\Sigma, C)$ that has been widely studied and for which many results are known. In particular, using a technique due to Ibarra ([12]), in [1], [11] it is shown that Equivalence for $\mathrm{Rat}(\Sigma, C)$ is undecidable when $\Sigma = \{a, b, c\}$ and $C = {}_a\overset{b}{\wedge}_c$. On the other hand, Inclusion turns out to be decidable when C is transitive ([5]). We also recall that Equivalence for $\mathrm{Rat_U}(\Sigma, C)$ is decidable for any commutation relation C ([19]), while, for the same class, Inclusion is undecidable if C is the relation $\begin{smallmatrix} & a & \\ d & & b \\ & c & \end{smallmatrix}$ ([4]). Last but not least, in [7] it is shown that Inclusion is decidable for the class $\mathrm{Rat_{Fin}}(\Sigma, C)$ of finitely ambiguous rational trace languages over an alphabet $\Sigma = A \uplus B$ with $C = (A \times \Sigma \cup \Sigma \times A) \setminus I$.

* Partially supported by the Project M.I.U.R. COFIN 2003-2005 "Formal languages and automata: methods, models and applications"

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 350–361, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The main result we present here is that Inclusion is decidable for the class $\mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$ with alphabet $\Sigma = \{a_1, b_1, c_1, a_2, b_2, c_2, c_3\}$ and commutation relation C given by



We follow a technique similar to that used in [8] to show that Equivalence is decidable for a class $\mathcal{C}$ of recursive languages that is c-holonomic and c-closed under intersection (i.e. the elements of $\mathcal{C}$ admit holonomic generating functions, have finite computable specifications, their intersection is in $\mathcal{C}$ and has a specification computed in a finite time). More precisely, given two trace languages $L_1, L_2 \in \mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$, we reduce the problem of deciding whether $L_1 \subseteq L_2$ to the problem of verifying an equality between generating functions:

$$\phi_{L_1}(z) = \phi_{L_1 \cap L_2}(z) \ .$$

By showing that these generating functions are holonomic, it turns out that Inclusion for $\mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$ is reduced to Equivalence for holonomic functions, a problem that is well known to be decidable (see, for instance, [20]).

2 Preliminaries

In this section we recall some basic definitions and results about trace languages and formal series.

2.1 Monoids and Languages

Let M_1, M_2 be two submonoids of a monoid M. The free product of M_1 and M_2 is defined as the monoid $M_1 \cdot M_2 = \{m \in M \mid m = m_1 m_1' m_2 m_2' \cdots m_k m_k', m_i \in M_1, m_i' \in M_2\}$. The direct product of monoids is denoted by $\times$ and defined as $M_1 \times M_2 = \{(m_1, m_2) \mid m_1 \in M_1, m_2 \in M_2\}$. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a finite alphabet and let $\Sigma^\star$ be the free monoid generated by Σ, that is, the monoid $\sigma_1^\star \cdot \sigma_2^\star \cdot \ldots \cdot \sigma_n^\star$. The elements of $\Sigma^\star$ are called words. If $w = \sigma_{i_1} \cdots \sigma_{i_n} \in \Sigma^\star$ its length is $|w| = n$. A language is simply a subset of $\Sigma^\star$.
A commutation relation on Σ is an irreflexive and symmetric relation $C \subseteq \Sigma \times \Sigma$. We denote by $F(\Sigma, C)$ the free partially commutative (f.p.c.) monoid $\Sigma^\star / \rho_C$ where $\rho_C \subseteq \Sigma^\star \times \Sigma^\star$ is the congruence generated by C. We call *trace* an element of a f.p.c. monoid. A trace can be interpreted as an equivalence set of words: given a string $w \in \Sigma^\star$ we denote by $[w]_{\rho_C}$ the equivalence class of w (i.e. the trace generated by w). A *trace language* is a subset of $F(\Sigma, C)$; given a language $L \subseteq \Sigma^\star$ and a commutation relation C, the trace language generated by L is

$$[L]_{\rho_C} = \{[w]_{\rho_C} \mid w \in L\} \subseteq F(\Sigma, C) \ .$$

When $C = \Sigma \times \Sigma$, we denote by Σ^c the free commutative monoid generated by Σ. An element $\underline{\sigma^a} = \sigma_1^{a_1} \cdots \sigma_n^{a_n}$ of Σ^c is called monomial and the product of monomials becomes $\underline{\sigma^a} \cdot \underline{\sigma^b} = \sigma_1^{a_1+b_1} \cdots \sigma_n^{a_n+b_n}$.

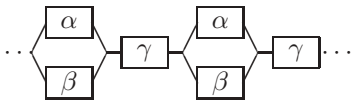
We define here a class $\mathcal{M}$ of monoids that is of particular interest for the cases we consider later on.

Definition 1. *$\mathcal{M}$ is the smallest class of monoids containing the monoids $\sigma^\star$ ($\sigma \in \Sigma$) and closed under $\cdot$ and $\times$.*

It is easily shown that all the elements of $\mathcal{M}$ are f.p.c. monoids and that there exist f.p.c. monoids that are not in $\mathcal{M}$ as, for example, the f.p.c. monoid defined by the commutation relation

The elements of a monoid $m \in \mathcal{M}$ are called *serial/parallel* traces since they can be obtained as serial/parallel composition of basic components belonging to free noncommutative monoids.

Example 1. Let $\Sigma = \{a, b, c, d, e\}$. An element of the monoid $((a^\star \cdot d^\star) \times (b^\star \cdot c^\star)) \cdot e^\star$ has a structure of type



where α belongs to the monoid $a^\star \cdot d^\star$, β belongs to $b^\star \cdot c^\star$ and $\gamma \in e^\star$.

2.2 Formal Series and Rational Languages

Let $\mathbb{Q}$ be the field of rational numbers. A *formal series* ψ on a monoid M with coefficients in $\mathbb{Q}$ is a function $\psi : M \mapsto \mathbb{Q}$. We denote by (ψ, w) the coefficient in $\mathbb{Q}$ associated with w by ψ, and we encode ψ by the formal sum $\sum_{w \in M} (\psi, w)w$. The *support* of a series ψ is the set $\mathrm{Supp}(\psi) = \{m \in M \mid (\psi, m) \neq 0\}$. The sum and the Cauchy product of formal series are defined respectively as

$$(\phi + \psi, w) = (\phi, w) + (\psi, w), \qquad (\phi \cdot \psi, w) = \sum_{xy=w} (\phi, x) \cdot (\psi, y) \ .$$

We also consider the Hadamard product of two series, defined as

$$(\phi \odot \psi, w) = (\phi, w) \cdot (\psi, w) \ .$$

Let Σ be a finite alphabet, we denote by $\mathbb{Q}\langle\langle \Sigma \rangle\rangle$ the ring of formal series on the free monoid $\Sigma^\star$ having values in $\mathbb{Q}$. We also indicate by $\mathbb{Q}\langle \Sigma \rangle$ the ring of polynomials, that is, the ring of series in $\mathbb{Q}\langle\langle \Sigma \rangle\rangle$ having finite support.

Definition 2. *A series $\phi \in \mathbb{Q}\langle\langle \Sigma \rangle\rangle$ is called* rational *if it is an element of the rational closure of $\mathbb{Q}\langle \Sigma \rangle$ (operations of sum, Cauchy product, external product of $\mathbb{Q}$ on $\mathbb{Q}\langle\langle \Sigma \rangle\rangle$, star).*

Definition 3. *A series $\phi \in \mathbb{Q}\langle\langle \Sigma \rangle\rangle$ is said recognizable if and only if there exist an integer $n \geq 1$, a morphism of monoids $\mu : \Sigma^\star \mapsto \mathbb{Q}^{n\times n}$, a row vector $\lambda \in \mathbb{Q}^{1\times n}$ and a column vector $\gamma \in \mathbb{Q}^{n\times 1}$ such that for all $w \in \Sigma^\star$ $(\phi, w) = \lambda\mu w\gamma$. The triple $\langle \lambda, \mu, \gamma \rangle$ is called the linear representation of ϕ.*

Theorem 1 ([17]). *A formal series is recognizable if and only if it is rational.*

We refer to [3] for a detailed introduction to the class of rational formal series. When we consider series on a f.p.c. monoid $\mathrm{F}(\Sigma, C)$, we say that $\phi : \mathrm{F}(\Sigma, C) \mapsto \mathbb{Q}$ is rational if and only if there exists a linear representation $\langle \lambda, \mu, \gamma \rangle$ such that

$$(\phi, t) = \sum_{\substack{w \in \Sigma^\star \\ [w]_C = t}} \lambda\mu w\gamma \ .$$

Given a subset $L \subseteq M$, the *characteristic series* of L is the formal series $\chi_L : M \mapsto \{0, 1\}$ defined as

$$(\chi_L, m) = \begin{cases} 1 & m \in L, \\ 0 \text{ otherwise} & . \end{cases}$$

Given a formal series ϕ, the generating function associated with ϕ is the function

$$f_\phi(z) = \sum_{m \in M} (\phi, m) z^{|m|} \ .$$

In particular, the *generating function* of L is the generating function associated with the characteristic series of L, that is,

$$f_L(z) = f_{\chi_L}(z) = \sum_{m \in M} (\chi_L, m) z^{|m|} = \sum_{n \geq 0} c_n z^n \ ,$$

where $c_n = \sharp\{m \in L \mid |m| = n\}$.

We also consider formal series on the free commutative monoid Σ^c. A formal series in commutative variables is a function $\psi : \Sigma^c \mapsto \mathbb{Q}$,

$$\psi(\sigma_1, \ldots, \sigma_n) = \sum_{\underline{\sigma^\imath} \in \Sigma^c} \psi(\underline{\sigma^\imath})\underline{\sigma^\imath} = \sum_{\underline{\imath} \geq \underline{0}} \psi(\underline{\sigma^\imath})\underline{\sigma^\imath} \ ,$$

where $\psi(\underline{\sigma^\imath})$ indicates the coefficient in ψ of the monomial $\underline{\sigma^\imath}$. In the rest of the paper, we use the operator $[\underline{\sigma^\imath}]$ applied to $\psi(\sigma_1, \ldots, \sigma_n)$, $[\underline{\sigma^\imath}]\psi(\sigma_1, \ldots, \sigma_n)$, in order to extract the coefficient $\psi(\underline{\sigma^\imath})$.
The set of formal series in commutative variables Σ with coefficients in $\mathbb{Q}$ is denoted by $\mathbb{Q}[[\Sigma]]$. On $\mathbb{Q}[[\Sigma]]$ we consider the usual operations of sum $(+)$, Cauchy product $(\cdot)$ and

- Partial derivative: $(\partial_i\phi)(\sigma_1^{a_1}\cdots\sigma_i^{a_i}\cdots\sigma_n^{a_n}) = (a_i+1)\phi(\sigma_1^{a_1}\cdots\sigma_i^{a_i+1}\cdots\sigma_n^{a_n})$,
- Primitive diagonal: if $p \neq q$ then $(\Delta_{pq}(\phi))(\sigma_1^{i_1}\cdots\sigma_{q-1}^{i_{q-1}}\sigma_{q+1}^{i_{q+1}}\cdots\sigma_n^{i_n}) = \phi(\sigma_1^{i_1}\cdots\sigma_{q-1}^{i_{q-1}}\sigma_q^{i_p}\sigma_{q+1}^{i_{q+1}}\cdots\sigma_n^{i_n})$,
- Substitution: $\phi(\psi_1(\tau_1,\ldots,\tau_m),\ldots,\psi_n(\tau_1,\ldots,\tau_m))$.

Some interesting subclasses of $\mathbb{Q}[[\Sigma]]$ are the class of commutative polynomials $\mathbb{Q}[\Sigma]$ and the class of rational formal series $\mathbb{Q}[[\Sigma]]_r$. A series $\phi \in \mathbb{Q}[[\Sigma]]_r$ can be thought as the power series expansion of a function P/Q with $P, Q \in \mathbb{Q}[\Sigma]$ and $Q(0) = 1$. It is known that $\mathbb{Q}[[\Sigma]]_r$ is not closed with respect to the Hadamard product and primitive diagonal (see, for instance, [10]), and so it is quite natural to look for an extension of $\mathbb{Q}[[\Sigma]]_r$ that is closed with respect to these operations: this extension consists of the class of holonomic series $\mathbb{Q}[[\Sigma]]_h$, formally defined as follows.

Definition 4. *A formal series $\phi \in \mathbb{Q}[[\Sigma]]$ is said to be holonomic iff there exist some polynomials $p_{ij} \in \mathbb{Q}[\Sigma]$, $1 \leq i \leq n$, $\quad 0 \leq j \leq d_i$, $p_{id_i} \neq 0$, such that*

$$\sum_{j=0}^{d_i} p_{ij}\partial_i^j\phi = 0, \quad 1 \leq i \leq n \ .$$

As a matter of fact, holonomic series are power series expansions of suitable functions that belong to the class of holonomic functions. This class was first introduced by I.N. Bernstein in the '1970 ([2]) and deeply investigated by Stanley, Lipshitz, Zeilberger et al. (see [9], [13], [14], [18] and [20]).
The closure properties of $\mathbb{Q}[[\Sigma]]_h$ are summarized in the following theorem.

Theorem 2. *The class $\mathbb{Q}[[\Sigma]]_h$ is closed under the operations of sum, Cauchy product, Hadamard product, primitive diagonal, substitution with algebraic series.*

Proof. See, for instance, [14]. □

At last, we recall that $\mathbb{Q}[[\Sigma]]_h$ properly contains the class $\mathbb{Q}[[\Sigma]]_a$ of algebraic formal series (see [16] for a definition of algebraic series). So, we have

$$\mathbb{Q}[[\Sigma]]_r \subset \mathbb{Q}[[\Sigma]]_a \subset \mathbb{Q}[[\Sigma]]_h \ .$$

A language $L \subseteq \Sigma^\star$ is said to be *rational* if and only if it is the support of a rational series on $\Sigma^\star$. In the case of trace languages we have the following:

Definition 5. *A trace language $L \in F(\Sigma, C)$ is rational if and only if there exists a rational series ϕ on $F(\Sigma, C)$ such that $L = Supp(\phi)$.*

Definition 6. *A trace language $L \in F(\Sigma, C)$ is said unambiguous rational if and only if there exists a rational series $\phi : F(\Sigma, C) \mapsto \{0, 1\}$ such that $L = Supp(\phi)$.*

We denote by $\mathrm{Rat}(\Sigma, C)$ the set of rational trace languages on $F(\Sigma, C)$ and by $\mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$ the set of unambiguous rational trace languages.

3 The Inclusion Problem for $\mathbf{Rat_U}(\Sigma, C)$

We formally define the Inclusion Problem for $\mathrm{Rat_U}(\Sigma, C)$ as follows.

Problem (Inclusion for $\mathbf{Rat_U}(\Sigma, C)$). *Given two linear representations*

$$\langle \lambda_1, \mu_1, \gamma_1 \rangle , \quad \langle \lambda_2, \mu_2, \gamma_2 \rangle$$

defining two unambiguous rational trace languages L_1, L_2, decide whether

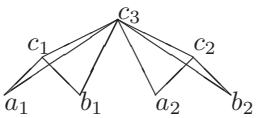
$$L_1 \subseteq L_2 \ .$$

We approach the Inclusion Problem by means of generating functions. In fact, given two subsets of a monoid, $L_1, L_2 \subseteq M$, it is immediate to note that $L_1 \subseteq L_2$ if and only if $L_1 \cap L_2 = L_1$, that is, if and only if $f_{\chi_{L_1} \odot \chi_{L_2}}(z) = f_{\chi_{L_1}}(z)$. Therefore, a natural way to solve Inclusion for $\mathrm{Rat_U}(\Sigma, C)$ leads to the following steps:

Step 1: Given $L_1, L_2 \in \mathrm{Rat_U}(\Sigma, C)$ compute $f_{\chi_{L_1} \odot \chi_{L_2}}(z)$ and $f_{\chi_{L_1}}(z)$.
Step 2: Given $f_{\chi_{L_1} \odot \chi_{L_2}}(z)$, $f_{\chi_{L_1}}(z)$, decide whether $f_{\chi_{L_1} \odot \chi_{L_2}}(z) = f_{\chi_{L_1}}(z)$.

By showing that for a particular commutation relation C Step 1 and Step 2 are decidable, we prove the following:

Theorem 3. *The Inclusion Problem for $Rat_U(\Sigma, C)$ is decidable when C is given by*



Observe that the previous commutation relation can be obtained by considering a complete binary tree of height 2 (with symbols associated with nodes) and setting that two nodes a, b commute if and only if a is in the subtree rooted at b or vice versa.

3.1 Step 1: Computing $f_{\chi_{L_1} \odot \chi_{L_2}}(z)$ and $f_{\chi_{L_1}}(z)$

In this section we show two examples that illustrate how to compute the generating function $f_{\chi_{L_1} \odot \chi_{L_2}}(z)$ when $L_1, L_2 \in \mathrm{Rat_U}(\Sigma, C)$ are subsets of particular monoids that belong to $\mathcal{M}$ (see Definition 1). Recall that the characteristic series $f_{\chi_L}(z)$ of $L \in \mathrm{Rat_U}(\Sigma, C)$ is rational and so it admits a linear representation.

Example 2. Let $L_1, L_2 \in \mathrm{Rat_U}(\{a, b\}, \emptyset)$ be defined by two rational characteristic series χ_{L_1}, χ_{L_2} (on the free monoid $a^\star \cdot b^\star$) with linear representations $\langle \lambda_1, \mu_1, \gamma_1 \rangle$, $\langle \lambda_2, \mu_2, \gamma_2 \rangle$ respectively. Given $w = x_1 \cdots x_n$, $x_i \in \{a, b\}$, we have

$$\begin{aligned} (\chi_{L_1}, x_1 \cdots x_n) &= \lambda_1 \mu_1(x_1) \cdots \mu_1(x_n) \gamma_1 \ , \\ (\chi_{L_2}, x_1 \cdots x_n) &= \lambda_2 \mu_2(x_1) \cdots \mu_2(x_n) \gamma_2 \end{aligned}$$

and, by a well-known result (see, for instance, chapter 2 in [16]),

$$(\chi_{L_1} \odot \chi_{L_2}, x_1 \cdots x_n) = \lambda_1 \otimes \lambda_2 \cdot \mu_1(x_1) \otimes \mu_2(x_1) \cdots \mu_1(x_n) \otimes \mu_2(x_n) \cdot \gamma_1 \otimes \gamma_2 \ ,$$

where $\otimes$ is the usual Kronecker product of matrices.
Then, the generating function associated with $\chi_{L_1} \odot \chi_{L_2}$ is

$$\begin{aligned} f_{\chi_{L_1} \odot \chi_{L_2}}(z) &= \sum_{w \in \{a,b\}^\star} (\chi_{L_1} \odot \chi_{L_2}, w) z^{|w|} = \sum_{n \geq 0} z^n \sum_{\substack{w \in \{a,b\}^\star \\ |w|=n}} (\chi_{L_1} \odot \chi_{L_2}, w) = \\ &= \sum_{n \geq 0} z^n \lambda_1 \otimes \lambda_2 \cdot (\mu_1(a) \otimes \mu_2(a) + \mu_1(b) \otimes \mu_2(b))^n \cdot \gamma_1 \otimes \gamma_2 = \\ &= \lambda_1 \otimes \lambda_2 \left(I - z(\mu_1(a) \otimes \mu_2(a) + \mu_1(b) \otimes \mu_2(b))\right)^{-1} \cdot \gamma_1 \otimes \gamma_2 \end{aligned}$$

and belongs to $\mathbb{Q}[[z]]_r$.

The next example deals with unambiguous rational trace languages that are subsets of the monoid $(a^\star \cdot b^\star) \times c^\star \in \mathcal{M}$.

Example 3. Let $L_1, L_2 \in \mathrm{Rat}_{\mathrm{U}}(\{a,b,c\}, {}_a{}^c\!\wedge_b)$ be defined by the rational characteristic series χ_{L_1}, χ_{L_2} with linear representations $\langle \lambda_1, \mu_1, \gamma_1 \rangle$, $\langle \lambda_2, \mu_2, \gamma_2 \rangle$ respectively. We define the matrices

$$A_\sigma(c) = \mu_1(\sigma) \cdot \sum_{k \geq 0} c^k \mu_1(c)^k = \mu_1(\sigma) \cdot (I - c\mu_1(c))^{-1} \quad (\sigma \in \{a,b\}) \ ,$$

$$A'_\sigma(c) = \mu_2(\sigma) \cdot \sum_{k \geq 0} c^k \mu_2(c)^k = \mu_2(\sigma) \cdot (I - c\mu_2(c))^{-1} \ ,$$

and the row vectors

$$\Lambda_1(c) = \lambda_1 \sum_{k \geq 0} c^k \mu_1(c)^k = \lambda_1 \cdot (I - c\mu_1(c))^{-1} \ ,$$

$$\Lambda_2(c) = \lambda_2 \sum_{k \geq 0} c^k \mu_2(c)^k = \lambda_2 \cdot (I - c\mu_2(c))^{-1} \ .$$

Note that the entries in $A_\sigma(c), A'_\sigma(c), \Lambda_1(c)$ and $\Lambda_2(c)$ belong to $\mathbb{Q}[[c]]_r$. Therefore, for a trace $x_1 \cdots x_n c^k$, $x_i \in \{a,b\}$, we have

$$(\chi_{L_1}, x_1 \cdots x_n c^k) = [c^k] \Lambda_1(c) A_{x_1}(c) \cdots A_{x_n}(c) \gamma_1 \ ,$$

$$(\chi_{L_2}, x_1 \cdots x_n c^k) = [c^k] \Lambda_2(c) A'_{x_1}(c) \cdots A'_{x_n}(c) \gamma_2 \ .$$

By setting

$$A''_\sigma(c_1, c_2) = A_\sigma(c_1) \otimes A'_\sigma(c_2), \quad \Upsilon(c_1, c_2) = \Lambda_1(c_1) \otimes \Lambda_2(c_2), \quad \Gamma = \gamma_1 \otimes \gamma_2 \ ,$$

we have

$$(\chi_{L_1} \odot \chi_{L_2}, x_1 \cdots x_n c^k) = [c_1^k c_2^k](\Upsilon(c_1, c_2) \cdot A''_{x_1}(c_1, c_2) \cdots A''_{x_n}(c_1, c_2) \cdot \Gamma) \ .$$

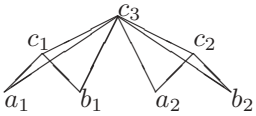
So, the generating function associated with $\chi_{L_1} \odot \chi_{L_2}$ is

$$\begin{aligned} f_{\chi_{L_1}\odot\chi_{L_2}}(z) &= \sum_{\substack{w\in\{a,b\}^\star \\ k\geq 0}} (\chi_{L_1}\odot\chi_{L_2}, wc^k) z^{|w|+k} = \\ &= \sum_{\substack{n\geq 0 \\ k\geq 0}} z^n z^k [c_1^k c_2^k] \sum_{\substack{w\in\{a,b\}^\star \\ |w|=n}} (\chi_{L_1}\odot\chi_{L_2}, w) = \\ &= \sum_{k\geq 0} z^k [c_1^k c_2^k] \sum_{\substack{n\geq 0 \\ x_1\cdots x_n\in\{a,b\}^n}} z^n \Upsilon(c_1,c_2)\cdot A''_{x_1}(c_1,c_2)\cdots A''_{x_n}(c_1,c_2)\cdot \Gamma = \\ &= \sum_{k\geq 0} z^k [c_1^k c_2^k] \Upsilon(c_1,c_2)(I-(A''_a(c_1,c_2)+A''_b(c_1,c_2))z)^{-1}\Gamma = \\ &= F(z,z) \ , \end{aligned}$$

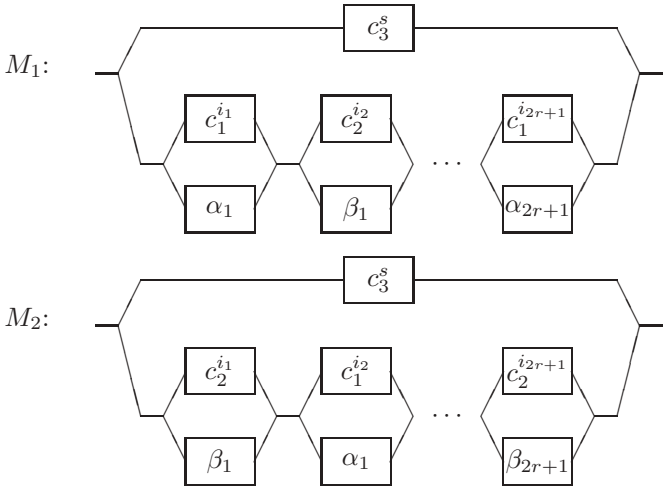
where $F(c_1, z) = \Delta_{c_1 c_2}(\Upsilon(c_1,c_2)(I-(A''_a(c_1,c_2)+A''_b(c_1,c_2))z)^{-1}\Gamma)$ is the diagonal of a rational function. So, the generating function $f_{\chi_{L_1}\odot\chi_{L_2}}(z)$ turns out to be algebraic (see [14]).

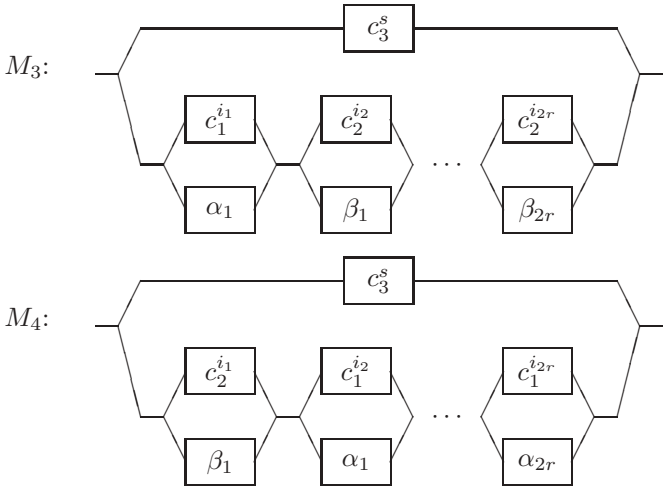
3.2 Rational Languages in $(((a_1^\star \cdot b_1^\star) \times c_1^\star) \cdot ((a_2^\star \cdot b_2^\star) \times c_2^\star)) \times c_3^\star$

Let C be the commutation relation defined by the following diagram:

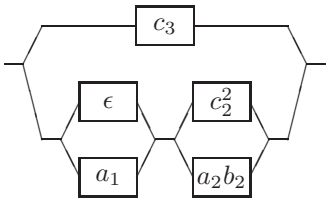


Then, it is easy to observe that there exist four classes $M_1, \ldots, M_4$ that describe the structure of all traces, as shown in the following block-diagrams ($\alpha_i \in \{a_1, b_1\}^\star, \beta_i \in \{a_2, b_2\}^\star$),





For instance, the trace $[a_1c_3a_2c_2b_2c_2]_{\rho_C}$ belongs to M_3 since its structure is



We now consider trace languages in $\mathrm{Rat_U}(\{a_1, b_1, c_1, a_2, b_2, c_2, c_3\}, C)$ and we prove the following:

Theorem 4. *Let $L_1, L_2 \in Rat_U(\{a_1, b_1, c_1, a_2, b_2, c_2, c_3\}, C)$. Then, the generating function $f_{\chi_{L_1} \odot \chi_{L_2}}(z)$ is holonomic.*

Proof. Let $L_1, L_2 \in \mathrm{Rat_U}(\{a_1, b_1, c_1, a_2, b_2, c_2, c_3\}, C)$ be defined by the rational characteristic series χ_{L_1}, χ_{L_2} with linear representations $\langle \lambda_1, \mu_1, \gamma_1 \rangle$, $\langle \lambda_2, \mu_2, \gamma_2 \rangle$ respectively. Let $\Lambda = \lambda_1 \otimes \lambda_2$ and $\Gamma = \gamma_1 \otimes \gamma_2$. For $\sigma \in \{a_1, b_1, a_2, b_2\}$ and $i = 1, 2$ we define the matrices of rational entries

$$
\begin{aligned}
C_i(c_i, c_3) &= \textstyle\sum_{k\geq 0}(c_i\mu_1(c_i) + c_3\mu_1(c_3))^k = (I - (c_i\mu_1(c_i) + c_3\mu_1(c_3))^{-1} \ , \\
C_i'(c_i, c_3) &= \textstyle\sum_{k\geq 0}(c_i\mu_2(c_i) + c_3\mu_2(c_3))^k = (I - (c_i\mu_2(c_i) + c_3\mu_2(c_3))^{-1} \ , \\
C_i''(c_i, c_3, c_i', c_3') &= C_i(c_i, c_3) \otimes C_i'(c_i', c_3') \ , \\
A_\sigma(c_i, c_3) &= \mu_1(\sigma) C_i(c_i, c_3) \ , \\
A_\sigma'(c_i, c_3) &= \mu_2(\sigma) C_i'(c_i, c_3), \\
A_\sigma''(c_i, c_3, c_i', c_3') &= A_\sigma(c_i, c_3) \otimes A_\sigma'(c_i', c_3')
\end{aligned}
$$

and

$$T_1(z, c_1, c_3, c_1', c_3') = (I - z(A''_{a_1}(c_1, c_3, c_1', c_3') + A''_{b_1}(c_1, c_3, c_1', c_3')))^{-1} - I \ ,$$

$$T_2(z, c_2, c_3, c_2', c_3') = (I - z(A''_{a_2}(c_2, c_3, c_2', c_3') + A''_{b_2}(c_2, c_3, c_2', c_3')))^{-1} - I \ .$$

Then, we define the matrices of algebraic entries

$$\xi_1 = \sum_{n\geq 0}(\Delta_{c_1,c_1'}T_1(\Delta_{c_2,c_2'}C_2'' + \Delta_{c_2,c_2'}T_2) + \Delta_{c_1,c_1'}C_1''(\Delta_{c_2,c_2'}T_2 + \Delta_{c_2,c_2'}C_2''))^n \ ,$$

$$\xi_2 = \sum_{n\geq 0}(\Delta_{c_2,c_2'}T_2(\Delta_{c_1,c_1'}C_1'' + \Delta_{c_1,c_1'}T_1) + \Delta_{c_2,c_2'}C_2''(\Delta_{c_1,c_1'}T_1 + \Delta_{c_1,c_1'}C_1''))^n \ .$$

By considering the partition of traces in 4 classes $M_1, \ldots, M_4$, we observe that the generating function $f_{\chi_{L_1}\odot\chi_{L_2}}(z)$ can be written as

$$f_{\chi_{L_1}\odot\chi_{L_2}}(z) = f^{(1)}_{\chi_{L_1}\odot\chi_{L_2}}(z) + f^{(2)}_{\chi_{L_1}\odot\chi_{L_2}}(z) + f^{(3)}_{\chi_{L_1}\odot\chi_{L_2}}(z) + f^{(4)}_{\chi_{L_1}\odot\chi_{L_2}}(z)$$

where $f^{(i)}_{\chi_{L_1}\odot\chi_{L_2}}(z) = \sum_{w\in\{M_i\}}(\chi_{L_1}\odot\chi_{L_2}, w)z^{|w|}$. By noting that $f^{(i)}_{\chi_{L_1}\odot\chi_{L_2}}(z) = F_i(z,z,z,z)$ with

$$\begin{aligned}
F_1(c_1,c_2,c_3,z) &= \Delta_{c_3,c_3'}(\Lambda\xi_1\Gamma) \ , \\
F_2(c_1,c_2,c_3,z) &= \Delta_{c_3,c_3'}(\Lambda(\Delta_{c_2,c_2'}T_2)\xi_1\Gamma) \ , \\
F_3(c_1,c_2,c_3,z) &= \Delta_{c_3,c_3'}(\Lambda(\Delta_{c_1,c_1'}T_1)\xi_2\Gamma) \ , \\
F_4(c_1,c_2,c_3,z) &= \Delta_{c_3,c_3'}(\Lambda(\Delta_{c_2,c_2'}T_2)T_1\xi_2)\Gamma) \ ,
\end{aligned}$$

we obtain that the functions $f^{(i)}_{\chi_{L_1}\odot\chi_{L_2}}(z)$ are holonomic: in fact, the entries in T_1,T_2, ξ_1, ξ_2 are holonomic and we know that the class of holonomic functions is closed under the operations of diagonal, product and substitution with algebraic functions. At last, we conclude that $f_{\chi_{L_1}\odot\chi_{L_2}}(z)$ is holonomic since it is the sum of holonomic functions. □

3.3 Step 2: $f_{\chi_{L_1}\odot\chi_{L_2}}(z) = f_{\chi_{L_1}}(z)$?

Given two unambiguous rational trace languages $L_1, L_2 \subseteq (((a_1^\star \cdot b_1^\star) \times c_1^\star) \cdot ((a_2^\star \cdot b_2^\star) \times c_2^\star)) \times c_3^\star$, we know that the generating functions $f_{\chi_{L_1}}(z)$, $f_{\chi_{L_2}}(z)$ and $f_{L1\cap L_2}(z) = f_{\chi_{L_1}\odot\chi_{L_2}}(z)$, are holonomic. So, starting from the rational characteristic series χ_{L_1}, χ_{L_2}, it is possible to compute three linear differential equations with polynomial coefficients satisfied by the functions $f_{\chi_{L_1}}(z)$, $f_{\chi_{L_2}}(z)$ and $f_{\chi_{L_1}\odot\chi_{L_2}}(z)$ respectively (see [20] for details on computing with holonomic functions). Useful packages for these computations under Maple may be found in the Chyzak's Mgfun Project page (http://algo.inria.fr/chyzak/mgfun.html).

Hence, the problem of verifying whether $f_{\chi_{L_1}\odot\chi_{L_2}}(z) = f_{\chi_{L_1}}(z)$ is reduced to the problem of testing that the holonomic function $f_{\chi_{L_1}\odot\chi_{L_2}}(z) - f_{\chi_{L_1}}(z)$ is the zero function. So, we first compute the linear differential equation

$$\sum_{i=0}^{d} q_i(z)D^i g(z) = 0$$

satisfied by $g(z) = f_{\chi_{L_1}\odot\chi_{L_2}}(z) - f_{\chi_{L_1}}(z) = \sum_{n\geq 0} c_n z^n$. Then, by noting that $zDg(z)$ corresponds to $\{nc_n\}$ and $z^k g(z)$ to $\{c_{n-k}\}$, we find a linear recurrence

equation with polynomial coefficients satisfied by $\{c_n\}$, $\sum_{i=0}^{e} p_i(n)c_{n-i} = 0$. At last, $f_{\chi_{L_1} \odot \chi_{L_2}}(z) - f_{\chi_{L_1}} = 0$ if and only if $c_i = 0$, $i = 0, \ldots e$, and $\{0\}$ satisfies the recurrence equation.

3.4 Conclusions

We conclude with two remarks. We have shown that Inclusion is decidable for $\mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$ with respect to a particular commutation relation associated with a complete binary tree of height 2. Thus, what happens if we consider commutation relations associated with complete binary trees of arbitrary height (two nodes commute if and only if one is in the subtree of the other)? Note that traces in such languages have a serial/parallel structure so it naturally rises a more general question: is it possible to extend Theorem 4 to unambiguos rational trace languages with commutation relations as those associated with monoids in $\mathcal{M}$ (see Definition 1)?
Another interesting topic is that of studying the complexity of Inclusion for $\mathrm{Rat}_{\mathrm{U}}(\Sigma, C)$, a problem closely related to the complexity of computing with holonomic functions, for which several results are known (see, for instance, [6]).

Acknowledgement

We greatly thank professor A. Bertoni of Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, for stimulating discussions and suggestions.

References

1. IJ. J. Aalbersberg, H. J. Hoogeboom, *Characterizations of the decidability of some problems for regular trace languages*, Mathematical Systems Theory, **22** (1989), p. 1–19.
2. I. N. Bernstein, *Modules over a ring of differential operators, study of the fundamental solutions of equations with constant coefficients*, Functional Anal. Appl. vol. 5 (1971), p. 1-16 (russian); p. 89–101 (english).
3. J. Berstel, C. Reutenauer, *Rational Series and Their Languages*, Springer-Verlag, EATCS Monographs on Theoretical Computer Science, vol. 12 (1988).
4. A. Bertoni, M. Goldwurm, G. Mauri, N. Sabadini, *Counting Techniques for Inclusion, Equivalence and Membership Problems*, in The Book of Traces (V. Diekert, G. Rozemberg eds.), World Scientific (1995), p. 131–163.
5. A. Bertoni, G. Mauri, N. Sabadini, *Unambiguous regular trace languages*, Proc. of the Coll. on Algebra, Combinatorics and Logic in Computer Science, Colloquia Mathematica Soc. J. Bolyay, North Holland, vol.42 (1985), p. 113–123.
6. A. Bertoni, P. Massazza, *A parallel algorithm for the hadamard product of holonomic formal series*, IJAC, vol.4 N.4 (1994), p. 561–573.
7. A. Bertoni, P. Massazza, *On the Inclusion Problem for Finitely Ambiguous Rational Trace Languages*, RAIRO Theoretical Informatics and Applications, vol.32 N.1-2-3 (1998), p. 79–98.

8. A. Bertoni, P. Massazza, N. Sabadini, *Holonomic generating functions and context free languages*, International Journal of Foundations of Computer Science, vol.3 N.2 (1992), p. 181–191.
9. F. Chyzak. *An Extension of Zeilberger's Fast Algorithm to General Holonomic Functions*, Proceedings of FPSAC 1997, Universitat Wien (1997), p. 172–183.
10. H. Furstenberg, *Algebraic functions over finite fields*, Journal of Algebra, **7** (1967), p. 271–277.
11. A. Gibbons, W. Rytter, *On the decidability of some problems about rational subsets of free partially commutative monoids*, Theoretical Computer Science, **48** (1986), p. 329–337.
12. O. H. Ibarra, *Reversal-bounded multicounter machines and their decision problems*, Journal of the Association of Computing Machinery, vol.25 (1978), p. 116–133.
13. L. Lipshitz. *The diagonal of a D−Finite Power Series is D−Finite*, Journal of Algebra, **113** (1988), p. 373–378.
14. L. Lipshitz, *D-Finite Power Series*, Journal of Algebra, **122** (1989), p. 353–373.
15. A. Mazurkiewicz, *Concurrent program schemes and their interpretations*, DAIMI Rep. PB-78, Aarhus Univ., 1977.
16. A. Salomaa, M. Soittola, *Automata-Theoretic Aspects of formal power series*, Springer-Verlag, New York, 1978.
17. M.P. Schützenberger, *On the definition of a family of automata*, Information and Control **4** (1961), p.245–270.
18. R.P. Stanley. *Differentiably Finite Power Series*, European Journal of Combinatorics, **1** (1980), p. 175–188.
19. S. Varricchio, *On the decidability of the equivalence problem for partially commutative rational power series*, LITP, Univ. Paris VII, Internal report 90.97, 1990.
20. D. Zeilberger, *A holonomic systems approach to special functions identities*, J. Comput. Appl. Math., **32** (1990), p. 321–368.

LR Parsing for Boolean Grammars

Alexander Okhotin*

Department of Mathematics, University of Turku, Turku FIN-20014, Finland
http://www.cs.queensu.ca/home/okhotin/

Abstract. The generalized LR parsing algorithm for context-free grammars, invented by Tomita in 1986, is extended for the case of Boolean grammars, which are a recently introduced generalization of context-free grammars with logical connectives added to the formalism of rules. In addition to the standard LR operations, *Shift* and *Reduce*, the new algorithm uses a third operation called *Invalidate*, which reverses a previously made reduction; this makes the algorithm considerably different from its prototype, though it can still be made to work in time $O(n^3)$.

1 Introduction

The generalized LR parsing was introduced in 1986 by Tomita [15] as a polynomial-time method of simulating nondeterminism in the classical Knuth's LR [5]. Every time a deterministic LR parser is faced with a choice of actions to perform (Shift or Reduce), a generalized LR parser performs both actions at the same time, storing all possible contents of an LR parser's stack in the form of a graph. Although the number of possible computations of a nondeterministic LR parser can exponentially depend on the length of the string, the compact graph-structured representation always contains $O(n)$ vertices and therefore fits in $O(n^2)$ memory. If carefully implemented, the algorithm is applicable to any context-free grammar, and its complexity can be bounded by a polynomial of degree as low as cubic.

Initially, the algorithm was proposed with linguistic applications in mind [15], and in the recent years there has been a growing interest in the use of this method in the programming languages community [2]. In particular, efficient implementation techniques are being researched [3], and application-oriented parser generators are being implemented [6]. The algorithm has also been extended for conjunctive grammars by the author [8].

In this paper, the generalized LR parsing algorithm is further extended to handle the case of *Boolean grammars* [10], which are context-free grammars augmented with Boolean operations in the formalism of rules. Boolean grammars can specify many abstract non-context-free languages, such as $\{a^n b^n c^n \mid n \geqslant 0\}$, $\{ww \mid w \in \{a,b\}^*\}$ and $\{a^{2^n} \mid n \geqslant 0\}$, the latter being outside of the Boolean closure of the context-free languages. Another evidence of their expressive power is given by a fairly compact grammar for the set of well-formed programs in a simple model programming language [12], which is the first specification of any programming language by a formal grammar from a computationally feasible

* Supported by the Academy of Finland under grant 206039

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 362–373, 2005.
© Springer-Verlag Berlin Heidelberg 2005

class. The generalized LR algorithm presented in this paper allows to convert this particular grammar to a square-time correctness checker. The algorithm has been implemented in an ongoing parser generator project [9], and such a correctness checker has been successfully produced out of that Boolean grammar.

The most interesting quality of the proposed algorithm is that the negation is implemented by removing arcs from the graph-structured stack. In terms of the theory of parsing schemata [14], this means that the atomic items can be not only gained, but also lost, and regained back, etc. This is a clear departure from the paradigm of *parsing as deduction* [13]. Establishing the correctness of a parsing algorithm that behaves in such an uncommon way is a new task to be solved. Let us see how this can be done.

2 Boolean Grammars

Boolean grammars are context-free grammars augmented with propositional connectives. In addition to the implicit disjunction represented by multiple rules for a single nonterminal, Boolean grammars include explicit conjunction and negation in the formalism of rules.

Definition 1 ([10]). *A Boolean grammar [10] 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 of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \quad (m+n \geqslant 1,\ \alpha_i, \beta_i \in (\Sigma \cup N)^*), \tag{1}$$

while $S \in N$ is the start symbol of the grammar. For each rule (1), the objects $A \to \alpha_i$ and $A \to \neg\beta_j$ (for all i, j) are called conjuncts, positive and negative respectively. A conjunct with unknown sign can be denoted $A \to \pm\gamma$, which means "$A \to \gamma$ or $A \to \neg\gamma$". Let $conjuncts(P)$ be the sets of all conjuncts, let $uconjuncts(P) = \{A \to \gamma \mid A \to \pm\gamma \in conjuncts(P)\}$.

A Boolean grammar is called a conjunctive grammar [7], if negation is never used, i.e., $n = 0$ for every rule (1); it degrades to a standard context-free grammar if neither negation nor conjunction are allowed, i.e., $m = 1$ and $n = 0$ for all rules. Assume, without loss of generality, that $m \geqslant 1$ in every rule. Intuitively, a rule (1) can be read as *"if a string satisfies the syntactical conditions $\alpha_1, \ldots, \alpha_m$ and does not satisfy any of the syntactical conditions $\beta_1, \ldots, \beta_n$, then this string satisfies the condition represented by the nonterminal A".* Formal semantics of Boolean grammars (i.e., the language specified by a grammar) is defined using systems of language equations with concatenation and all set-theoretic operations, similarly to the well-known characterization of the context-free grammars due to Ginsburg and Rice [4], which uses language equations with concatenation and union.

Definition 2. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. The system of language equations associated with G is a system over Σ in variables N, in which the equation for each variable $A \in N$ is*

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \in P} \left[\bigcap_{i=1}^{m} \alpha_i \ \cap \ \bigcap_{j=1}^{n} \overline{\beta_j} \right] \quad (\textit{for all } A \in N) \tag{2}$$

A vector $L = (L_1, \ldots, L_n)$ is called a naturally reachable solution of (2) if for every finite substring-closed $M \subseteq \Sigma^*$ and for every string $u \notin M$ (such that all proper substrings of u are in M) every sequence of vectors of the form

$$L^{(0)}, L^{(1)}, \ldots, L^{(i)}, \ldots \tag{3}$$

(where $L^{(0)} = (L_1 \cap M, \ldots, L_n \cap M)$ and every next vector $L^{(i+1)} \neq L^{(i)}$ in the sequence is obtained from the previous vector $L^{(i)}$ by substituting some j-th component with $\varphi_j(L^{(i)}) \cap (M \cup \{u\})$) converges to

$$(L_1 \cap (M \cup \{u\}), \ldots, L_n \cap (M \cup \{u\})) \tag{4}$$

in finitely many steps regardless of the choice of components at each step.

Let $(L_1, \ldots, L_n)$ be the naturally reachable solution of the system associated with a grammar; then the language $L_G(A_i)$ generated by every i-th nonterminal A can be defined as L_i, while the language of the grammar is $L(G) = L_G(S)$.

Despite the increased descriptive power, the theoretical upper bound for the parsing complexity for Boolean grammars is still $O(n^3)$ [10], the same as in the context-free case. As it will be demonstrated in this paper, a more practical parsing algorithm, the *Generalized LR*, can be generalized for Boolean grammars as well, and it also retains its worst-case cubic-time complexity.

3 The Parsing Table

Let us begin defining the generalized LR parsing algorithm for Boolean grammars from its parsing table. Internally, it is the same as in the deterministic context-free case [1, 5], but requires a newly formulated construction.

The first step is to construct a deterministic finite automaton over the alphabet $\Sigma \cup N$, called the LR(0) automaton, which recognizes the bodies of grammar rules in the stack. In our case this step is the same as in the context-free case. While in the context-free case the states of the LR(0) automaton are sets of dotted rules, dotted unsigned conjuncts are used in the case of Boolean grammars:

Definition 3. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. $A \to \alpha \cdot \beta$ is called a dotted conjunct, if the grammar contains a conjunct $A \to \pm\alpha\beta$. Let $\mathrm{dc}(P)$ denote the (finite) set of all dotted conjuncts.*

Let the set of states be $Q = 2^{\mathrm{dc}(P)}$. In order to define the initial state and the transitions between states, the functions *closure* and *goto* are used. They are defined as in the classical context-free LR theory, with the only difference that the objects they deal with are called conjuncts rather than rules.

For every set of dotted conjuncts X and for every $s \in \Sigma \cup N$, define

$$goto(X, s) = \{A \to \alpha s \cdot \beta \mid A \to \alpha \cdot s\beta \in X\} \tag{5}$$

Next, $closure(X)$ is defined as the least set of dotted conjuncts that contains X and satisfies the condition that for each $A \to \alpha \cdot B\gamma \in closure(X)$ (where

$\alpha, \gamma \in (\Sigma \cup N)^*$, $B \in N$) and for each conjunct $B \to \pm\beta \in \mathit{conjuncts}(P)$ it holds that $B \to \cdot\beta \in \mathit{closure}(X)$.

Define the initial state of the automaton as $q_0 = \mathit{closure}(\{S \to \cdot\sigma \mid S \to \pm\sigma \in \mathit{conjuncts}(P)\})$, while the transition from a state $q \subseteq \mathrm{dc}(P)$ by a symbol $s \in \Sigma \cup N$ is defined as follows: $\delta(q, s) = \mathit{closure}(\mathit{goto}(q, s))$. The state $\varnothing \subset \mathrm{dc}(P)$ is an error state and will be denoted by $-$. Note that, in the terminology of Aho, Sethi and Ullman [1], $\delta(q, a) = q'$ ($a \in \Sigma$) is expressed as "$\textsc{Action}[q, a] =$ Shift q'", while $\delta(q, A) = q'$ ($A \in N$) means "$\textsc{Goto}(q, A) = q'$".

The finite automaton constructed so far recognizes the bodies of the conjuncts, providing the pertinent information in the states it computes. The other component of an LR automaton, the *reduction function*, decodes this information from the numbers of the states and reports which rules can be applied.

Let $\Sigma^{\leqslant k} = \{w \mid w \in \Sigma^*, |w| \leqslant k\}$. For any string w, define

$$\mathit{First}_k(w) = \begin{cases} w, & \text{if } |w| \leqslant k \\ \text{first } k \text{ symbols of } w, & \text{if } |w| > k \end{cases} \tag{6}$$

This definition is extended to languages as $\mathit{First}_k(L) = \{\mathit{First}_k(w) \mid w \in L\}$.

In the case of the context-free SLR(k), the reduction function is constructed using the sets $\textsc{Follow}_k(A) \subseteq \Sigma^{\leqslant k}$ ($A \in N$) that specify the possible continuations of strings generated by a nonterminal A. This is formalized by context-free derivations: $u \in \textsc{Follow}_k(A)$ means that there exists a derivation $S \Longrightarrow^* xAy$, such that $\mathit{First}_k(y) = u$. The corresponding notion for the case of Boolean grammars is, in the absence of derivation, somewhat harder to define:

Definition 4 ([11]). *Let us say that $u \in \Sigma^*$ follows $\sigma \in (\Sigma \cup N)^*$ if there exists a sequence of conjuncts $A_0 \to \pm\alpha_1 A_1 \beta_1$, $A_1 \to \pm\alpha_2 A_2 \beta_2$, $\ldots$, $A_{\ell-1} \to \pm\alpha_\ell A_\ell \beta_\ell$, $A_\ell \to \pm\eta\sigma\theta$, such that $A_0 = S$ and $u \in L_G(\theta\beta_\ell \ldots \beta_1)$*

Now, for every nonterminal $A \in N$, define $\textsc{First}_k(A) = \mathit{First}_k(L_G(A))$ and $\textsc{Follow}_k(A) = \{\mathit{First}_k(u) \mid u \text{ follows } A\}$.

Already for conjunctive grammars there cannot exist an algorithm to compute the sets $\textsc{First}_k$ and $\textsc{Follow}_k$ precisely [8]. However, since the LR algorithm uses the lookahead information solely to eliminate some superfluous reductions, if the sets $\textsc{First}_k(A)$ and $\textsc{Follow}_k(A)$ are replaced by some of their supersets, the resulting LR parser will still work, though it will have to spend extra time doing some computations that will not influence the result. Algorithms to construct suitable supersets $\textsc{Pfirst}_k(A) \supseteq \textsc{First}_k(A)$ and $\textsc{Pfollow}_k(A) \supseteq \textsc{Follow}_k(A)$ have been developed for top-down parsing of Boolean grammars and can be found in the corresponding technical report [11]; let us reuse them for LR parsing.

The sets $\textsc{Pfollow}_k(A)$ are used to define the *reduction function* $R : Q \times \Sigma^{\leqslant k} \to 2^{\mathit{uconjuncts}(P)}$, which tells the conjuncts recognized in a given state if the unread portion of the string starts with a given k-character string. In the SLR(k) table construction method, it is defined as follows:

$$R(q, u) = \{A \to \alpha \mid A \to \alpha\cdot \in q,\ u \in \textsc{Pfollow}_k(A)\} \tag{7}$$

for every $q \in Q$ and $u \in \Sigma^{\leqslant k}$. In the notation of Aho, Sethi and Ullman, $A \to \alpha \in R(q,u)$ means "ACTION$[q,u]$ = Reduce $A \to \alpha$", assuming $A \to \alpha \in P$.

As in the context-free case, the states from $Q \setminus \{-\}$ can be enumerated with consecutive numbers $0, 1, \ldots, |Q|-1$, where 0 refers to the state q_0.

4 The Algorithm

The new LR-class parsing algorithm for Boolean grammars is a generalization of the algorithm for conjunctive grammars [8], which in turn extends Tomita's algorithm [15] for context-free grammars.

All three algorithms share a common data structure: a *graph-structured stack*, introduced by Tomita [15] as a compact representation of the contents of the linear stack of Knuth's LR algorithm in all possible branches of nondeterministic computation. The graph-structured stack is an oriented graph with a designated *source node*. The nodes are labelled with the states of the LR automaton (such as the SLR(k) automaton constructed in the previous section), of which the source node is labelled with the initial state. The arcs are labelled with symbols from $\Sigma \cup N$. There is a designated nonempty collection of nodes, called *the top layer* of the stack. Every arc leaving one of these nodes has to go to another node from the top layer. The labels of these nodes should be pairwise distinct, and hence there can be at most $|Q|$ top layer nodes.

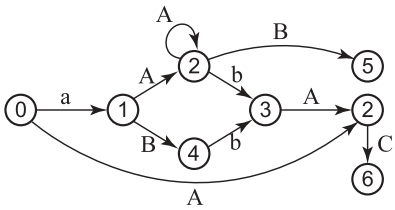


Fig. 1. Sample contents of the graph-structured stack

Consider the graph in Figure 1: the leftmost node labelled 0 is the source node; the three rightmost nodes labelled 5, 2 and 6 are assumed to form the top layer. There is another node labelled 2 (the direct predecessor of 5), which is not in the top layer.

Initially, the stack contains a single source node, which at the same time forms the top layer. The computation of the algorithm is an alternation of *reduction phases*, which modify the nodes in the top layer without reading the input, and *shift phases*, each reading and consuming a single input symbol and forming an entirely new top layer as a successor of the former top layer.

The shift phase is done identically in all three algorithms. Let a be the next input symbol. For each top layer node labelled with a state q, the algorithm looks up the transition table, $\delta(q,a)$. If $\delta(q,a) = q' \in Q$, then a new node labelled q' is created and q is connected to q' with an arc labelled a; this action is called *Shift* q'. If $\delta(q,a) = -$, no new nodes are created, and this condition is called

"Local error". The nodes created during a shift phase form the new top layer of the graph, while the previous top layer nodes are demoted to regular nodes. The branches of the graph-structured stack that do not get extended to the new top layer (due to local errors during the shift phase) are removed; if this is the case for all the nodes, then all the graph is effectively deleted, and the algorithm terminates, reporting a syntax error.

In Figure 2(a) the top layer contains the nodes 1, 2, 3 and 4. Since $\delta(1, a) = 5$, $\delta(2, a) = -$ and $\delta(3, a) = \delta(4, a) = 6$, a new top layer formed of 5 and 6 is created, while 2 and its predecessors that were disconnected from the new top layer are accordingly removed from the stack.

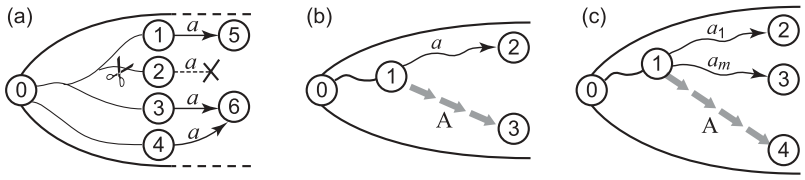


Fig. 2. (a) Shifting; (b) Reductions in context-free and (c) conjunctive cases

The reduction phase in each of the cases amounts to doing some identical transformations of the top layer until the stack comes to a stable condition, i.e., no further transformations are applicable. The difference between the three algorithms is in the particular transformations used.

In the **context-free** case [15], the only operation is *reduction*. Suppose there exists a top layer node q, a node q' and a rule $A \to \alpha$, such that $A \to \alpha \in R(q)$ and q' is connected to q by a path α. Then the algorithm can perform the operation "Reduce $A \to \alpha$" at q', adding a new arc labelled with A, which goes from q' to a top layer node labelled $\delta(q', A)$. If there is no node $\delta(q', A)$ in the top layer, it is created. This case is illustrated in Figure 2(b).

In the **conjunctive** case [8], *reduction* is still the only operation. However, now rules may consist of multiple conjuncts, and hence the conditions of performing a reduction are slightly more complicated. Let $A \to \alpha_1 \& \ldots \& \alpha_m$ be a rule, let q be a node and let $q_{i_1}, \ldots, q_{i_m}$ be top layer nodes, such that, for all j, $A \to \alpha_j \in R(q_{i_j})$ and q is connected to each q_{i_j} by a path α_j. The operation "Reduce $A \to \alpha_1 \& \ldots \& \alpha_m$" can be done, adding a new nonterminal arc A from q to a top layer node $\delta(q, A)$. See Figure 2(c).

The case of **Boolean** grammars is much more complicated. There are two operations: *reduction*, which is the same as in the previous cases, but with yet more complicated prerequisites, and *invalidation*, which means removing an arc placed by an earlier reduction. In order to reduce by a rule $A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n$ from a node q, this q should be connected to the top layer by each of the paths $\alpha_1, \ldots, \alpha_m$ and by none of the paths $\beta_1, \ldots, \beta_n$. This nonexistence of paths is shown in Figure 3(left) by dotted lines ending with crosses. This allows the algorithm to do "Reduce $A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \beta_n$", adding an arc labelled A from q to $\delta(q, A)$ in the top layer.

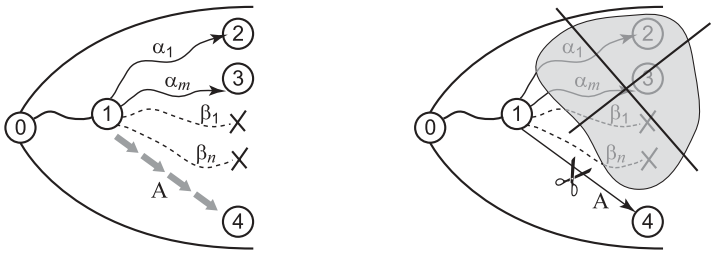


Fig. 3. Reduction phase for Boolean grammars: *Reduce* and *Invalidate*

Invalidation is the opposite of reduction. Suppose there exists a node q and an arc labelled with A from q to a node in the top layer, such that the conditions for making a reduction by any rule for A from the node q are not met – i.e., for every rule $A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \beta_n$ for A either for some i there is no path from q to the top layer labelled α_i, or for some j there exists a path from q to the top layer with the labels forming β_j. Then the earlier reduction (which added this A arc to the graph) has to be invalidated, by removing the arc from q to the top layer node $\delta(q, A)$. Note that an invalidation of an arc can make the graph disconnected.

Let us note that in the case of context-free and conjunctive grammars, in the absence of negation, arcs can only be added, and the conditions for invalidation would never hold. On the other hand, if there is negation, then a reduction by a rule $A \to \alpha \& \neg\beta$ at a node q can be made at the time when there is a path α from q to the top layer, but there is no path β; however, subsequent reductions may cause this path β to appear, rendering the earlier reduction invalid. This is something that does not have an analog in LR parsing for negation-free grammars.

The reduction phase as a whole is defined by the following nondeterministic algorithm:

while any reductions or invalidations can be done
    choose a nonempty set of reductions/invalidations to do
    add/remove these arcs simultaneously

The use of the nondeterminism leaves open a dangerous possibility of different computations' yielding different results. Also, if two possible actions are being done at once, one of them can change the graph-structured stack so that the prerequisites for making the other will no longer hold. Hence this kind of simultaneous transformation arouses natural suspicions regarding its validity. However, in Section 5 below the correctness of the algorithm will be proved for any possible choice of reductions/invalidations at every step, under certain reasonably weak assumptions on the grammar.

One possible implementation of the reduction phase is to do just one action at once; this is the obvious approach, yet it can theoretically lead to exponential time complexity. The implementation described in Section 6 does all valid reductions and invalidations at every step, which allows to prove a polynomial complexity upper bound.

Except for these major differences in the reduction phase, the three algorithms are identical in all other respects. Following is the general schedule of the generalized LR parsing:

Input: a string $w = a_1 \dots a_n$.
Let the stack contain a single node x labelled q_0, let the top layer be $\{x\}$.
do the *Reduction phase* using lookahead $First_k(w)$
for $i = 1$ **to** $|w|$
    do the *Shift phase* using a_i
    if the top layer is empty, **then** Reject
    do the *Reduction phase* using lookahead $First_k(a_{i+1} \dots a_n)$
    remove the nodes unreachable from the source node
if there is an arc S from the source node to $\delta(q_0, S)$ in the top layer, **then**
    Accept
else
    Reject

5 Proof of Correctness

Let us impose a certain weak condition on the grammar, which guarantees the algorithm's correctness.

Definition 5. *Let G be a Boolean grammar. Define the grammar*

$$G_+ = (\Sigma, N, \{A \to \alpha_1 \& \dots \& \alpha_m \mid A \to \alpha_1 \& \dots \& \alpha_m \& \neg\beta_1 \& \dots \& \neg\beta_n \in P\}, S)$$

The original grammar G is said to have a negatively fed cycle, if there exists a sequence of conjuncts $A_1 \to \pm\eta_1 A_2 \theta_1$, $A_2 \to \pm\eta_2 A_3 \theta_2$, $\dots A_\ell \to \pm\eta_\ell A_1 \theta_\ell$, such that $\varepsilon \in L_{G_+}(\eta_i)$ and $\varepsilon \in L_{G_+}(\theta_i)$ for all i, and also there exists another sequence of conjuncts $B_1 \to \pm\mu_1 B_2 \nu_1$, $B_2 \to \pm\mu_2 B_3 \nu_2$, $\dots B_k \to \neg\beta$, where $\varepsilon \in L_{G_+}(\nu_j)$ and $A_1 = B_1$.

Generally, a negatively fed cycle is something very unnatural, which one would not typically write (unless aiming to produce a counterexample for this algorithm, of course!), and which can be easily removed from a grammar. For example, the grammar $S \to S \mid a\&\neg aE$, $E \to \varepsilon$, which generates the language $\varnothing$, contains such a cycle ($\ell = k = 1$, $A_1 = B_1 = S$), and it can indeed lead the algorithm astray (towards accepting ε). In the following it will be required that a grammar has no negatively fed cycles.

In order to analyze the LR parsing algorithm, it is convenient to redefine the graph-structured stack by augmenting its nodes with the information on when they were added: a layer number corresponding to a position in the input. There is no need to store this extended information in an implementation; simply the properties of the algorithm become much clearer in these terms:

Definition 6. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let (Q, q_0, δ, R) be the SLR(k) automaton, let $w = a_1 \dots a_{|w|}$ be the input string.*

The graph-structured stack is an acyclic graph with a set of vertices $V \subseteq Q \times \{0, 1, \ldots, |w|\}$ *and with the arcs labelled with symbols from* $\Sigma \cup N$*, such that the following condition holds: for every arc from* (q', p') *to* (q'', p'') *labelled with* $s \in \Sigma \cup N$*,* $p' \leqslant p''$ *and* $\delta(q', s) = q''$*.*

For each p $(0 \leqslant p \leqslant n)$ *the set of all vertices of the form* (q, p)*, where* $q \in Q$*, is called the* p*-th layer of the graph. The nonempty layer with the greatest number is called the top layer.*

Definition 7. *An arc from a node* (q, p) *to a node* $(\delta(q, s), p')$ *labelled* s $(s \in \Sigma \cup N)$ *is called* ***correct*** *if and only if* $a_{p+1} \ldots a_{p'} \in L_G(s)$ *and, in the case* $s = A \in N$*,* $First_k(a_{p+1} \ldots a_n) \in \textsc{Pfollow}_k(A)$*.*

A correct arc is called ***reachable*** *if and only if either it originates from the source node, or it originates from a node to which there comes a correct arc known to be reachable.*

Let $0 \leqslant i \leqslant |w|$*. A reachable correct arc is called* i***-reaching*** *if and only if either it goes to the layer* i*, or it goes to a node, from which there originates a reachable correct arc known to be* i*-reaching.*

The goal of the whole algorithm is to construct the graph comprised of all reachable and $|w|$-reaching correct arcs. Let us first formulate the correctness statements for individual phases.

Lemma 1 (The computation done by the reduction phase). *Let* G *be a Boolean grammar without negatively fed cycles, let* (Q, q_0, δ, R) *be an SLR(k) automaton, let* $w \in \Sigma^*$ *be an input string, let* $p \geqslant 0$*. Suppose the graph-structured stack contains exactly the following arcs:*

- *all reachable and* p*-reaching correct arcs going to the layers* $0, 1, \ldots, p-1$*,*
- *all reachable and* p*-reaching correct terminal arcs coming to the layer* p*.*

Then every computation of the reduction phase terminates and constructs a graph with exactly the following arcs:

- *all reachable and* p*-reaching correct arcs going to the layers* $0, 1, \ldots, p-1, p$*,*
- *depending on the computation, some set of nonterminal arcs within the layer* p *that are not reachable from the source node.*

The reduction phase followed by the removal of unreachable arcs yields a graph consisting of exactly all reachable and p*-reaching correct arcs going to the layers* $0, 1, \ldots, p-1, p$*.*

The proof of the lemma is technically difficult; it proceeds as follows. First, it is established that the reduction phase always terminates, and every computation of the reduction phase ends with the same graph (modulo reachability). Second, a single "model computation" is presented, which constructs exactly what is stated in Lemma 1. Then it is left to conclude that all computations produce the same correct graph.

Lemma 1 confers the essence of the reduction phase. Let us now determine what does the shift phase do.

Lemma 2 (The computation done by the shift phase). *Let the graph contain exactly all reachable and p-reaching correct arcs up to the layer p. Then the shift phase from the layer p to the layer $p+1$ constructs a graph with the following arcs:*

- *all reachable and $(p+1)$-reaching correct arcs to the layers up to p;*
- *all reachable and $(p+1)$-reaching correct terminal arcs to the layer $p+1$.*

These lemmata can be combined to establish the algorithm's correctness.

Theorem 1 (Correctness of the algorithm). *Let G be a Boolean grammar without negatively fed cycles, let (Q, q_0, δ, R) be the $SLR(k)$ automaton for G, let $w \in \Sigma^*$ be an input string. Then the generalized LR algorithm constructed with respect to G and (Q, δ, R), terminates on w, and:*

- *After i iterations of the main loop, the graph-structured stack contains all reachable and i-reaching correct arcs to the layers $0, 1, \ldots, i$.*
- *The string is accepted if and only if $w \in L(G)$.*

6 Implementation

The overall composition of the algorithm has been given in Section 4, and it can be directly used in an implementation. It remains to give an algorithm for doing the reduction phase. The results of Section 5 imply that it can be implemented in many different ways, and the algorithm will always remain correct. The implementation suggested in this section is based upon doing all possible actions at every step of the reduction phase, and upon utilizing some straightforward breadth-first graph search techniques.

Definition 8. *Consider a fixed state of the graph-structured stack. For each vertex v and for each number $\ell \geqslant 0$, let $predecessors_\ell(v)$ be the set of vertices that are connected to v with a path that is exactly ℓ arcs long.*

Now the algorithm for doing the reduction phase reads as follows:

while the graph can be modified
  // *Conjunct gathering*
  let $x[\,]$ be an array of sets of vertices, indexed by conjuncts.
  for each node $v = (q, p_{top})$ in the top layer
    for each $A \to \alpha \in R(q, u)$
      $x[A \to \alpha] = x[A \to \alpha] \cup predecessors_{|\alpha|}(v)$
  // *Reductions*
  let *valid* be a set of arcs, initially empty
  for each rule $A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \in P$
    for each node $v \in \bigcap_{i=1}^{m} x[A \to \alpha_i] \setminus \bigcup_{i=1}^{n} x[A \to \beta_i]$
      if v is not connected to the top layer by an arc labelled A, **then**
        add an arc from v to the top layer labelled A
      $valid = valid \cup \{$the arc from v to the top layer labelled $A\}$

```
    // Invalidations
    for each arc (v', v) going to the top layer and labelled A
        if this arc is not in the set valid, then
            remove the arc from the graph
end while
```

On the stage of conjunct gathering, the algorithm scans the stack and determines, which reductions and invalidations can possibly be done. The set $x[\,]$ of gathered conjuncts stores this information, which refers to the state of the graph at the time of conjunct gathering. Then these operations are applied sequentially on the basis of the gathered conjuncts. This ensures that all reductions and invalidations are being done independently.

Let us give an upper bound for the complexity of this implementation of the reduction phase, and of the algorithm as a whole. The number of iterations in each reduction phase should be $O(n)$, because the dependencies of the arcs upon each other are $O(n)$ deep, and hence $O(n)$ parallel applications of all possible reductions and invalidations should be enough. It is worth note that if these operations are applied in series, then exponential time can be reached.

The conjunct gathering stage computes $predecessors_\ell$ a constant number of times. Computing the set of predecessors involves considering $O(n)$ nodes, each of which has $O(n)$ predecessors, and thus each conjunct gathering stage takes $O(n^2)$ steps. The reduction and invalidation stages take $O(n)$ time.

This yields a cubic upper bound for the complexity of the reduction phase, hence an $O(n^4)$ upper bound for the whole algorithm. The worst-case time can be improved to $O(n^3)$ by using a clumsy method for conjunct gathering [8].

7 On Parsing for Boolean Grammars

Another parsing method for Boolean grammars is a generalization of the well-known *recursive descent* [11]. As in the context-free case, the rules for each nonterminal $A \in N$ are mechanically transcribed into a procedure $A()$, which performs the parsing. The choice of a rule is determined by an almost standard LL(k) table, conjunction is implemented by scanning the input multiple times, while the machinery of exception handling is used to implement negation. This method is applicable to a proper subset of Boolean grammars, which is likely limited to the Boolean closure of the LL(k) context-free languages.

The generalized LR algorithm proposed in this paper is, in contrast, applicable to any Boolean grammar. It is as easy to implement as its prototypes [8, 15] and performs as efficiently. However, a quick glance at the (omitted) proof of its correctness shows that its mathematical justification is much more complicated. Let us try to consider it in terms of the theory of parsing schemata [14].

A parsing schema consists of a set of elementary propositions (items), axioms (items assumed to be true) and inference rules, which are used to deduce the truth of other items [14]. In the case of Boolean LR parsing, the items are the arcs in the stack, and the parser manipulates them as bits. However, the bits are not only set, but also reset and set back again, which certainly does not

fit into the deductive paradigm. The algorithm actually conducts a search for a solution of a Boolean equation. It would be interesting to extend the framework of parsing schemata to explain such "nonmonotonic" parsing.

Generally, it can be concluded that the major parsing techniques have been extended to Boolean grammars without loss of their feasibility. This gives further evidence that this theoretically defined family can be suitable for practical use.

References

1. A. V. Aho, R. Sethi, J. D. Ullman. *Compilers: Principles, Techniques and Tools*, Addison-Wesley, Reading, Mass., 1986.
2. J. Aycock, "Why Bison is becoming extinct", *ACM Crossroads*, 7 (2001).
3. J. Aycock, R. N. Horspool, J. Janousek, B. Melichar, "Even faster generalized LR parsing", *Acta Informatica*, 37:9 (2001), 633–651.
4. S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.
5. D. E. Knuth, "On the translation of languages from left to right", *Information and Control*, 8 (1965), 607–639.
6. S. McPeak, G. C. Necula, "Elkhound: a fast, practical GLR parser generator", *Compiler Construction* (Proceedings of CC 2004), LNCS 2985, 73–88.
7. A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.
8. A. Okhotin, "LR parsing for conjunctive grammars", *Grammars* 5 (2002), 81–124.
9. A. Okhotin, "Whale Calf, a parser generator for conjunctive grammars", *Implementation and Application of Automata* (CIAA 2002), LNCS 2608, 213–220.
10. A. Okhotin, "Boolean grammars", *Inform. and Comput.*, 194:1 (2004), 19–48.
11. A. Okhotin, "An extension of recursive descent parsing for Boolean grammars", Tech. Rep. 2004–475, School of Computing, Queen's University, Kingston, Canada.
12. A. Okhotin, "On the existence of a Boolean grammar for a simple programming language", *AFL 2005* (Dobogókö, Hungary), to appear.
13. S. M. Shieber, Y. Schabes, F. C. N. Pereira, "Principles and implementation of deductive parsing", *Journal of Logic Programming*, 24 (1995), 3–36.
14. K. Sikkel, *Parsing Schemata*, Springer-Verlag, 1997.
15. M. Tomita, "An efficient augmented context-free parsing algorithm", *Computational Linguistics*, 13:1 (1987), 31–46.

On Some Properties of the Language of 2-Collapsing Words

E.V. Pribavkina*

Ural State University,
620083 Ekaterinburg, Russia
selen_x@mail.ru

Abstract. We present two new results on 2-collapsing words. First, we show that the language of all 2-collapsing words over 2 letters is not context-free. Second, we prove that the length of a 2-collapsing word over an arbitrary finite alphabet Σ is at least $2|\Sigma|^2$ thus improving the previously known lower bound $|\Sigma|^2 + 1$.

Introduction

Throughout the word 'language' means a language over a fixed finite alphabet Σ and the word 'automaton' means a deterministic finite automaton with the input alphabet Σ. Recall that such an automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is defined by specifying a finite *state set* Q and a *transition function* $\delta : Q \times \Sigma \to Q$. The function δ naturally extends to the free monoid Σ^*; this extension is still denoted by δ. For each $v \in \Sigma^*$ and $q \in Q$ we write $q \,.\, v = \delta(q, v)$ and put $Q \,.\, v = \{q \,.\, v \mid q \in Q\}$.

The *deficiency* of a word $v \in \Sigma^+$ with respect to an automaton $\mathcal{A}$ is the number $\mathrm{df}(v) = |Q| - |Q \,.\, v|$. An automaton $\mathcal{A}$ is said to be *n-compressible* if there is a word $v \in \Sigma^*$ such that $\mathrm{df}(v) \geq n$ (the word v is then called *n-compressing with respect to $\mathcal{A}$*). A word $w \in \Sigma^*$ is called *n-collapsing* if it is n-compressing with respect to every n-compressible automaton whose input alphabet is Σ. Sauer and Stone [12] were arguably the first to introduce n-collapsing words (under the name '*words with property Δ_n*'); they did it in a purely combinatorial context. In particular, in [12] it was proved that such words exist for every n, a fact that is not trivial a priori. In [11] and [1] n-collapsing words have been applied to solving certain problems of universal algebra, the theory of transformation semigroups, and the theory of profinite semigroups. Connections between n-collapsing words and the Černý conjecture, a well-known open problem in automata theory, have been discussed in [3, 4, 9].

The study of the language $\mathcal{C}_n$ of all n-collapsing words was initiated in [5], where in particular it was announced that for $|\Sigma| > 1$ and $n > 1$ this language is not regular [5, Proposition 3]. (We note that a detailed proof of this fact is given in [5] only in case $n = 2$.) In [2] it is shown that the language $\mathcal{C}_2$ is context-sensitive. Thus, in order to determine the location of $\mathcal{C}_2$ within the classical Chomsky hierarchy, it remains to find out whether or not this language

* The work was supported by the Federal Education Agency of Russia, grant no. 49123

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 374–384, 2005.
© Springer-Verlag Berlin Heidelberg 2005

is context-free. In Section 2 of the present note we show that $\mathcal{C}_2$ is not context-free at least for $|\Sigma| = 2$.

Yet another intriguing open question about $\mathcal{C}_n$ concerns the length of the shortest words in $\mathcal{C}_n$. (We mention in passing that this question also may be of certain interest from the viewpoint of possible applications of n-collapsing words in biocomputing, see a discussion in the introduction to [4].) The problem of determining the minimum possible length of a n-collapsing word as a function $c(n,t)$ of n and $t = |\Sigma|$ was posed already in [12]. The best known bounds for $c(n,t)$ have been obtained in [9]:

$$t^n + n - 1 \le c(n,t) \le t^{\frac{1}{2}n(n+1)} + (n+1)t^{\frac{1}{2}n(n+1)-1} + \cdots + nt. \tag{1}$$

As for exact values of this function, only two of them have been found so far: $c(2,2) = 8$ [12] and $c(2,3) = 21$ [4]. In Section 3 we exhibit a new lower bound for the length of 2-collapsing words, namely $2t^2$, thus improving the lower bound $t^2 + 1$ that follows from (1) in case $n = 2$.

1 Preliminaries About 2-Collapsing Words

We start with recalling a simple result from [9]. A word $w \in \Sigma^*$ is said to be 2-*full*, if it has all the words of length 2 over Σ among its factors.

Proposition 1 ([9, Theorem 4.2]). *Any 2-collapsing word is 2-full.*

All our results make an essential use of the membership criterion for the language $\mathcal{C}_2$ proved in [3]. In its full generality this criterion is rather sophisticated but we need only some of its relatively simple special cases. For the reader's convenience we present them here in the form adapted to the use in this paper.

Let us choose an arbitrary letter $b \in \Sigma$ and denote $\Pi = \Sigma \setminus \{b\}$ (we assume that $|\Sigma| > 1$). Then any word $w \in \Sigma^+$ can be uniquely represented in the form

$$w = b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots p_m b^{\alpha_m} \tag{2}$$

with $\alpha_0, \alpha_m \ge 0$; $\alpha_i > 0$, $i = 1, \ldots, m-1$; $p_i \in \Pi^+$, $i = 1, \ldots, m$; $m \ge 0$. The factor p_i is said to be an *inner segment* of w if $\alpha_{i-1} \cdot \alpha_i \ne 0$, i.e. there is an occurrence of the letter b on both sides of p_i. By S we denote the set of all inner segments of the word w. Let $FG(\Pi)$ denote the free group over the alphabet Π; clearly, S can be treated as a subset of $FG(\Pi)$. We extend the operations of the group $FG(\Pi)$ to its subsets in a natural way so that, for instance, H^{-1} for $H \subseteq FG(\Pi)$ means $\{h^{-1} \mid h \in H\}$.

Proposition 2. *Given a 2-collapsing word $w \in \Sigma^+$, for any choice of a letter $b \in \Sigma$, the set S of all inner segments of the word w should satisfy the following conditions:*

(i) *the set S generates the group $FG(\Pi)$;*
(ii) *for any choice of a subset $P \subseteq S$, if $\overline{P} = S \setminus P$, and H_P is the subgroup of $FG(\Pi)$ generated by the set $P \cup (\overline{P} \cdot \overline{P}^{-1})$, then the index of H_P in $FG(\Pi)$ is not greater than 2.*

Proof. Proposition 2 is a special case of a more general result, see [3, Proposition 3.1]. For the reader's convenience and trying to make this note to a reasonable extent self-contained, we have extracted a proof of our proposition from the proof of that general result in [3].

Arguing by contradiction, suppose that for some choice of $b \in \Sigma$ and $P \subseteq S$, the subgroup H_P fulfills one of the two assumptions:

(a) $|FG(\Pi) : H_P| > 2$;

(b) $P = S$ and $|FG(\Pi) : H_P| = 2$.

In the case (a), take a subgroup H containing H_P and such that $2 < |FG(\Pi) : H| < \infty$. Such a subgroup H always exists. Indeed, if H_P itself is of finite index, we can simply set $H = H_P$. If the index of H_P in $FG(\Pi)$ is infinite, then by M. Hall's classical result [8], H_P as a finitely generated subgroup is equal to the intersection of all subgroups of finite index containing H_P. Some of these subgroups must have index greater than 2 and thus can be taken as H.

Starting from the chosen subgroup H we build an automaton $\mathcal{A}_H = \langle Q, \Sigma, \delta\rangle$. Its state set Q is the set of all left cosets of $FG(\Pi)$ with respect to H. We note that Q is finite since the index of H is finite. Each element of Q can be represented as Hp for some word $p \in \Pi^*$. To define the transition function δ, we fix an arbitrary coset $H' \neq H$. Then we set:

$Hp \,.\, a = Hpa$ for all $a \in \Pi$,

$Hp \,.\, b = Hp$ if $Hp \neq H$,

$H \,.\, b = Hp_0$ with $p_0 \in \overline{P}$ if $\overline{P} \neq \varnothing$,

$H \,.\, b = H'$ if $\overline{P} = \varnothing$.

The third rule may seem to be non-deterministic but this is not the case since for any $p_0, p_1 \in \overline{P}$ we have $p_0 p_1^{-1} \in H_P \subseteq H$ whence $Hp_1 = Hp_0 p_1^{-1} p_1 = Hp_0$.

By definition all letters $a \in \Pi$ act on the set Q as permutations, while $Q \,.\, b = Q \setminus \{H\}$. Since the action of a group on its left cosets with respect to a subgroup is transitive, there exists a word $p \in \Pi^*$ such that $H \in Q \,.\, bp$. This easily implies that $\mathrm{df}(bpb) \geq 2$ whence the automaton $\mathcal{A}_H$ is 2-compressible.

Next we consider the sequence

$$Q \,.\, b^{\alpha_0},\ Q \,.\, b^{\alpha_0} p_1,\ Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1}, \ldots, Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} \cdots p_m b^{\alpha_m} = Q \,.\, w$$

of sets and show by induction that $\mathrm{df}(w) = 1$.

Induction basis. If $\alpha_0 \neq 0$, then $Q \,.\, b^{\alpha_0} = Q \setminus \{H\}$, else $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} = Q \setminus \{H\}$.

Induction step. Assume that $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} = Q \setminus \{H\}$ for some $k < m$. Then $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} p_k = Q \setminus \{H \,.\, p_k\}$. By definition $H \,.\, p_k = Hp_k$. If $p_k \in P \subset H$ then $Hp_k = H$ else if $p_k \in \overline{P}$ then $Hp_k = Hp_0$. In both cases $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} p_k b^{\alpha_k} = Q \setminus \{H\}$. If $\alpha_m \neq 0$, then $Q \,.\, w = Q \setminus H$, else $Q \,.\, w = Q \setminus \{H \,.\, p_m\}$. Thus the word w is not 2-compressing with respect to the 2-compressible automaton $\mathcal{A}_H$, a contradiction.

Now consider the case (b) in which, we recall, $P = S$ and the subgroup $H = H_P = \langle S\rangle$ has index 2 in $FG(\Pi)$. Let $H' = FG(\Pi) \setminus H$; then $\{Hw \mid w \in \Pi^*\} = \{H, H'\}$. We define an automaton $\mathcal{B}_H = \langle Q, \Sigma, \delta\rangle$ with the state set $Q = \{H, H', q\}$, where q is a new symbol. The transition function δ is defined as follows:

$q \,.\, a = q$ for all $a \in \Sigma$,
$Hp \,.\, a = Hpa$ for all $a \in \Pi$,
$H' \,.\, b = H'$
$H \,.\, b = q$.

The automaton $\mathcal{B}_H$ is 2-compressible. Indeed, since $H \subsetneqq FG(\Pi)$ there exists a letter $a \in \Pi$ such that $H'a = H$, therefore $\mathrm{df}(bab) \geq 2$.

As in the case (a) we show that $\mathrm{df}(w) = 1$. The induction basis is the same as in (a).

Induction step. Assume that $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} = Q \setminus \{H\}$ for some k. Then $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} p_k = Q \setminus \{H \,.\, p_k\}$. By definition $H \,.\, p_k = Hp_k$, in its turn, $Hp_k = H$ since $p_k \in P = S \subset H$, whence $Q \,.\, b^{\alpha_0} p_1 b^{\alpha_1} p_2 \cdots b^{\alpha_{k-1}} p_k b^{\alpha_k} = Q \setminus \{H\}$.

Thus we again have found a 2-compressible automaton with respect to which the word w is not 2-compressing. This completes the proof.

Now consider the special case when $|\Sigma| = 2$ as it will be essential for the proof of the main result in Section 2. For such Σ there are only two possible choices of the 'distinguished' letter, and the group $FG(\Pi)$ is isomorphic to the group $\langle \mathbb{Z}, + \rangle$ of integers. Therefore the inner segments of any word $w \in \Sigma^*$ are in fact powers of a letter, and so we can consider the set of exponents of these powers instead of the set of segments. Thus, representing any word w over the alphabet $\Sigma = \{a, b\}$ in the form

$$w = a^{u_0} b^{p_1} a^{u_1} \ldots b^{p_{m-1}} a^{u_{m-1}} b^{p_m}$$

with $u_0, p_m \geq 0;\ u_i, p_i > 0,\ i = 1, \ldots, m-1;\ m \geq 0$, we assign to it two sets of positive integers:

$$S_a = \{u_i \mid p_i \cdot p_{i+1} \neq 0\}, \quad S_b = \{p_i \mid u_{i-1} \cdot u_i \neq 0\}.$$

Let S be a fixed set of integers. Given a subset $P \subseteq S$, we put $\overline{P} = S \setminus P$ and define H_P to be the subgroup of $\langle \mathbb{Z}, + \rangle$ generated by the set $P \cup (\overline{P} - \overline{P})$.

Proposition 3. *Let $\Sigma = \{a, b\}$. A word $w \in \Sigma^+$ is 2-collapsing if and only if it satisfies the following two conditions:*

(i) $\langle S_a \rangle = \langle S_b \rangle = \mathbb{Z}$;
(ii) *for all subsets $P_a \subsetneqq S_a$ and $P_b \subsetneqq S_b$, each of the subgroups H_{P_a} and H_{P_b} has index at most 2 in $\mathbb{Z}$.*

Proof. The 'only if' part follows directly from Proposition 2. Hence it remains to prove the 'if' part. It is easy to see that if a word fulfills (i) and (ii) then it is 2-full.

Consider an arbitrary 2-compressible automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$. We note that if there exists a word $v \in \Sigma^+$ of length not greater than 2 such that $\mathrm{df}(v) \geq 2$ then also $\mathrm{df}(w) \geq 2$. If for any word v of length not greater than 2 holds $\mathrm{df}(v) < 2$ then for any letter $x \in \Sigma$ either

$$Q \,.\, x = Q, \tag{3}$$

i.e. x acts as a permutation on the state set Q, or

$$|Q\,.\,x| = |Q| - 1 \text{ and } Q\,.\,x^2 = Q\,.\,x. \tag{4}$$

It is clear that since $\mathcal{A}$ is 2-compressible, there exists at least one letter in Σ satisfying (4). Besides that, by [3, Lemma 2.5] we may assume that there exists at least one letter in Σ satisfying (3). Since the conditions (i) and (ii) are symmetric with respect to a and b, we may assume with no loss that the letter a satisfies (3) while the letter b satisfies (4).

By e we denote *the exception state* of the letter b, i.e. the only state in $Q\setminus Q\,.\,b$. Since b acts as a permutation on the set $Q\,.\,b$ there exists a unique state $d \in Q$ such that $e\,.\,b = d\,.\,b$. We note that

$$\mathrm{df}(ba^\alpha b) \geq 2 \text{ if and only if } e\,.\,a^\alpha \notin \{e, d\}. \tag{5}$$

Indeed, the condition $\mathrm{df}(ba^\alpha b) \geq 2$ means $e, d \in Q\,.(ba^\alpha) = (Q\,.\,b)\,.\,a^\alpha$. Since $Q = \{e\} \cup Q\,.\,b$ and a^α is a permutation on Q we have

$$Q = \{e\,.\,a^\alpha\} \cup (Q\,.\,b)\,.\,a^\alpha.$$

Thus $e, d \in (Q\,.\,b)\,.\,a^\alpha$ if and only if $e\,.\,a^\alpha \notin \{e, d\}$.

Consider the orbit $E = \{e\,.\,a^\alpha \mid \alpha \in \mathbb{Z}\}$ of e and the stabilizer $St(e) = \{\alpha \in \mathbb{Z} \mid e\,.\,a^\alpha = e\}$ of e under the action of $\mathbb{Z}$ on Q defined via the action of the letter a. Clearly, the index of $St(e)$ in $\mathbb{Z}$ is equal to $|E|$.

Let $P_a = St(e) \cap S_a$ and consider $\alpha \in \overline{P_a} = S_a \setminus P_a$. By definition $e\,.\,a^\alpha \neq e$ and, if besides that $e\,.\,a^\alpha \neq d$, then by (5) we get $\mathrm{df}(ba^\alpha b) \geq 2$ whence $\mathrm{df}(w) \geq 2$. Now assume that $e\,.\,a^\alpha = e\,.\,a^\beta$ for all $\alpha, \beta \in \overline{P_a}$. This means that $\alpha - \beta \in St(e)$, i.e. all generators of the subgroup H_{P_a} belong to $St(e)$ whence $H_{P_a} \leq St(e)$. In view of the conditions (i), (ii) this implies that either $St(e) = \mathbb{Z}$ or $St(e)$ has index 2 in $\mathbb{Z}$ and $P_a \neq S_a$. In the first case $\mathcal{A}$ cannot be 2-compressible. In the second case $E = \{e, f\}$, $f \neq e$. If $f = d$, then again $\mathcal{A}$ is not 2-compressible by (5) whence $f \neq d$. Let $\alpha_0 \in \overline{P_a}$, then $e\,.\,a^{\alpha_0} \neq e$, i.e. $e\,.\,a^{\alpha_0} = f \neq d$ whence by (5) we obtain that $\mathrm{df}(ba^{\alpha_0} b) \geq 2$ and $\mathrm{df}(w) \geq 2$.

2 The Location of $\mathcal{C}_2$ in the Chomsky Hierarchy

In order to prove that the language $\mathcal{C}_2$ is not context-free we use the well-known lemma by Ogden [10], see also [6, Lemma 2.5].

Lemma 1 (Ogden). *Given a context-free language $\mathcal{L}$, there exists a positive integer N such that any word $w \in \mathcal{L}$ of length at least N admits a factorization $w = xuyvz$ such that*

1) *at least one of the factors u and v is non-empty;*
2) *the length of the factor uyv does not exceed N;*
3) *$xu^k yv^k z \in \mathcal{L}$ for each positive integer k.*

Theorem 1. *The language $\mathcal{C}_2$ of all 2-collapsing words over the alphabet $\Sigma = \{a, b\}$ is not context-free.*

Proof. Arguing by contradiction, suppose that $\mathcal{C}_2$ is a context-free language. Then its intersection with the regular language $\mathcal{R} = a(bb)^*ba(bb)^*ba^2(bb)^*ba$ is also context-free. By Proposition 3, the intersection consists of all words $ab^{p_1}ab^{p_2}a^2b^{p_3}a$ such that the numbers p_1, p_2, p_3 are odd, and the set $S_b = \{p_1, p_2, p_3\}$ satisfies the conditions of the proposition. (It is easy to see that the set $S_a = \{1, 2\}$ satisfies the conditions of Proposition 3.) This gives us the following bunch of restrictions on the numbers p_1, p_2, p_3:

1. $\mathrm{g.c.d.}(p_1,\, p_2) = 1$;
2. $\mathrm{g.c.d.}(p_2,\, p_3) = 1$;
3. $\mathrm{g.c.d.}(p_1,\, p_3) = 1$;
4. $\mathrm{g.c.d.}(p_1,\, p_2 - p_3) = 1$;
5. $\mathrm{g.c.d.}(p_2,\, p_1 - p_3) = 1$;
6. $\mathrm{g.c.d.}(p_3,\, p_2 - p_1) = 1$;
7. $\mathrm{g.c.d.}(p_2 - p_1,\, p_3 - p_1) = 2$.

Let N be the constant from Ogden's lemma applied to the context-free language $\mathcal{C}_2 \cap \mathcal{R}$. Let M be an even number such that $M > N$ and consider the word

$$w = ab^{p_1}ab^{p_2}a^2b^{p_3}a \in \mathcal{R}$$

with $p_1 = M! - 1,\ p_2 = M! + 1,\ p_3 = (M+1)! - 1$. We aim to show that w is in $\mathcal{C}_2 \cap \mathcal{R}$. Indeed, let us check that the chosen numbers p_1, p_2, p_3 satisfy the restrictions 1–7.

1. Let $d = \mathrm{g.c.d.}(M! - 1,\, M! + 1)$. Obviously d is odd and divides the difference $M! + 1 - (M! - 1) = 2$, consequently $d = 1$.

2. In the same way, if $d = \mathrm{g.c.d.}(M! + 1,\, (M+1)! - 1)$ then d divides the sum

$$M! + 1 + (M+1)! - 1 = M!(M+2) = 2M!\frac{(M+2)}{2}.$$

Here the number $\dfrac{M+2}{2} < M$ is an integer since M is even. Therefore d is a product of integers less than or equal to M, but $M!+1$ is coprime with any such integer whence we obtain $d = 1$ also in this case.

3. $\mathrm{g.c.d.}(M! - 1,\, (M+1)! - 1) = 1$ by the same reason as above since the difference $(M+1)! - 1 - (M! - 1) = M!M$ is a product of integers less than or equal to M.

4. We obtain

$$\mathrm{g.c.d.}(M! - 1,\, (M+1)! - 1 - (M! + 1)) = \mathrm{g.c.d.}(M! - 1,\, M!M - 2) = 1$$

in the same way as in the case 2 since $M!M - 2 - 2(M! - 1) = M!(M-2)$.

5. $\mathrm{g.c.d.}(M! + 1,\, (M+1)! - 1 - (M! - 1)) = \mathrm{g.c.d.}(M! + 1,\, MM!) = 1$.

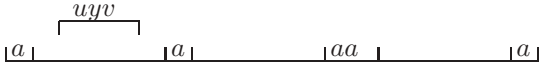
6. $\mathrm{g.c.d.}((M+1)! - 1,\, M! + 1 - (M! - 1)) = \mathrm{g.c.d.}((M+1)! - 1,\, 2) = 1$.

7. $\mathrm{g.c.d.}(M! + 1 - (M! - 1),\, (M+1)! - 1 - (M! - 1)) =$
$\mathrm{g.c.d.}(2,\, MM!) = 2$.

Thus we have proved that $w \in \mathcal{C}_2 \cap \mathcal{R}$. It is clear that $|w| > N$ whence Ogden's lemma applies yielding a factorization $w = xuyvz$ with the properties

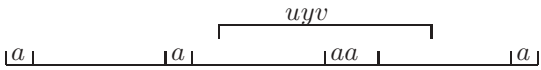
1)–3). In particular, the word xu^kyv^kz is in $\mathcal{C}_2 \cap \mathcal{R}$ for any positive integer k. Since the length of the factor uyv does not exceed N, this factor cannot overlap all three blocks of b's in the word w because $N < M < p_2 = M! + 1$. Therefore we have to analyze two cases.

Case 1. The entire factor uyv fits into one of the blocks b^{p_i}.



In this case $u = b^i,\ v = b^j, y = b^s$, where $0 < r = i + j < N$. Observe that r is coprime with each of the numbers $p_1 = M! - 1,\ p_2 = M! + 1,\ p_3 = (M + 1)! - 1$ because $r < M$. For certainty, let uyv be situated inside the block b^{p_1}. Then the set S_b for the word xu^kyv^kz consists of the numbers $p_1 + (k - 1)r, p_2, p_3$. Since $\text{g. c. d.}(r,\ p_2) = 1$, there exists a positive integer k_0 such that p_2 divides $p_1 + (k_0 - 1)r$. Then $\text{g. c. d.}(p_1 + (k_0 - 1)r,\ p_2) = p_2 > 2$ whence the restriction 1 fails for the numbers $p_1 + (k_0 - 1)r, p_2, p_3$. Therefore the word $xu^{k_0}yv^{k_0}z$ does not belong to the language $\mathcal{C}_2 \cap \mathcal{R}$, a contradiction. Clearly, the same reasoning applies in the cases where uyv is situated inside one of the blocks b^{p_2} or b^{p_3}.

Case 2. The factor uyv overlaps two adjacent blocks of b's.



If one of the factors u or v contains an occurrence of the letter a, then the word xu^kyv^kz with $k > 1$ contains more than five occurrences of a and does not belong to the language $\mathcal{R}$. Thus, we have to study the situation when $u = b^i,\ v = b^j, y = b^sab^t$ or $y = b^sa^2b^t$ with $0 < i + j < N$. Observe that at least one of the numbers i and j is positive and hence coprime with each of $p_1 = M! - 1,\ p_2 = M! + 1,\ p_3 = (M + 1)! - 1$.

Now assume that $y = b^sab^t$. Then the set S_b for the word xu^kyv^kz consists of the numbers $p_1 + (k - 1)i, p_2 + (k - 1)j, p_3$. Suppose for certainty that $i \neq 0$. Since $\text{g. c. d.}(i, p_3) = 1$, there exists a positive integer k_0 such that p_3 divides $p_1 + (k_0 - 1)i$. Then $\text{g. c. d.}(p_1 + (k_0 - 1)i,\ p_3) = p_3 > 2$ whence the restriction 3 fails for the numbers $p_1 + (k_0 - 1)i, p_2 + (k_0 - 1)j, p_3$. We see that $xu^{k_0}yv^{k_0}z \notin \mathcal{C}_2 \cap \mathcal{R}$, a contradiction.

If $y = b^sa^2b^t$, then the set S_b consists of $p_1, p_2 + (k - 1)i, p_3 + (k - 1)j$, and a completely analogous argument yields the same contradiction.

We thus see that the language $\mathcal{C}_2 \cap \mathcal{R}$ is not context-free, and hence neither is the language $\mathcal{C}_2$.

In view of Theorem 1 we conjecture that also the languages formed by 2-collapsing words over larger alphabets are not context-free.

3 A Lower Bound for the Length of 2-Collapsing Words

In this section we use some classical facts from the theory of free groups. They all can be found for instance in [7, Chapter 1].

Recall that the *rank* of a free group F (denoted $\mathrm{rank}(F)$) is the number of its free generators. The notion of rank can be also applied to subgroups of free groups as any subgroup of a free group is free.

If F is a free group with $\mathrm{rank}(F) > 1$ and H is its subgroup of finite index $|F : H|$, then H has finite rank and the ranks of F and H are related by Schreier's formula

$$|F : H| = \frac{\mathrm{rank}(H) - 1}{\mathrm{rank}(F) - 1}. \tag{6}$$

Let w be a word over an alphabet Π. By $|w|_a$ we denote the number of occurrences of the letter $a \in \Pi$ in w. Clearly, the mapping $w \mapsto |w|_a$ extends to a homomorphism of the free group $FG(\Pi)$ onto the group $\langle \mathbb{Z}, + \rangle$. We fix an ordering of Π and assign to each element $g \in FG(\Pi)$ a $|\Pi|$-dimensional vector $(g) \in \mathbb{Z}^{|\Pi|}$ whose components are the numbers $|g|_a$, where a runs over Π. The mapping $g \mapsto (g)$ is then a homomorphism of $FG(\Pi)$ onto the free Abelian group $\langle \mathbb{Z}^{|\Pi|}, + \rangle$. Therefore, if a letter $a \in \Pi$ can be represented in the free group $FG(\Pi)$ as a group word in some elements $g_1, g_2, \ldots, g_m \in FG(\Pi)$, then

$$(a) = n_1(g_1) + n_2(g_2) + \cdots + n_m(g_m)$$

for some integers $n_1, n_2, \ldots, n_m$. In particular, the equality

$$1 = n_1|g_1|_a + n_2|g_2|_a + \cdots + n_m|g_m|_a$$

must hold true.

Lemma 2. *If w is a 2-collapsing word over an alphabet Σ, $|\Sigma| > 2$, then for an arbitrary choice of a letter $b \in \Sigma$, the set S of all inner segments of the decomposition (2) of the word w has at least $2(|\Sigma| - 1)$ elements.*

Proof. Let $\Pi = \Sigma \setminus \{b\}$. The set $S = \{p_1, \ldots, p_m\} \subset \Pi^+$ of inner segments of the word w must satisfy the condition (ii) of Proposition 2. In particular, for each $P \subset S$, the subgroup $H_P = \left\langle P \cup \overline{P} \cdot \overline{P}^{-1} \right\rangle$ of the group $FG(\Pi)$ has finite index in $FG(\Pi)$. Observe that if $P \neq S$, then the rank of H_P is not greater than $|P| + |\overline{P}| - 1 = |S| - 1 = m - 1$. Indeed, if one fixes an element $p_{i_0} \in \overline{P}$, then the subgroup H_P is easily seen to be generated by the set $P \cup \{p_{i_0} p_j^{-1} \mid p_j \in \overline{P} \setminus \{p_{i_0}\}\}$.

Now arguing by contradiction suppose that $m < 2|\Pi|$. Then from Schreier's formula (6) we get

$$|FG(\Pi) : H_P| = \frac{\mathrm{rank}(H_P) - 1}{|\Pi| - 1} \leq \frac{m - 2}{|\Pi| - 1} < \frac{2|\Pi| - 2}{|\Pi| - 1} = 2,$$

whence $H_P = FG(\Pi)$ for any $P \subset S$. By the condition (i) of Proposition 2 the set S also generates $FG(\Pi)$ so that $H_P = FG(\Pi)$ also for $P = S$.

Now fix a letter $a \in \Pi$ and consider the set $P = \{p_i \in S \mid |p_i|_a \text{ is even}\}$. Then $\overline{P} = S \setminus P = \{p_j \mid |p_j|_a \text{ is odd}\}$. Since $H_P = FG(\Pi)$, the letter a can be

expressed as a group word in $P \cup \overline{P} \cdot \overline{P}^{-1}$. As observed before the formulation of the lemma, this implies the equality

$$1 = \sum_{p_i \in P} n_i |p_i|_a + \sum_{p_j, p_k \in \overline{P}} n_{jk}(|p_j|_a - |p_k|_a) \tag{7}$$

for some integers n_i, n_{jk}. However, by the choice of P the right hand side of (7) is an even integer, a contradiction. Thus $m \geq 2|\Pi| = 2(|\Sigma| - 1)$.

In a certain sense, the obtained lower bound on the number of inner segments of a 2-collapsing words is precise. Indeed, consider the following 2-collapsing word over the 3-letter alphabet $\Sigma = \{a, b, c\}$:

$$w_{21} = aba^2 \cdot c^2 \cdot bab^2a \cdot c \cdot bab \cdot c \cdot a \cdot c \cdot b \cdot c \cdot b. \tag{8}$$

(It was constructed by Petrov, cf. [4].) Inspecting (8), one observes that choosing c as the distinguished letter yields exactly $4 = 2(|\Sigma| - 1)$ inner segments. On the other hand, other choices of distinguished letters in w_{21} produce more inner segments: 5 for a and 6 for b. We are not aware of any example of a 2-collapsing word over some alphabet Σ such that for any choice of a letter in Σ the set of all inner segments of the word would consists of precisely $2(|\Sigma| - 1)$ elements, however, none of the known results exclude the possibility that such a word may exist.

Combining Lemma 2 with some known facts leads to the main result of this section:

Theorem 2. *The length of an arbitrary 2-collapsing word w over an alphabet Σ with t letters is at least $2t^2$.*

Proof. It is obvious that the length of any 2-collapsing word over the singleton alphabet is at least 2, and it is observed in [12] that the length of any 2-collapsing word over the 2-letter alphabet is at least 8. Thus, the theorem holds true for $|\Sigma| \leq 2$. Now let $|\Sigma| > 2$. By Lemma 2, for any choice of a letter $b \in \Sigma$, the set of all inner segments of the word w with respect to this letter has at least $2(|\Sigma| - 1) = 2(t - 1)$ elements. By definition of an inner segment, each such segment immediately follows an occurrence of the letter b and immediately precedes such an occurrence. Therefore b occurs in w at least $2(t-1)+1 = 2t-1$ times. By Proposition 1 the word w is 2-full whence in particular the factor b^2 should occur in w. This gives at least one additional occurrence of b in w. Altogether, b occurs in w at least $2t - 1 + 1 = 2t$ times. Since b was chosen in an arbitrary way, each of t letters of Σ occurs in w at least $2t$ times whence the length of w is at least $2t \cdot t = 2t^2$.

The lower bound of Theorem 2 is precise for $t = 1, 2$, and for $t = 3$ it gives the value 18 that is quite close to the exact value of the minimum length of a 2-collapsing word over 3 letters (recall that this exact value is 21, see [4]). It is very interesting to see what happens in the case $t = 4$ in which the gap between the value 32 of the lower bound provided by Theorem 2 and the value 58 of the

best upper bound known so far (found by Martjugin, unpublished) still remains rather large.

We have found our new lower bound by studying the number of inner segments of a 2-collapsing word. Perhaps it is worth mentioning that the approach based on purely quantitative properties of inner segments (such as their number, their lengths, etc) is not yet powerful enough to characterize 2-collapsing words over alphabets with more than 2 letters, and thus, we doubt that it alone might yield the precise value of the minimum length of such words. In order to justify our claim, consider the mirror image $\overleftarrow{w}_{21}$ of the word w_{21} from (8). Of course, the inner segments of $\overleftarrow{w}_{21}$ are nothing but the mirror images of the ones of w_{21}, and therefore, they share all possible numeric characteristics. Nevertheless in contrast to w_{21} the word $\overleftarrow{w}_{21}$ is not 2-collapsing. Indeed, it can be easily verified that $\overleftarrow{w}_{21}$ is not 2-compressing with respect to the automaton on Fig. 1, which is in fact 2-compressible (for example, $\mathrm{df}(cbac) = 2$).

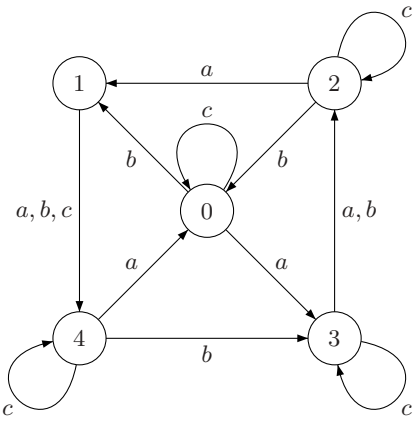


Fig. 1. A 2-compressible automaton with $\mathrm{df}(\overleftarrow{w}_{21}) = 1$

Acknowledgment

The author is grateful to Prof. M. V. Volkov for his guidance and to Prof. Jorge Almeida for a number of useful remarks.

References

1. J. Almeida, M. V. Volkov. *Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety*, J. Algebra Appl. **2** (2003) 137–163.
2. D. S. Ananichev, A. Cherubini, M. V. Volkov. *An inverse automata algorithm for recognizing 2-collapsing words*, in M. Ito, M. Toyama (eds.). Developments in Language Theory. [Lecture Notes in Computer Science, **2450**] Springer, Berlin, 2003, 270–282.
3. D. S. Ananichev, A. Cherubini, M. V. Volkov. *Image reducing words and subgroups of free groups*, Theor. Comp. Sci. **307** (2003) 77–92.

4. D. S. Ananichev, I. V. Petrov. *Quest for short synchronizing words and short collapsing words*, WORDS. Proc. 4th Int. Conf., Univ. of Turku, Turku, 2003, 411–418.
5. D. S. Ananichev, M. V. Volkov. *Collapsing words vs. synchronizing words*, in W. Kuich, G. Rozenberg, A. Salomaa (eds.). Developments in Language Theory. [Lecture Notes in Computer Science, **2295**] Springer, Berlin, 2002, 166–174.
6. J. Berstel. Transductions and Context-free Languages, Teubner, Stuttgart, 1969.
7. R. Lyndon, P. Schupp. Combinatorial Group Theory, Springer, Berlin, 1977.
8. M. Hall, *A topology for free groups and related groups*, Ann. Math. **52** (1950) 127–139.
9. S. W. Margolis, J.-E. Pin, M. V. Volkov. *Words guaranteeing minimum image*, Int. J. Foundations Comp. Sci. **15** (2004) 259–276.
10. W. Ogden. *A helpful result for proving inherent ambiguity*, Math. Systems Theory **2** (1968) 191–194.
11. R. Pöschel, M. V. Sapir, N. Sauer, M. G. Stone, M. V. Volkov, *Identities in full transformation semigroups*, Algebra Universalis **31** (1994) 580–588.
12. N.Sauer and M.G. Stone, *Composing functions to reduce image size*, Ars Combinatoria **31** (1991) 171–176.

Semi-rational Sets of DAGs

Lutz Priese

Fachbereich Informatik
Universität Koblenz-Landau, Germany
priese@uni-koblenz.de

Abstract. We call a set of DAGs (directed acyclic graphs) *semi-rational* if it is accepted by a Petri net. It is shown that the class of semi-rational sets of DAGs coincides with the synchronization closure of Courcelles class of recognizable sets of unranked, unordered trees (or forests).

1 Introduction

For sets of words various concepts proved to be equivalent: recognizability (invers homomorphisms of finite monoids, or congruences of finite index), rationality (acceptance by finite automata), definability by regular expressions, definability by monadic second-order logics over certain signatures, generation by right-linear grammars, etc. These correspondences could be transferred to languages of infinite words and finite and infinite trees. However, a generalization to DAGs (directed acyclic graphs) has failed. Even for such simple sub-classes like finite ladders or grids it turned out that planar ladders cannot be accepted by grid automata and the emptiness problem for finite grid automata is undecidable [1]. Courcelle [2], [3] has defined a multi-sorted algebra for finite graphs, however with infinitely many sorts. As a consequence, even some non-recursive sets of square grids are recognizable in the sense of [3]. For finite grids this concept of recognizability is stronger than that of monadic second-order definability which is stronger than that of acceptability with finite-state graph automata [1] which is again stronger than acceptability by automata over planar DAGs [4], see [1].

We will study here semi-rational sets of DAGs: A set L of words over some alphabet Σ is rational if there exists a finite automaton A s.t. $L = \{w \in \Sigma^* \mid \exists$ run r in A from the initial state to a terminal state s.t. w is the labelling of $r\}$. In analogy, we call a set L of DAGs over Σ *semi-rational* if there exists a Petri net $\mathcal{N}$ s.t. $L = \{\alpha \mid \alpha$ is a DAG over Σ and $\exists$ a process π in $\mathcal{N}$ from the initial marking to a terminal marking s.t. α is the labelling of $\pi\}$. The emptiness problem of semi-rational sets of DAGs is decidable: the DAG language of a terminal Petri net is non-empty if and only if the word language of this Petri net is non-empty if and only if at least one of the finitely many terminal markings is reachable from the initial marking.

A graph is called *ranked* if the label of a node determines its number of sons, and is called *ordered* if there exists a total ordering between all sons of any node. The theory of ranked, ordered trees goes back to the early works of Church, Büchi, Rabin, Doner, Thatcher, Trakhtenbrot. A good overview is given

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 385–396, 2005.
© Springer-Verlag Berlin Heidelberg 2005

in the TATA book project [5]. A ranked and ordered tree is usually regarded as a correct term over a ranked alphabet. A tree automaton for binary trees may be regarded as a finite set of Horn formulas $p(fxy) \leftarrow q(x) \wedge r(y)$, $p(a) \leftarrow true$, telling that tree fxy is accepted in state p if sub trees x and y are accepted in states q and r, respectively, and leaf a is accepted in state p, see [6].

Most automata theoretical results transfer also to ordered, unranked trees, see the well-known report [7]. Courcelle [8] has introduced many-sorted algebras (magmas in his terminology) with equations for unordered and unranked trees and forests and gave a precise definition of recognizability of sets of those trees and forests. As a theory of unranked, unordered trees is not very well developed we present a definition of their recognizability in the following chapter. We also introduce a rather smooth concept of regular grammars for unranked and unordered trees and forests that operates with empty (i.e, ε-labelled) nodes. The equivalence of recognizability and regularity is shown in an Appendix. In chapter 3 we introduce a new version of synchronization of graphs. In chapter 4 Petri net processes and their DAG semantics are introduced. We prove that semi-rational sets of DAGs coincide with the synchronization closure of recognizable sets of unranked, unordered trees. As an intermediate step we use merge grammars that are a special case of the graph structure grammars of Starke [9].

2 Recognizable and Regular Sets of Unordered, Unranked Trees

2.1 $T_{u,u}$-Magmas

Σ denotes a finite, non-empty alphabet of *labels*. A formal definition for a graph in set theoretical terms is simple: A *graph* γ over Σ is a triple $\gamma = (N, E, \lambda)$ of two finite sets N of *nodes* and $E \subseteq N \times N$ of *edges* and a labelling mapping $\lambda : N \to \Sigma$. Two graphs $\gamma_i = (N_i, E_i.\lambda_i)$ are *isomorphic* if there exists a bijective mapping $h : N_1 \to N_2$ with $(v, v') \in E_1 \Leftrightarrow (h(v), h(v')) \in E_2$ and $\lambda_2(h(v)) = \lambda_1(v)$ holds for all v, v' in N_1. We identify isomorphic graphs and thus deal with abstract graphs. Directed acyclic graphs (DAGs) and trees are defined as usual as special types of graphs. In this paper, the terms tree, forest, DAG, and graph always refer to finite, directed, node labelled, unranked and unordered trees, forests, DAGs, or graphs. A forest is a graph consisting of a family of trees. A pomset is a transitive closed DAG. $\Sigma^{\mathcal{T}_{u,u}}, \Sigma^{\mathcal{F}_{u,u}}, \Sigma^{\mathcal{D}_{u,u}}$ and $\Sigma^{\mathcal{G}_{u,u}}$ denote the classes of all trees, forests, dags and graphs over Σ. The sub-script u, u reminds to 'unranked' and 'unordered'. $\Sigma^{\mathcal{P}}$ is the class of all pomsets over Σ. As pomsets are never ordered or ranked we drop the sub-script u, u. A multi-set m over some set M is a mapping $m : M \to \mathbb{N}$. m is often denoted by $m = \sum_{a \in M} m(a) \cdot a$. The *size* $|m|$ of m is $|m| := \sum_{a \in M} m(a)$. 0 denotes the empty multi-set m with $m(a) = 0\ \forall a \in M$.

Courcelle [8] has introduced $T_{u,u}$-magmas to define recognizability for trees and forests. The signature Θ^{Σ} is the tuple $\Theta^{\Sigma} := (S, Op^{\Sigma})$ where S consists of the two sorts tree and forest , and Op^{Σ} consists of the operations a of profile forest $\to$ tree (for each $a \in \Sigma$), ι of profile tree $\to$ forest , $+$ of profile

forest $\times$ forest $\rightarrow$ forest , and ϕ of profile $\rightarrow$ forest . T^t and T^f denote all ground terms of sort tree and forest in Θ^{Σ}. A $T_{u,u}$-magma $\mathbb{M}$ has the signature Θ^{Σ} and is defined as $\mathbb{M} = (M^t, M^f, \{a_{\mathbb{M}}\}_{a\in\Sigma}, \iota_{\mathbb{M}}, +_{\mathbb{M}}, \phi_{\mathbb{M}})$ of sets M^t of objects of sort tree , M^f of objects of sort forest , and total mappings $a_{\mathbb{M}} : M^f \rightarrow M^t$, $\iota_{\mathbb{M}} : M^t \rightarrow M^f$, $+_{\mathbb{M}} : M^f \times M^f \rightarrow M^f$, $\phi_{\mathbb{M}} :\rightarrow M^f$. $\mathbb{M}_i^{\Sigma} := (T^t, T^f, \{a\}_{a\in\Sigma}, \iota, +, \phi)$ is the initial $T_{u,u}$-magma. One must adjoin to a $T_{u,u}$-magma the equations $+(x_1, x_2) = +(x_2, x_1)$, $+(x_1, +(x_2, x_3)) = +(+(x_1, x_2), x_3)$, $+(x_1, \phi) = x_1$ with nullary symbols (variables) x_i of sort forest , forcing $+$ to be commutative and associative with ϕ as a null-element. This ensures unranked and unordered objects. Let $\sim$ be the congruence on $T^t \cup T^f$ induced by those equations. One can identify a tree (forest) over Σ with a term of sort tree (forest , respectively) in $\mathbb{M}_i^{\Sigma}/\sim$, using the following standard interpretation: ϕ denotes the empty forest, $+$ is the union of multi-sets, $\iota(\alpha)$ regards the tree α as the forest $1\cdot\alpha$, consisting of the only tree α. $a(m)$ for $a \in \Sigma$ is the tree with the root labelled with a and all sons of this root are exactly the roots of all trees in the forest m. The derived forest of Figure 2 is represented by any term congruent to $\iota(a(\iota(a(\phi)) + \iota(b(\phi)) + \iota(b(\iota(a(\phi))))) + \iota(a(\iota(a(\phi)) + \iota(b(\iota(b(\phi)) + \iota(a(\phi))))))$.

A set A of trees (or forests, respectively) over Σ is *recognizable* if there exist a finite $T_{u,u}$-magma $\mathbb{M}_A = (M_A^t, M_A^f, \{a_A\}_{a\in\Sigma}, \iota_A, +_A, \phi_A)$, and a subset B of M_A^t (of M^f, respectively) s.t. $A = i^{-1}(B)$ holds for the initial homomorphism $i : \mathbb{M}_i \rightarrow \mathbb{M}_A$. $Rec^{\mathcal{T}_{u,u}}$ and $Rec^{\mathcal{F}_{u,u}}$ denote the classes of all recognizable languages of trees and forests.

In [8] Courcelle has introduced labels as a third sort plus an operation p of profile label $\times$ forest $\rightarrow$ tree . However, using the labels instead as unary operations allows us to regard $\mathbb{M}_i^{\Sigma}$ as an initial algebra and to define recognizability via the initial homomorphism.

2.2 Regular Graph Grammars with ε-Rules

ε is here the symbol for the empty word, empty graph, and also the 'empty label'. We assume that ε is not a symbol in the alphabet Σ and define $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$. A graph over Σ_ε thus may possess nodes labelled with ε (ε-*nodes*). Let α' result from a graph α over Σ_ε by removing one ε-node v and declaring all sons of v in α to be in α' the sons of each father of v in α. In this case α' is called a *reduct* of α. The ε-*equivalence* of graphs over Σ_ε is the reflexive, symmetric and transitive closure of the reduct-relation. Figure 1 presents some ε-equivalent DAGs. Note, reducing a tree over Σ_ε may result in a forest over Σ.

A *regular forest grammar* (*rfg*) G is a tuple $G = (V, \Sigma, S, R)$ of a finite set V of *variables*, a finite set Σ of *terminals*, s.t. $V \cap \Sigma = \emptyset$ and $\varepsilon \notin \Sigma \cup V$, a *start* variable $S \in V$, and a finite set $R \subseteq V \times ((\Sigma \cup \{\varepsilon\}) \times \mathbb{N}^V)$ of *rules*. A rule $r = (X, (a, m))$ with $X \in V, a \in \Sigma \cup \{\varepsilon\}$, m a multi-set over V, is also denoted by $r = X \rightarrow a(m)$, or as $X \rightarrow a$ if m equals the empty multi-set 0. X is the *premise* and $a(m)$ the *conclusion* of r. A conclusion $a(m)$ will be interpreted as in $\mathbb{M}_i^{\Sigma}/\sim$ as the tree of depth 1 with the root labelled with a and with the multi-set m as leaves. A rule $X \rightarrow a(m)$ applies to a graph γ only on leaves: choose in γ one leaf labelled with X and substitute it by the tree $a(m)$ in the

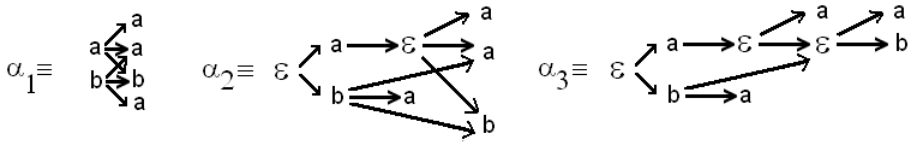


Fig. 1. Three ε-equivalent dags. α_2 is a reduct of α_3 and α_1 is a reduct of a reduct of α_2

obvious way. For a rule $r = X \to a(m)$ and two graphs γ, γ' we write $\gamma \Rightarrow_r \gamma'$ if γ' results from γ by applying rule r. $\Rightarrow_G$ and $\Rightarrow^*$, etc., are used as usual.

$L(G) := \{\gamma \in \Sigma^{\mathcal{G}_{u,u}} \mid S \Rightarrow^* \gamma\}$ is the set of graphs generated by G. A set of forests is called *regular* if it is generated by some *rfg*. $Reg^{\mathcal{F}_{u,u}}$ ($Reg^{\mathcal{F}_{u,u},\varepsilon-free}$, respectively) denote the class of all regular sets of forests (that can be generated by ε-free *rfg*s, respectively).

Example 1.

$\Sigma^{\mathcal{F}_{u,u}}$ is itself a regular forest language and is generated by $G_0 = (\{S\}, \Sigma, S, R)$ with the rules $S \to x$, $S \to x(S)$, $S \to x(2 \cdot S)$ for all $x \in \Sigma_\varepsilon$. The use of ε-rules is here essential. A set A of forests is of *unbounded rank* if for any integer k there exists a forest in A with an out-degree larger than k. Obviously, sets of unbounded rank cannot be generated by a regular forest grammars without ε-rules. Figure 2 gives an example of a derivation in G_0. If one drops the rule $S \to \varepsilon(2 \cdot S)$ in G_0 one generates exactly $\Sigma^{\mathcal{T}_{u,u}}$. □

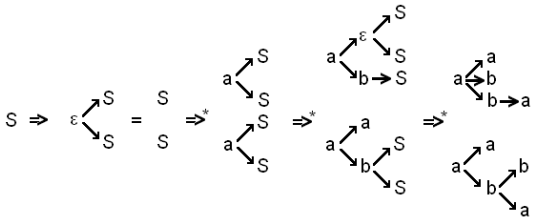


Fig. 2. A derivation in G_0 with $\Sigma = \{a, b\}$

Let $S \Rightarrow^* \gamma$ be a derivation in a *rfg* $G = (V, \Sigma, R, S)$. Then γ must be a forest with labels $V \cup \Sigma$ where only the leaves of γ possess labels in V. A forest with more than one root can only be obtained by 'splitting' the start variable by some derivation $S \Rightarrow^*_G X \Rightarrow \varepsilon(X_1 + X_2 + ...)$.

A *regular tree grammar (rtg)* $G = (V, \Sigma, R, S)$ is a *rfg* where for any rule $S \to \varepsilon(m) \in R$ there holds $|m| \leq 1$ and for any two rules $S \to \varepsilon(X)$, $X \to x(m) \in R$ there holds $x \neq \varepsilon$. A set of trees is called *regular* if it is generated by some *rtg*. $Reg^{\mathcal{T}_{u,u}}$ denotes the class of all regular sets of trees.

2.3 Equivalence of Recognizable and Regular Sets of Trees or Forests

The equivalence of recognizability and regularity is known for languages of words, for ordered, ranked trees, and for ordered, unranked trees. It also holds for unordered, unranked trees.

Theorem 1.

$Rec^{\mathcal{T}_{u,u}} = Reg^{\mathcal{T}_{u,u}}$, $Rec^{\mathcal{F}_{u,u}} = Reg^{\mathcal{F}_{u,u}}$

There exists an Appendix to this paper in the Publications of the Image Recognition Lab (www.uni-koblenz.de/∼lb) in Koblenz. A proof of Theorem 1 is presented there.

3 Operations on Graphs

We need some operations on graphs and DAGs. Let Σ, Δ be two alphabets. A *(very) fine morphism* h from Σ to Δ is a mapping $h : \Sigma \to \Delta_\varepsilon$ ($h : \Sigma \to \Delta$, respectively). $h(\alpha)$ for a graph $\alpha = (N, E, \lambda)$ over Σ and a fine or very fine homomorphism h is the graph $(N, E, h \circ \lambda)$ over Δ. The *shuffle* $\|$ of two graphs α_1, α_2 is the disjoint union of both. For a set M we define the *big shuffle* $\|^* M$ inductively by $\|^0 M := \{\varepsilon\}$, $\|^{n+1} M := M \,\|\, (\|^n M)$, and $\|^* M := \bigcup_{n \geq 0} \|^n M$. Here ε is the empty graph $\varepsilon = (\emptyset, \emptyset, \emptyset)$. For $M \subseteq \Sigma$, $a \in \Sigma_\varepsilon$, and $\alpha \in \Sigma^{\mathcal{G}_{u,u}}$ define $merge[M, a](\alpha)$ as the set of all graphs one can get by merging in α some M-subset into a single node labelled with a. Here an M-subset is simply a set of $|M|$ nodes with M as their multi-set of labels. There are in general different M-subsets that may be merged into a. Thus, $merge[M, a](\alpha)$ may possess non isomorhic graphs.

We introduce a new *(generalized) synchronization* $Syn[M, a] : \Sigma^{\mathcal{G}_{u,u}} \to 2^{\Gamma^{\mathcal{G}_{u,u}}}$ for $M \subseteq \Sigma$, $a \in (\Sigma - M)_\varepsilon$, $\Gamma := (\Sigma - M) \cup \{a\}$ as follows: For a graph α over Σ $Syn[M, a](\alpha)$ consists of all graphs over Γ that result from the algorithm:

$A_{new} := \{\alpha\}$;
repeat until $A_{new} = A_{old}$:
begin $A_{old} := A_{new}$; $A_{new} := A_{old} \cup \bigcup_{\beta \in A_{old}} merge[M, a](\beta)$ end;
$Syn[M, a](\alpha) := A_{new} \cap 2^{\Gamma^{\mathcal{G}_{u,u}}}$.

Thus, one repeatedly replaces in α sets M of labels by a new label a until there is no further label from the set M left. All resulting graphs belong to $Syn[M, a](\alpha)$. One may define this synchronization also for DAGs and pomsets:
$Syn^{\mathcal{D}}[M, a] : \Sigma^{\mathcal{D}_{u,u}} \to 2^{\Gamma^{\mathcal{D}_{u,u}}}$ maps a DAG α into the set $Syn[M, a](\alpha) \cap 2^{\Gamma^{\mathcal{D}_{u,u}}}$, and
$Syn^{\mathcal{P}}[M, a] : \Sigma^{\mathcal{P}} \to 2^{\Gamma^{\mathcal{P}}}$ maps a pomset α into the set of transitive closures of $Syn^{\mathcal{D}}[M, a](\alpha)$.

This synchronization is rather powerful. Even a fine homomorphism is expressible as a synchronization. There is another synchronization operation $syn^{\mathcal{P}}_M$:

$\Sigma^{\mathcal{P}} \times \Sigma^{\mathcal{P}} \to 2^{\Sigma^{\mathcal{P}}}$ known in the theory of Petri nets, which was introduced by Grabowski [10] under the name $M - section$ for pomsets. To apply $syn^{\mathcal{P}}_M$ to two pomsets α_i both pomsets must possess the same number of a-nodes for any $a \in M$, and any a-node of α_1 is merged with some a-node of α_2, keeping the common label a, following the rule that the result must be acyclic. The set of all transitive closures of all acyclic graphs obtainable in this way is the set of pomsets in $\alpha_1\ syn^{\mathcal{P}}_M\ \alpha_2$. It is easily seen that a $syn^{\mathcal{P}}$-operation can be obtained by a multiple application of $Syn^{\mathcal{P}}$-operations. Figure 3 presents two forests α, β with $\alpha\ syn_{\{a\}}\ \beta = \{\gamma, \delta\}$ and $Syn^{\mathcal{D}}[\{b, c\}, x](\delta) = \{\kappa\}$. The directed arcs always point from left to right.

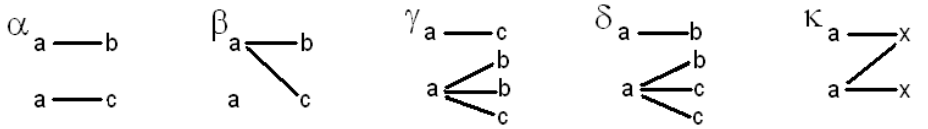


Fig. 3. $\alpha\ syn_{\{a\}}\ \beta = \{\gamma, \delta\}$, $Syn^{\mathcal{D}}[\{b, c\}, x](\delta) = \{\kappa\}$

The following lemma is rather obvious.

Lemma 1. *$Reg^{\mathcal{F}_{u,u}}$ is closed under union, fine homomorphism, inverse very fine homomorphism, shuffle and big shuffle, but not under synchronization $Syn^{\mathcal{D}}$ or $syn^{\mathcal{D}}$.*

$Reg^{\mathcal{F}_{u,u},\varepsilon-free}$ is closed under union, very fine homomorphisms, inverse very fine homomorphisms, but not under synchronization $Syn^{\mathcal{D}}$ or $syn^{\mathcal{D}}$.

$Reg^{\mathcal{T}_{u,u}}$ is closed under union, fine homomorphism, inverse very fine homomorphism, but not under synchronization $Syn^{\mathcal{D}}$, $syn^{\mathcal{D}}$ or shuffle.

4 Semi-rational and Semi-regular Sets of DAGs

4.1 Graph Grammars with Merge

Merge rules in a graph grammar allow to merge several variables into one. This leaves the concept of context-free replacement rules. A *merge graph grammar (mgg)* G is a tuple $G = (V, \Sigma, S, R)$ of V, Σ, S as before and a finite set $R = R^{replace} \cup R^{merge}$ with $R^{replace} \subseteq V \times ((\Sigma \cup \{\varepsilon\}) \times \mathbb{N}^V)$, the set of *replacement* rules, and $R^{merge} \subseteq (V \times V) \times V$, the set of *merge* rules. A merge rule $r = ((X_1, X_2), X)$ is denoted by $r = X_1 + X_2 \to X$. For two graphs α, β we set $\alpha \Rightarrow_r \beta$ for a replacement rule r as before, and for a merge rule $r = X_1 + X_2 \to X$ if β results from α by choosing a leaf v_1 labelled with X_1 and a leaf v_2 labelled with X_2, merging them into one leaf v with the new label X. Thus, all sons of v_1 and v_2 are the sons of v, the fathers of v_1 and v_2 are the fathers of v. We transfer the notations $\Rightarrow^*$ and $L(G)$ canonically to *mgg*s. Obviously, starting from the variable S a *mgg* can generate only DAGs over $V \cup \Sigma$ where all V-nodes must be leaves.

A set L of DAGs is called *semi-regular* if $L = \{\alpha \in \Sigma^{\mathcal{D}_{u,u}} \mid S \Rightarrow_G^* \alpha\}$ for some *mgg* $G = (V, \Sigma, S, R)$. We denote by $\mathcal{D}^{merge}$ the class of all semi-regular DAG languages.

Lemma 2. *$\mathcal{D}^{merge}$ is closed under fine homomorphisms, inverse very fine homomorphisms, union, shuffle and $Syn^{\mathcal{D}}$.*

A sketch of the proof can be found in the Appendix.

Example 2.

Let $L_{a,b}$ be the language of all non-empty DAGs over $\{a, b\}$ where any chain from some root to some leaf is labelled with a word in $(ab)^+$. $L_{a,b}$ is generate by a *mgg* with the rules: $S \to \varepsilon(2{\cdot}S) \mid a(T)$, $S + S \to S$, $T \to \varepsilon(2{\cdot}T) \mid b(S) \mid b$, $T + T \to T$. □

4.2 Petri Net DAG Semantics

We regard unrestricted Petri nets. This means that we allow multiple arcs, unbounded places, auto-concurrency, etc. A *Petri net* $\mathcal{N}$ over Σ is a tuple $\mathcal{N} = (P, T, \mathcal{F}, \phi, s_0, F)$ of two finite sets P of *places* and T of *transitions*, $P \cap T = \emptyset$, a multi set $\mathcal{F}$ over $T \times P \cup P \times T$ of *arcs*, a *labelling mapping* $\phi : T \to \Sigma_\varepsilon$, an *initial* state $s_0 \in \mathbb{N}^P$, and a finite set $F \subseteq \mathbb{N}^P$ of *terminal* states. A transition labelled with $a \in \Sigma_\varepsilon$ is also called an a-transition. The multi sets ${}^t\mathcal{F}$ and $\mathcal{F}^t$ over P, defined by ${}^t\mathcal{F}(p) = \mathcal{F}(t, p)$, $\mathcal{F}^t(p) = \mathcal{F}(p, t)$, denote the sets of *output* and *input* places for a transition t. A *state* or *marking* s of $\mathcal{N}$ is a multi set over P. A multi set M over T is *fireable* or *enabled* in a state s, $s \Rightarrow^M$, iff $\sum_{t \in M} \mathcal{F}^t \leq s$. M *fires* from s to s', $s \Rightarrow^M s'$, iff $s \Rightarrow^M$ and $s' = s + \sum_{t \in M}({}^t\mathcal{F} - \mathcal{F}^t)$ hold. In case of $s \Rightarrow^M$ all transitions in M are in state s *concurrent*. If a transition t occurs in M with some multiplicity greater than 1 it is called *auto-concurrent*. We use the standard concepts of a firing sequence $x \in T^*$ from s to s', $s \Rightarrow^x s'$, and of a process, see, e.g., [11]. A process π (for $s \Rightarrow^x s'$) describes a possible concurrent execution of a firing $s \Rightarrow^x s'$. It is a DAG over $P \cup T$, with an in- and out-degree ≤ 1 for the P-nodes. The roots (leaves) of π are the multi sets s (s', respectively). A path from a root to a leave describes the life span of a single token in $s \Rightarrow^x s'$. Let $\Pi_{\mathcal{N}}(s, s')$ be the set of all processes of $\mathcal{N}$ from s to s'. We extend the labelling mapping ϕ of $\mathcal{N}$ to a fine morphism ϕ^+ from $T \cup P$ to Σ_ε by setting $\phi^+(x) := \phi(x)$ for $x \in T$ and $\phi^+(x) := \varepsilon$ for $x \in P$.

$\mathcal{D}_t(\mathcal{N}) := \bigcup_{s' \in F} \phi^+(\Pi_{\mathcal{N}}(s_0, s'))$ is the (terminal) DAG language accepted by $\mathcal{N}$. $\mathcal{P}_t(\mathcal{N}) := \{\alpha \in \Sigma^{\mathcal{G}_{u,u}} \mid \exists \beta \in \mathcal{D}_t(\mathcal{N})$ and α is the transitive closure of $\beta\}$ is the pomset language accepted by $\mathcal{N}$ and is the standard *pomset semantics* of $\mathcal{N}$. A set of DAGs (pomsets) is called *semi-rational* if it is accepted by some Petri net. $\mathcal{D}_t^{PN}$ ($\mathcal{P}_t^{PN}$) denotes the class of all semi-rational sets of DAGs (pomsets, respectively).

As ε-transitions are allowed in Petri nets it is possible to transform $\mathcal{N}$ into a 'normed' Petri net without changing the terminal DAG language L, see, e.g., [11]. In a normed Petri net there has to hold: $\mathcal{F} : P \times T \cup T \times P \to \{0, 1\}$, $s_0 = 1{\cdot}p_{start}$, for a certain *start* place, $\mathcal{F}^t \neq 0$ for all $t \in T$, and F consists only of 0, the empty

state, as terminal state. In contrast to the interleaving case in [12] there cannot exist a run-place as it would destroy concurrency. A Petri net $\mathcal{N}'$ with 0 as final state and a normed Petri net $\mathcal{N}''$, both equivalent to the Petri net $\mathcal{N}$ of Figure 4, are shown in Figure 5.

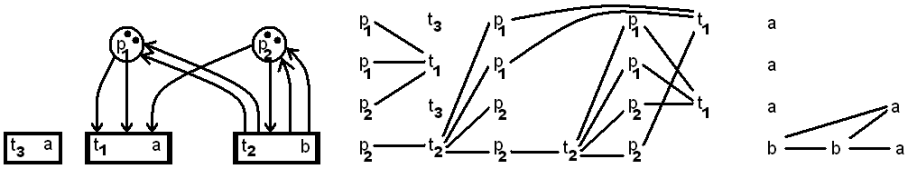


Fig. 4. A Petri net $\mathcal{N}$ with final state $1 \cdot p_2$, with a process π and the DAG $\alpha = \phi^+(\pi)$

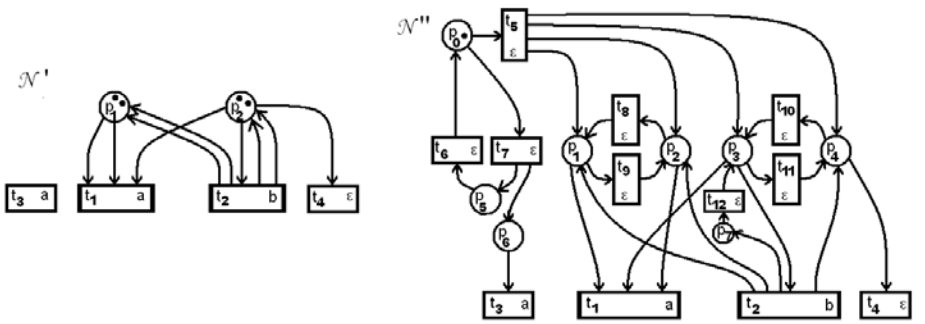


Fig. 5. Two Petri nets equivalent to $\mathcal{N}$

4.3 DAG Characterization Theorem

For a set $\mathcal{O}$ of operations on languages and a set $\mathcal{C}$ of languages $Cl^{\mathcal{O}}(\mathcal{C})$ denotes the smallest set of languages that contains all languages of $\mathcal{C}$ and is closed under all operations of $\mathcal{O}$. Let ε, (a) and $\binom{a}{b}$ denote the three discrete DAGs consisting of none node, one node labelled with a, and two concurrent nodes labelled with a and b. In the theory of concurrency a Dyck language $D^{\mathcal{D}} := \|^* \{a \to b\}$ of DAGs is known. The elements of $D^{\mathcal{D}}$ are thus forests consisting of several concurrent occurrences of the trivial chain $a \to b$. E.g., $\begin{smallmatrix} a \to b \\ a \to b \end{smallmatrix}$ is a DAG in $D^{\mathcal{D}}$. The set of all 'linearizations' of $D^{\mathcal{D}}$ is the Dyck language over words. The following is known

Theorem 2. *Algebraic Characterization Theorem*

$\mathcal{D}_t^{PN} = Cl^{Syn^{\mathcal{D}},\ fine\ hom,\ inverse\ very\ fine\ hom}\ (D^{\mathcal{D}}, \{(a)\}, \{\varepsilon, (a)\})$.

Sketch of proof: In [13] and [11], $\mathcal{P}_t^{PN} = Cl^{syn^{\mathcal{P}},\ fine\ hom,\ inverse\ very\ fine\ hom}$ $(D^{\mathcal{D}}, \{(a)\}, \{\varepsilon, \binom{a}{b}\})$ is shown. This proof follows an approach with a universal context for Petri nets, generalizing the universal context in [14]. $syn^{\mathcal{P}}$ can be expressed by $Syn^{\mathcal{P}}$. Schuth has shown in [15] that the basic set $\{\varepsilon, \binom{a}{b}\}$ may be

replaced by the much simpler set $\{\varepsilon, a\}$ and that this characterization also holds for DAG languages, where, of course, $Syn^{\mathcal{P}}$ must be replaces by $Syn^{\mathcal{D}}$. ■

Theorem 2 is closely related to the work of Hack [12] on words and of Grabowski [10] on pomsets. We now can state and prove the main result of this paper.

Theorem 3. *Graph Closure Characterization Theorem*

$Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{T}_{u,u},\varepsilon-free}) = Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{T}_{u,u}}) = Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{F}_{u,u}}) = \mathcal{D}^{merge} = \mathcal{D}_t^{PN}$.

Sketch of Proof: $Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{T}_{u,u},\varepsilon-free}) \subseteq Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{T}_{u,u}}) \subseteq Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{F}_{u,u}})$ is obvious. Also, $Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{F}_{u,u}}) \subseteq \mathcal{D}^{merge}$ is obvious as $\mathcal{D}^{merge}$ contains $Reg^{\mathcal{F}_{u,u}}$ and is closed under synchronization. $\mathcal{D}^{merge} \subseteq \mathcal{D}_t^{PN}$ is a trivial exercise: To get a Petri net from a *mgg* choose the variables as places. A merge rule $X_1 + X_2 \to X$ becomes an ε-transition with X_1 and X_2 as input places and X as an output place. A rule $X \to a(m)$ becomes an a-transition with X as the only input place and all variables occurring in m as output places.

It remains to transform a Petri net $\mathcal{N}$ into a regular graph grammar G such that the DAG semantics of $\mathcal{N}$ is the synchronization of the language of G. There are two different techniques for regular forest grammars with ε-rules and for regular tree grammars without. We show firstly $\mathcal{D}_t^{PN} \subseteq Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{F}_{u,u}})$:

We have to construct for any Petri net $\mathcal{N} = (P, T, \mathcal{F}, \Sigma, \phi, s_0, F)$ an *rgg* $G_{\mathcal{N}}$ s.t. $\mathcal{D}_t(\mathcal{N})$ is some synchronization of the forest language $L(G_{\mathcal{N}})$. As $\mathcal{N}$ may possess ε-transitions we assume that $F = \{0\}$ holds. Let $P = \{p_1, ..., p_n\}$. Define $G_{\mathcal{N}} := (V, \Delta, S, R)$ with $V := P \cup \{S, S'\}, \Delta := \{[t,i,j] \mid t \in T \wedge 0 \leq i \leq n \wedge 0 \leq j \leq \mathcal{F}(p_i,t)\}$, $R := \{S \to s_0 + S', S' \to \varepsilon(2 \cdot S'), S' \to \varepsilon\} \cup \{S' \to [t,0,0]({}^t\mathcal{F}) \mid t \in T\} \cup \{p_i \to [t,i,j] \mid t \in T \wedge 0 \leq i \leq n \wedge 0 \leq j \leq \mathcal{F}(p_i,t)\}$. The idea is to generate in $L(G_{\mathcal{N}})$ a skeleton of a valid process π of $\mathcal{N}$. The first rules generate a multi set of variables representing the initial state and an arbitrary number of variables S'. Each variable S' guesses one occurrence of a transition t in π and produces the simple tree consisting of the root $[t,0,0]$ and the multi set ${}^t\mathcal{F}$ as leaves. Each variable in ${}^t\mathcal{F}$ represents one token produced by a firing of t. The terminal $[t,0,0]$ stands for a virtual firing of t, where not all tokens on the input places of t might be available to enable a 'real' firing of t, or, in other words, where not all required variables of ${}^t\mathcal{F}$ may have been produced by $G_{\mathcal{N}}$. A variable p_i already generated represents a token on place p_i. This token on place p_i guesses to which transition t with $\mathcal{F}(p_i,t) > 0$ it will contribute by changing into $[t,i,j]$. To be more exact, $[t,i,j]$ tells that the j-th token on place p_i with $1 \leq j \leq \mathcal{F}(p_i,t)$ required to fire t has been produced. A forest derivable in $G_{\mathcal{N}}$ consists thus of trivial trees of depth 1 of the form $[t,0,0]({}^t\mathcal{F})$ or of depth 0 of the form p_i or $[t,i,j]$. To get a valid process from such a skeleton we apply for each $t \in T$ a generalized synchronization $Syn^{\mathcal{D}}[E(t),t]$ with $E(t) := \{[t,i,j] \mid 0 \leq i \leq n \wedge 0 \leq j \leq \mathcal{F}(p_i,t)\}$. As such a synchronization must not produce cycles it is ensured that no occurrences of variables $[t,i,j]$ and $[t,i',j']$ are merged where the occurrence of $[t,i,j]$ represents a token that is

produced by a side effect of a firing of this occurrence of $[t, i', j']$. Thus, $\mathcal{D}_t(\mathcal{N}) = Syn^{\mathcal{D}}[E(t_0), \phi(t_0)] \circ ... \circ Syn^{\mathcal{D}}[E(t_m), \phi(t_m)](L(G_{\mathcal{N}})$, for $T = \{t_0, ..., t_m\}$.

Example 3.

We present $G_{\mathcal{N}'}$ for $\mathcal{N}'$ in Figure 5. The rules are: $S \to \varepsilon(2 \cdot p_1 + 2 \cdot p_2 + S')$, $S' \to \varepsilon(2 \cdot S') \mid \varepsilon \mid [t_1, 0, 0] \mid [t_3, 0, 0] \mid [t_4, 0, 0] \mid [t_2, 0, 0](2 \cdot p_1 + 2 \cdot p_2)$, $p_1 \to [t_1, 1, 1] \mid [t_1, 1, 2]$, and $p_2 \to [t_1, 2, 1] \mid [t_2, 2, 1] \mid [t_4, 2, 1]$. A derivation in $G_{\mathcal{N}'}$ is given in Figure 6 that ends with a forest corresponding to the process in Figure 4. Applying now all $Syn^{\mathcal{D}}[E(t), \phi(t)]$ synchronization results in some DAGs including that of Figure 4. □

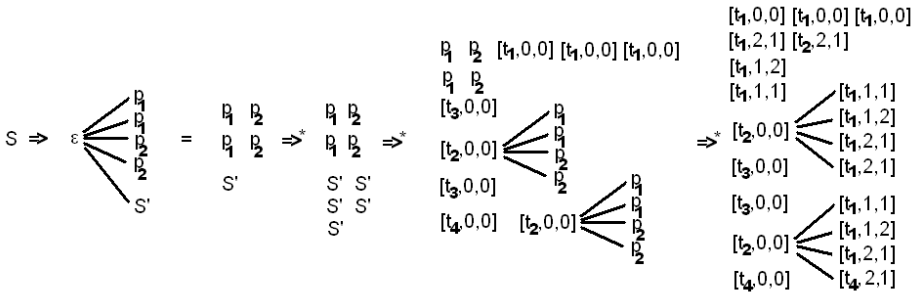


Fig. 6. A forest of 12 trees in $L(G_{\mathcal{N}'})$

$\mathcal{D}_t^{PN} \subseteq Cl^{Syn^{\mathcal{D}}}(Reg^{\mathcal{T}_{u,u},\varepsilon-free})$: We cannot use the above trick to generate sufficiently many concurrent variables S', each of which generates a virtual firing $[t, 0, 0]({}^t\mathcal{F})$, as we have no ε-rules in the grammar. The idea is to generate the virtual firings on the run, whenever we have generated a certain variable that belongs to $\mathcal{F}^t$. To distinguish uniquely one variable for this purpose we operate with ordered places in $\mathcal{N}$ and without multiple arcs between a place and a transition. Thus, without restriction we assume that $\mathcal{N} = (P, T, \mathcal{F}, \Sigma, \phi, 1p_0, \{0\})$ with $P = \{p_0, p_1, ..., p_n\}$ is a normed Petri net. Define $G'_{\mathcal{N}} := (V, \Delta, S, R)$ with $V := P, S := p_0, \Delta := \{[t, i] \mid t \in T \wedge 1 \leq i \leq n\}$ and R is constructed as follows: For $t \in T$ define p_t to be the place p_i with $\mathcal{F}(p_i, t) = 1$ and $\mathcal{F}(p_j, t) = 0$ for all $j < i$ (note, $\mathcal{F}^t \neq 0$ holds in a normed Petri net). For all $p_i, 1 \leq i \leq n$, and for all $t \in T$ we add to R the rule $p_i \to [t, i]({}^t\mathcal{F})$ if $p_t = p_i$ and the rule $p_i \to [t, i]$ if $p_t \neq p_i \ \wedge \ \mathcal{F}(p_i, t) = 1$. As before, $L(G'_{\mathcal{N}})$ consists of all skeletons of possible processes of $\mathcal{N}$. But now the virtual firings $[t, p_t]({}^t\mathcal{F})$ are not generated all at once from a variable S', but on the run whenever a distinguished variable p_t is available. $L(G'_{\mathcal{N}})$ consists thus of trees of arbitrary depth where all inner nodes are $[t, i]$-terminals. We conclude as above that $\mathcal{D}_t(\mathcal{N}) = Syn^{\mathcal{D}}[E(t_0), \phi(t_0)] \circ ... \circ Syn^{\mathcal{D}}[E(t_m), \phi(t_m)](L(G'_{\mathcal{N}})$ for $T = \{t_0, ..., t_m\}$ and $E(t) = \{[t, i] \mid 1 \leq i \leq \mathcal{F}(p_i, t)\}$. ■

Example 4.

We present $G'_{\mathcal{N}''}$ for the normed Petri net $\mathcal{N}''$ of Figure 5. The rules are $p_0 \to [t_7,0](p_5+p_6) \mid [t_5,0](p_1+p_2+p_3+p_4)$, $p_1 \to [t_1,1] \mid [t_9,1](p_2)$, $p_2 \to [t_1,2] \mid [t_8,2](p_1)$, $p_3 \to [t_1,3] \mid [t_2,3](p_1+p_2+p_4+p_7) \mid [t_{11},3](p_4)$, $p_4 \to [t_4,4] \mid [t_{10},4](p_5)$, $p_5 \to [t_6,5](p_0)$, and $p_7 \to [t_{12},7](p_3)$.

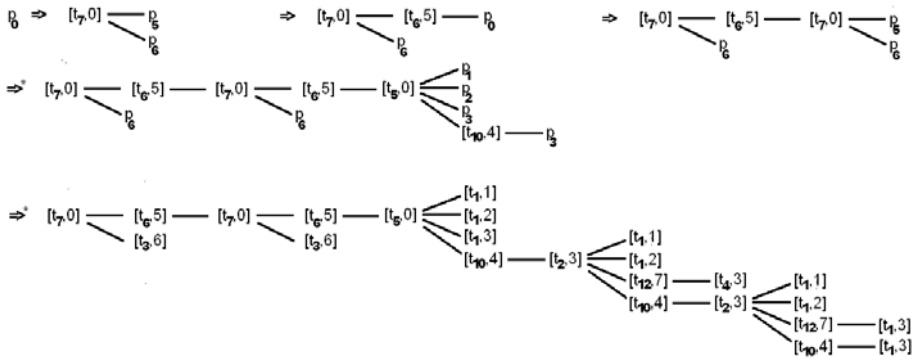


Fig. 7. A derivation in $G'(\mathcal{N}'')$

Figure 7 presents an example of a derivation in $G'_{\mathcal{N}''}$. Applying $Syn^{\mathcal{D}}[E(t_i),\varepsilon]$ for $4 \le i \le 12$ corresponds to the process of Figure 4, and a further application of $Syn^{\mathcal{D}}[\{[t_1,1],[t_1,2],[t_1,3]\},a]$, $Syn^{\mathcal{D}}[\{[t_2,3]\},b]$ and $Syn^{\mathcal{D}}[\{[t_3,6]\},a]$ results in DAGs including that one from Figure 4. □

Open Questions: Is it possible to extent the concept of $T_{u,u}$-magmas to 'DAG-magmas' in such a way that recognizable sets of DAGs coincide with semi-rational ones? Probably such 'recognizing DAG-magmas' will not be finite in all aspects, similar to Courcelles magmas for finite graphs with infinitely many sorts. Therefore a term 'semi-recognizability' might be better. A positive answer would thus result in an equivalence of semi-recognizability, semi-rationality, and semi-regularity for DAGs. Is there a similar Graph Closure Characterization Theorem for Petri nets without ε-transitions?

References

1. W. Thomas. Finite-state recognizability of graph properties. In D. Krob, editor, *Theorie des Automates et Applicationbs*, volume 172, pages 147–159. l'Universite de Rouen, France, 1992.
2. B. Courcelle. A representation of graphs by algebraic expressions and its use for graph rewriting systems. In *Proc. 3rd Internat. Workshop on Graph-Grammars, LNCS*, pages 112–132. Springer Verlag, 1988.

3. B. Courcelle. On context-free sets of graphs and their monadic second-order theory. In *Proc. 3rd Internat. Workshop on Graph-Grammars, LNCS*, pages 133–146. Springer Verlag, 1988.
4. T. Kamimura, G. Slutzki. Parallel and two-way automata on directed ordered acyclic graphs. *Inf. Control*, 49:10–51, 1981.
5. H. Comon, M. Daucher, R. Gilleron, S. Tison, M. Tommasi. Tree automata techniques and application. Available on the Web from 13ux02.univ-lille.fr in directoty tata, 1998.
6. A. Podelski. Model checking as constraint solving. In *Proc. of SAS'2000:Statistic Analysis Symposium, Santas Barbara*, 2000.
7. A. Brüggemann-Klein, M. Murata, D. Wood. Regular tree and hedge languages of unranked alphabets. Theor. Comp. Science Center Report HKUST-TCSC 2001-5,pp29, 2001.
8. B. Courcelle. On recognizable sets and tree automata. In H. Aït-Kaci, M. Nivat, editor, *Resolution of Equations in Algebraic Structures*, volume 1, pages 93–126. Academic Press, 1989.
9. P. Starke. Graph grammars for Petri net processes. *EIK*, 19:199–233, 1983.
10. J. Grabowski. On partial languages. *Annales Societatis Mathematicas Polonae, Fundamenta Informaticae IV.2*, pages 428–498, 1981.
11. L. Priese, H. Wimmel. *Theoretische Informatik: Petri-Netze*. Springer Verlag, 2002.
12. M. Hack. Petri net languages. Technical report, Computation Structure Group Memo 124, Project MAC, MIT, 1975.
13. H. Wimmel. *Algebraische Semantiken für Petri-Netze (Ph.D. Thesis)*. Verlag Fölbach, Koblenz, 2000.
14. M. Nielsen, L. Priese, V. Sassone. Characterizing behavioural congruences for Petri nets. In *Proc. CONCUR'95, LNCS 609*, volume B, pages 175–189, 1995.
15. M. Schuth. Petri-Netz Semantik mittels dags. Technical report, Master Thesis, Universität Koblenz-Landau, 2004.

On the Frequency of Letters in Pure Binary Morphic Sequences

Kalle Saari

Turku Centre for Computer Science
20520 Turku, Finland
kasaar@utu.fi

Abstract. It is well-known that the frequency of letters in primitive morphic sequences exists. We show that the frequency of letters exists in pure binary morphic sequences generated by non-primitive morphisms. Therefore, the letter frequency exists in every pure binary morphic sequence. We also show that this is somewhat optimal, in the sense that the result does not hold in the class of general binary morphic sequences. Finally, we give an explicit formula for the frequency of letters.

1 Introduction

Several important results on the frequency of letters in morphic sequences were established in the early 1970's. In his influential paper from 1972, Cobham [3] proved that the frequency of a letter in an automatic sequence, if it exists, is rational. Michel [4, 5] showed in 1975 that if a morphic sequence is primitive, then the frequency of all letters exists. These results can be used, for example, to show that certain sequences are not k-automatic for any integer $k \geq 1$ (see [2]). For an interesting application of the frequency of letters to transcendental number theory, see [1].

In this paper we are interested in the existence of the letter frequency in binary morphic sequences. As we see in Example 1, the frequency of letters does not have to exist in binary morphic sequences. We show, however, that in any *pure* binary morphic sequence the letter frequency does exist. To be precise, we prove that if a pure binary morphic sequence is generated by a non-primitive morphism, then the frequency of letters exists. Using Michel's result, the general statement then follows. Finally, we give an explicit formula for the frequency of letters.

2 Preliminaries

A morphism is a map $\varphi : \Sigma^* \to \Sigma^*$ that satisfies the identity $\varphi(xy) = \varphi(x)\varphi(y)$ for all words $x, y \in \Sigma^*$. Here Σ^* is the free monoid over the finite alphabet Σ. We say that φ is *primitive* if there exists an integer $k \geq 1$ such that, for every $a, b \in \Sigma$, the letter a occurs in $\varphi^k(b)$.

C. De Felice and A. Restivo (Eds.): DLT 2005, LNCS 3572, pp. 397–408, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The letter $a \in \Sigma$ is *mortal* if $\varphi^k(a) = \epsilon$, where ϵ is the empty word, for some integer $k \geq 1$. The morphism φ is *prolongable* on the letter $a \in \Sigma$ if $\varphi(a) = ax$ and $x \in \Sigma^*$ is a nonempty word that contains a non-mortal letter. In this case, φ generates the infinite sequence

$$\varphi^\omega(a) := \lim_{n\to\infty} \varphi^n(a) = a\, x\, \varphi(x)\, \varphi^2(x) \cdots ,$$

which is said to be a *fixed point* of φ. An infinite sequence generated in this fashion is called a *pure morphic sequence*. A sequence is *morphic* if it is the image of a pure morphic sequence under a letter-to-letter morphism and it is *binary* if it is defined over a binary alphabet.

Let $x \in \Sigma^*$ be a finite word, and let $y \in \Sigma^\infty$ be a finite or an infinite word. We call x a *prefix* of y if there exists a word $y' \in \Sigma^\infty$ such that $y = xy'$. This is denoted by $x \trianglelefteq y$. If in addition $x \neq y$, then the notation $x \triangleleft y$ is used.

Let $w = w_1 w_2 \cdots w_n$ be a finite word, where $w_i \in \Sigma$ are letters. Note that, in this paper, we index letters in a word starting from 1, not from 0. The length n of w is denoted by $|w|$ (we use the same notation for the absolute value of a real number, but this should cause no ambiguity). The number of occurrences of $a \in \Sigma$ in w is denoted by $|w|_a$.

Suppose that $\Sigma = \{a_1, \ldots, a_d\}$. Then the *incidence matrix* $M(\varphi)$ associated with φ is $M(\varphi) = (m_{i,j})_{1\leq i,j\leq d}$, where $m_{i,j} = |\varphi(a_j)|_{a_i}$.

Let $w = w_1 w_2 \cdots$ be an infinite sequence. We define $f_{w,a} \colon \mathbf{Z}^+ \to \mathbf{Z}^+ \cup \{0\}$ to be the counting function for the occurrences of the letter a in w. More precisely, if w_k is the prefix of w of length k, then $f_{w,a}(k) = |w_k|_a$. If the limit

$$\lim_{k\to\infty} \frac{f_{w,a}(k)}{k}$$

exists, it is said to be the *frequency* of the letter a in w.

3 Preparatory Results

In this section we present a couple of further definitions and establish some lemmata needed in the next section. We begin, however, by showing that the frequency of letters does not exist in all binary morphic sequences:

Example 1. Define the infinite sequence

$$w = 0100110000111100000 \cdots = 0^{2^0}\, 1^{2^0}\, 0^{2^1}\, 1^{2^1}\, 0^{2^2}\, 1^{2^2}\, 0^{2^3}\, 1^{2^3} \cdots$$

from the obvious rule. Let $A(n) = 2^n + 2^{n-1} - 2$ be the position of the last 0 in the nth block of 0's. Similarly, let $B(n) = 2^{n+1} - 2$ be the position of the last 1 in the nth block of 1's. Then $f_{w,0}\big(B(n)\big) = f_{w,0}\big(A(n)\big) = 2^n - 1$, and so

$$\frac{f_{w,0}\big(A(n)\big)}{A(n)} = \frac{2^n - 1}{2^n + 2^{n-1} - 2} \longrightarrow \frac{2}{3} \qquad \text{and} \qquad \frac{f_{w,0}\big(B(n)\big)}{B(n)} = \frac{2^n - 1}{2^{n+1} - 2} \longrightarrow \frac{1}{2},$$

as $n \longrightarrow \infty$. Consequently, the frequency of 0 in w does not exist.

Observe that w is a binary morphic sequence. Namely, $w = \psi(\phi^\omega(a))$, where ϕ is the generating morphism $\phi : a \mapsto ab, b \mapsto cc, c \mapsto b$ and ψ is a coding $\psi : a \mapsto 0, b \mapsto 1, c \mapsto 0$. Therefore there exist binary morphic sequences with no frequency of letters. Of course, this same example shows that there exists ternary pure morphic sequences for which the letter frequency does not exist.

Henceforth, we only consider pure binary morphic sequences generated by a non-primitive morphism. Without loss of generality, we assume that $\varphi : \{0,1\} \to \{0,1\}^*$ is a non-primitive, binary morphism prolongable on 0. By the definition of a non-primitive morphism, it follows that either $\varphi(0) \in 0^*$ or $\varphi(1) \in 1^*$. If $\varphi(0) \in 0^*$, then the frequency of letters 0 and 1 trivially exists, for then $\varphi^\omega(0) = 0^\omega$. This is why we now exclude this possibility from our discussion and assume that 1 occurs in $\varphi(0)$ and $\varphi(1) \in 1^*$. Thus the associated incidence matrix $M(\varphi)$ is of the form

$$\begin{pmatrix} a & 0 \\ c & d \end{pmatrix} := \begin{pmatrix} |\varphi(0)|_0 & |\varphi(1)|_0 \\ |\varphi(0)|_1 & |\varphi(1)|_1 \end{pmatrix},$$

where now $a, c, d > 0$.

Since the sequences we are now considering are exclusively of the form $\varphi^\omega(0)$, we will use the shorthand $f_\varphi = f_{\varphi^\omega(0),0}$ for the counting function of 0 in $\varphi^\omega(0)$.

In order to show that the frequency of letters in $\varphi^\omega(0)$ exists, we define two kinds of extremal morphisms, η and μ, which have the same associated matrix as φ. We shall show that the counting functions f_η, f_μ trap the behavior of f_φ in the sense that it will be enough to establish the existence of the letter frequency in $\eta^\omega(0)$ and $\mu^\omega(0)$.

Definition 1. *We define the morphisms* $\mu, \eta\colon \{0,1\}^* \to \{0,1\}^*$ *as follows:*

$$\begin{cases} \eta(0) = 0\,1^c\,0^{a-1}, \\ \eta(1) = 1^d, \end{cases} \qquad \begin{cases} \mu(0) = 0^a\,1^c, \\ \mu(1) = 1^d. \end{cases}$$

Note that $M(\eta) = M(\mu) = M(\varphi)$.

Remark 1. Suppose the morphism $\psi\colon \{0,1\}^* \to \{0,1\}^*$ has the same incidence matrix as φ, that is, $M(\psi) = M(\varphi)$. Then an easy proof by induction shows that $M(\psi^n) = M(\varphi^n)$. Consequently, for any $u, v \in \{0,1\}^*$, we have

$$|u|_0 \geq |v|_0 \qquad \text{implies} \qquad |\psi^n(u)|_0 \geq |\varphi^n(v)|_0.$$

Lemma 1. *For all* $k \geq 1$,

$$f_\mu(k) \geq f_\varphi(k).$$

Proof. We argue by contradiction; suppose the statement is false. Let k be the smallest positive integer such that $f_\mu(k) < f_\varphi(k)$, and let n be the smallest integer for which

$$|\mu^{n+1}(0)| = |\varphi^{n+1}(0)| \geq k. \tag{1}$$

Write $\mu(0) = 0^a\,1^c = x_1 \cdots x_{a+c}$ and $\varphi(0) = y_1 \cdots y_{a+c}$, where $x_i, y_j \in \{0,1\}$. Note that $\mu^{n+1}(0) = \mu^n(x_1 \cdots x_{a+c})$ and $\varphi^{n+1}(0) = \varphi^n(y_1 \cdots y_{a+c})$. Let p, q be the largest integers for which

$$|\mu^n(x_1 \cdots x_p)| < k \qquad \text{and} \qquad |\varphi^n(y_1 \cdots y_q)| < k.$$

Condition (1) implies that $p, q < a + c$. On the other hand, since n is minimal and $x_1 = y_1 = 0$, it follows that $p, q > 0$.

Due to the minimality of k, the letter in $\varphi^\omega(0)$ indicated by k, which occurs in the block $\varphi^n(y_{q+1})$, must be 0. Hence also $y_{q+1} = 0$. Moreover, we claim that $x_{p+1} = 0$. For if $x_{p+1} = 1$, then 0^a is a prefix of $x_1 \cdots x_p$, and thus

$$f_\varphi(k) > f_\mu(k) \geq |\mu^n(x_1 \cdots x_p)|_0 = |\mu^n(x_1, \ldots, x_{a+c})|_0 = |\varphi^{n+1}(0)|_0 \geq f_\varphi(k),$$

a contradiction. Thus $x_{p+1} = 0$, and hence $x_1 \cdots x_{p+1} = 0^{p+1}$.

We have the following three cases to consider:

Case(i). Suppose $|y_1 \cdots y_q|_0 > |x_1 \cdots x_p|_0$. Then $|y_1 \cdots y_q|_0 \geq |x_1 \cdots x_{p+1}|_0$, and since $x_1 \cdots x_{p+1} = 0^{p+1}$, we trivially have $|y_1 \cdots y_q|_1 \geq |x_1 \cdots x_{p+1}|_1$. Therefore,

$$k > |\varphi^n(y_1 \cdots y_q)| \geq |\mu^n(x_1 \cdots x_{p+1})| \geq k,$$

a contradiction.

Case (ii). Suppose $|y_1 \cdots y_q|_0 < |x_1 \cdots x_p|_0$. Then $|y_1 \cdots y_{q+1}|_0 \leq |x_1 \cdots x_p|_0$, and so $|\varphi^n(y_1 \cdots y_{q+1})|_0 \leq |\mu^n(x_1 \cdots x_p)|_0$. Consequently,

$$f_\mu(k) < f_\varphi(k) \leq |\varphi^n(y_1 \cdots y_{q+1})|_0 \leq |\mu^n(x_1 \cdots x_p)|_0 \leq f_\mu(k),$$

a contradiction.

Case (iii). Suppose $|y_1 \cdots y_q|_0 = |x_1 \cdots x_p|_0$. Denote $h = |\mu^n(x_1 \cdots x_p)|$ and $l = |\varphi^n(y_1 \cdots y_q)|$. As $|y_1 \cdots y_q|_0 = |x_1 \cdots x_p|_0$ and $x_1 \cdots x_p = 0^p$, it follows that $h \leq l$ and $f_\mu(h) = f_\varphi(l)$. Now the assumption $f_\mu(k) < f_\varphi(k)$ implies that the prefix of $\mu^n(x_{p+1})$ of length $k - h$ has less 0's than the prefix of $\varphi^n(y_{q+1})$ of length $k - l$. But as $x_{p+1} = 0$ and $y_{q+1} = 0$, this is the same as saying $f_\mu(k - h) < f_\varphi(k - l)$. Since $h \leq l < k$, it is plain that $f_\varphi(k - l) \leq f_\varphi(k - h)$. Hence $f_\mu(k - h) < f_\varphi(k - h)$, contradicting the minimality of k. This concludes the proof.

Lemma 2. *For all $k \geq 1$,*

$$f_\eta(k) \leq f_\varphi(k)\,.$$

Proof. Assume the contrary. Suppose k is the smallest integer such that the inequality $f_\eta(k) > f_\varphi(k)$ holds, and let n be the smallest integer for which

$$|\eta^{n+1}(0)| = |\varphi^{n+1}(0)| \geq k. \tag{2}$$

Write $\eta(0) = 0\,1^c 0^{a-1} = x_1 \cdots x_{a+c}$ and $\varphi(0) = y_1 \cdots y_{a+c}$ with $x_i, y_j \in \{0,1\}$. Note that $\eta^{n+1}(0) = \eta^n(x_1 \cdots x_{a+c})$ and $\varphi^{n+1}(0) = \varphi^n(y_1 \cdots y_{a+c})$. Let p, q be the largest integers for which

$$|\eta^n(x_1 \cdots x_p)| < k \qquad \text{and} \qquad |\varphi^n(y_1 \cdots y_q)| < k.$$

Condition (2) implies that $p, q < a + c$. Since n is minimal and $x_1 = y_1 = 0$, we also have $p, q > 0$.

Since k is assumed to be minimal, the letter of $\eta^\omega(0)$ occurring at position k, which occurs in the block $\eta^n(x_{p+1})$, must be 0, and so also $x_{p+1} = 0$. Thus, since $p > 0$, we see that 01^c must be a prefix of $x_1 \cdots x_p$. We have the following three cases to consider:

Case (i). Suppose $|x_1 \cdots x_p|_0 > |y_1 \cdots y_q|_0$. This assumption and the fact that $|x_1 \cdots x_p|_1 = c$, respectively, imply

$$|x_1 \cdots x_p|_0 \geq |y_1 \cdots y_{q+1}|_0 \qquad \text{and} \qquad |x_1 \cdots x_p|_1 \geq |y_1 \cdots y_{q+1}|_1. \tag{3}$$

Thus

$$k > |\eta^n(x_1 \cdots x_p)| \geq |\varphi^n(y_1 \cdots y_{q+1})| \geq k, \tag{4}$$

a contradiction.

Case(ii). Suppose $|x_1 \cdots x_p|_0 < |y_1 \cdots y_q|_0$. This implies

$$f_\varphi(k) < f_\eta(k) \leq |\eta^n(x_1 \cdots x_{p+1})|_0 \leq |\varphi^n(y_1 \cdots y_q)|_0 \leq f_\varphi(k),$$

a contradiction.

Case(iii). Suppose $|x_1 \cdots x_p|_0 = |y_1 \cdots y_q|_0$. If $y_{q+1} = 1$, then the inequalities in (3) hold, and so does (4), a contradiction. Thus $y_{q+1} = 0$. Denote $h := |\eta^n(x_1 \cdots x_p)|$ and $l := |\varphi^n(y_1 \cdots y_q)|$. As $|x_1 \cdots x_p|_0 = |y_1 \cdots y_q|_0$ and $|x_1 \cdots x_p|_1 = c \geq |y_1 \cdots y_q|_1$, it follows that $h \geq l$ and $f_\eta(h) = f_\varphi(l)$. Now the assumption $f_\eta(k) > f_\varphi(k)$ implies that the prefix of $\eta^n(x_{p+1})$ of length $k - h$ must contain more 0's that the prefix of $\varphi^n(y_{q+1})$ of length $k - l$. But as $x_{p+1} = y_{q+1} = 0$, this means that $f_\eta(k - h) > f_\varphi(k - l)$. Since $h \geq l$, it is certainly true that $f_\eta(k - l) \geq f_\eta(k - h)$. Consequently, $f_\eta(k - l) > f_\varphi(k - l)$, contradicting the minimality of k. This completes the proof.

Lemma 3. *For all $n \geq 1$,*

$$|\varphi^n(0)|_0 = a^n \qquad \textit{and} \qquad |\varphi^n(0)|_1 = \begin{cases} ca^{n-1}n & \textit{if } a = d, \\ c\dfrac{d^n - a^n}{d - a} & \textit{if } a \neq d. \end{cases} \tag{5}$$

Proof. An easy proof by induction shows that

$$\begin{pmatrix} |\varphi^n(0)|_0 \\ |\varphi^n(0)|_1 \end{pmatrix} = \begin{pmatrix} a & 0 \\ c & d \end{pmatrix}^n \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} a & 0 \\ c & d \end{pmatrix}^n = \begin{pmatrix} a^n & 0 \\ c\sum_{i=0}^{n-1} a^{n-1-i}d^i & d^n \end{pmatrix}$$

for all $n \geq 0$. Therefore, it is enough to observe that

$$c\sum_{i=0}^{n-1} a^{n-1-i}d^i = ca^{n-1}\sum_{i=0}^{n-1}\left(\frac{d}{a}\right)^i = \begin{cases} ca^{n-1}n & \text{if } a = d, \\ c\dfrac{d^n - a^n}{d - a} & \text{if } a \neq d. \end{cases}$$

Lemma 4. *The limit*

$$\alpha_0 := \lim_{n\to\infty} \frac{|\varphi^n(0)|_0}{|\varphi^n(0)|}$$

exists, and moreover

$$\alpha_0 = \begin{cases} 0 & \text{if } a \le d, \\ \dfrac{a-d}{a-d+c} & \text{if } a > d. \end{cases} \tag{6}$$

Proof. First, assume that $a = d$. Using the equations in Lemma 3, we see that

$$\frac{|\varphi^n(0)|_0}{|\varphi^n(0)|} = \frac{|\varphi^n(0)|_0}{|\varphi^n(0)|_0 + |\varphi^n(0)|_1} = \frac{a^n}{a^n + ca^{n-1}n} = \frac{1}{1 + \dfrac{cn}{a}},$$

so that the fraction converges to 0 as $n \longrightarrow \infty$.

Now suppose that $a \neq d$. As above, we get

$$\frac{|\varphi^n(0)|_0}{|\varphi^n(0)|} = \frac{|\varphi^n(0)|_0}{|\varphi^n(0)|_0 + |\varphi^n(0)|_1} = \frac{a^n}{a^n + c\dfrac{d^n - a^n}{d-a}} = \frac{1}{1 + c\dfrac{\left(\frac{d}{a}\right)^n - 1}{d-a}},$$

and so we see that, as $n \longrightarrow \infty$, the fraction converges to $(a-d)/(a-d+c)$ if $d < a$, and to 0 if $d > a$. Accordingly, the limit exists in each case and equals the asserted value.

Lemma 5. *Let $G \ge 1$ be an integer. Then every finite prefix w of $\mu^\omega(0)$ has a factorization of the form*

$$w = \mu^{n_1}(0)\mu^{n_2}(0)\cdots\mu^{n_r}(0)x, \tag{7}$$

where $r \ge 0$, $n_1 \ge n_2 \ge \cdots \ge n_r \ge G$, and either

$$|x| \le |\mu^G(0)| \qquad \text{or} \qquad x \trianglelefteq \mu^{n_r}(1^c).$$

Proof. We verify the claim by induction on $|w|$. If $|w| \le |\mu^G(0)|$, we may take $r = 0$ and $x = w$. Now assume that $|w| > |\mu^G(0)|$. Let n be the integer for which

$$|\mu^n(0)| < |w| \le |\mu^{n+1}(0)| = |\mu^n(0^a\,1^c)|.$$

Since $|w| > |\mu^G(0)|$, we have $n \ge G$. Now there are two cases to consider:

Case (i). Suppose $|\mu^n(0^a)| < |w| \le |\mu^n(0^a\,1^c)|$. Then we can write $w = \mu^n(0^a)x = \left(\mu^n(0)\right)^a x$, where $x \trianglelefteq \left(\mu^n(1)\right)^c$. Since $n \ge G$, we have a factorization in the correct form.

Case (ii). Suppose $|\mu^n(0)| < |w| \le |\mu^n(0^a)|$. Then $w = \left(\mu^n(0)\right)^i w'$ for some $1 \le i \le a$ and $w' \triangleleft \mu^n(0)$. Because w' is a prefix of $\mu^\omega(0)$, we can apply the induction assumption to find a factorization

$$w' = \mu^{n_1}(0)\mu^{n_2}(0)\cdots\mu^{n_r}(0)x,$$

where $r \ge 0$, $n_1 \ge n_2 \ge \cdots \ge n_r \ge G$, and either $|x| \le |\mu^G(0)|$ or $x \trianglelefteq \left(\mu^{n_r}(1)\right)^c$. Since $n > n_1$, the factorization

$$w = \left(\mu^n(0)\right)^i \mu^{n_1}(0)\mu^{n_2}(0)\cdots\mu^{n_r}(0)x$$

is of the required form. This concludes our proof.

Lemma 6. *Let $G \geq 1$ be an integer. Then every finite prefix w of $\eta^\omega(0)$ has a factorization of the form*

$$w = \eta^{n_1}\left(01^c0^{m_1}\right)\eta^{n_2}\left(01^c0^{m_2}\right)\cdots\eta^{n_r}\left(01^c0^{m_r}\right)x, \tag{8}$$

where $r \geq 0$, $n_1 > n_2 > \cdots > n_r \geq G$, $0 \leq m_1, m_2, \ldots, m_r \leq a-1$, and either

$$|x| \leq |\eta^G(0)| \qquad \text{or} \qquad \eta^{n_{r+1}}(0) \lhd x \trianglelefteq \eta^{n_{r+1}}(01^c)$$

with $n_r > n_{r+1} \geq G$.

Proof. The proof is similar to the proof of Lemma 5, and we omit the details.

4 The Existence of the Frequency of Letters

We are ready to employ the results from the previous section to establish some existence results about the frequency of letters.

Proposition 1. *Assume $a \leq d$. Then the frequency of letters in $\varphi^\omega(0)$ exists.*

Proof. Denote $l_n = |\varphi^n(0)|$. We claim that the frequency of the letter 0 is 0, and thus 1 occurs with frequency 1. It is enough to show that

$$H_n = \max\left\{\frac{f_\varphi(k)}{k} \,:\, l_n \leq k \leq l_{n+1}\right\} \longrightarrow 0$$

as $n \to \infty$.

To do that, we write $\varphi(0) = 0x$, where x is a nonempty word in $\{0,1\}^*$, so that $\varphi^{n+1}(0) = \varphi^n(0)\varphi^n(x)$. We denote $\gamma(n) = |\varphi^n(x)|_0$. Since

$$\frac{i}{j} \leq \frac{i+1}{j+1}$$

whenever $0 < i \leq j$, it follows that

$$\frac{f_\varphi(k)}{k} \leq \frac{|\varphi^n(0)\,0^{\gamma(n)}|_0}{|\varphi^n(0)\,0^{\gamma(n)}|} \tag{9}$$

for all $l_n \leq k \leq l_{n+1}$. Using the formulas in (5), we get

$$|\varphi^n(0)\,0^{\gamma(n)}|_0 = |\varphi^{n+1}(0)|_0 = a^{n+1} \tag{10}$$

and

$$\begin{aligned} |\varphi^n(0)\,0^{\gamma(n)}| &= |\varphi^n(0)0^{\gamma(n)}|_0 + |\varphi^n(0)0^{\gamma(n)}|_1 = a^{n+1} + |\varphi^n(0)|_1 \\ &= \begin{cases} a^{n+1} + ca^{n-1}n & \text{if } a = d, \\ a^{n+1} + c\dfrac{d^n - a^n}{d-a} & \text{if } a < d. \end{cases} \end{aligned} \tag{11}$$

Now Inequality (9) implies that

$$H_n \leq \frac{|\varphi^n(0)\, 0^{\gamma(n)}|_0}{|\varphi^n(0)\, 0^{\gamma(n)}|}.$$

But here, by (10) and (11),

$$H_n \leq \frac{|\varphi^n(0)\, 0^{\gamma(n)}|_0}{|\varphi^n(0)\, 0^{\gamma(n)}|} \longrightarrow 0$$

as $n \longrightarrow \infty$, which can be verified by a similar computation as in Lemma 4. This concludes the proof.

Proposition 2. *Assume $a > d$. Then the frequency of letters exists in $\mu^\omega(0)$.*

Proof. By Lemma 4, we know that the limit $\lim_{n\to\infty} |\mu^n(0)|_0/|\mu^n(0)|$ exists, and we denote it by α_0. We shall prove that $f_\mu(k)/k$ converges to α_0, so that α_0 is the frequency of 0 and $1-\alpha_0$ the frequency of 1.

Let $\epsilon > 0$. We will show that, for every sufficiently long prefix w of $\mu^\omega(0)$,

$$\left| \frac{|w|_0}{|w|} - \alpha_0 \right| < \epsilon,$$

which will conclude the proof.

Let m_1 be an integer such that

$$\left| |\mu^n(0)|_0 - \alpha_0 |\mu^n(0)| \right| < \frac{\epsilon}{2} |\mu^n(0)|$$

for all $n \geq m_1$.

Next, since $a > d$, we see that

$$\frac{|\mu^n(1)|}{|\mu^n(0)|} \leq \frac{d^n}{a^n} \longrightarrow 0$$

as $n \to \infty$. Thus there exists an integer m_2 such that

$$\frac{|\mu^n(1)|}{|\mu^n(0)|} < \frac{\epsilon}{2c}$$

for all $n \geq m_2$.

Now denote $G = \max\{m_1, m_2, 1\}$. Let w be a prefix of $\mu^\omega(0)$ of length

$$|w| > \frac{2|\mu^G(0)|}{\epsilon}.$$

As in Lemma 5, we write

$$w = \mu^{n_1}(0)\mu^{n_2}(0)\cdots\mu^{n_r}(0)x,$$

where $r \geq 0$, $n_1 \geq n_2 \geq \cdots \geq n_r \geq G$, and either $|x| \leq |\mu^G(0)|$ or $x \trianglelefteq \mu^{n_r}(1^c)$. Then

$$\Big||w|_0 - \alpha_0|w|\Big| = \Big|\sum_{i=1}^{r} \big(|\mu^{n_i}(0)|_0 - \alpha_0|\mu^{n_i}(0)|\big) + \big(|x|_0 - \alpha_0|x|\big)\Big|$$

$$\leq \sum_{i=1}^{r} \Big||\mu^{n_i}(0)|_0 - \alpha_0|\mu^{n_i}(0)|\Big| + \Big||x|_0 - \alpha_0|x|\Big|$$

$$< \frac{\epsilon}{2}\sum_{i=1}^{r}|\mu^{n_i}(0)| + |x| < \begin{cases} \frac{\epsilon}{2}|w| + |\mu^G(0)| & \text{if } |x| \leq |\mu^G(0)|, \\ \frac{\epsilon}{2}|w| + c|\mu^{n_r}(1)| & \text{if } x \trianglelefteq \mu^{n_r}(1^c). \end{cases}$$

Thus

$$\Big|\frac{|w|_0}{|w|} - \alpha_0\Big| < \begin{cases} \frac{\epsilon}{2} + \frac{|\mu^G(0)|}{|w|} \\ \frac{\epsilon}{2} + c\frac{|\mu^{n_r}(1)|}{|w|} \end{cases} \leq \begin{cases} \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ \frac{\epsilon}{2} + c\frac{|\mu^{n_r}(1)|}{|\mu^{n_r}(0)|} \end{cases} \leq \begin{cases} \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ \frac{\epsilon}{2} + c\frac{\epsilon}{2c} \end{cases} = \begin{cases} \epsilon, \\ \epsilon. \end{cases}$$

This completes the proof.

Proposition 3. *Assume $a > d$. Then the frequency of letters exists in $\eta^\omega(0)$.*

Proof. We proceed as in Proposition 2. We shall prove that 0 occurs in $\eta^\omega(0)$ with the frequency $\alpha_0 = \lim_{n\to\infty} |\eta^n(0)|_0/|\eta^n(0)|$.

Let $\epsilon > 0$. We will show that, for every sufficiently long prefix w of $\mu^\omega(0)$,

$$\left|\frac{|w|_0}{|w|} - \alpha_0\right| < \epsilon.$$

Let m_1 be an integer such that

$$\Big||\eta^n(0)|_0 - \alpha_0|\eta^n(0)|\Big| < \frac{\epsilon}{3a}|\eta^n(0)|$$

for all $n \geq m_1$.

Next, suppose $k_1 > k_2 > \cdots > k_l \geq 1$. Then, since $a > d$, we see that

$$\frac{\sum_{i=1}^{l}|\eta^{k_i}(1)|}{\sum_{i=1}^{l}|\eta^{k_i}(0)|} \leq \frac{\sum_{i=1}^{l} d^{k_i}}{\sum_{i=1}^{l} a^{k_i}} \leq \sum_{i=1}^{l}\left(\frac{d}{a}\right)^{k_i} \leq \sum_{j=k_l}^{\infty}\left(\frac{d}{a}\right)^{j} = \left(\frac{d}{a}\right)^{k_l}\frac{a}{a-d} < \frac{\epsilon}{3c},$$

whenever $k_l \geq m_2$ for some positive integer m_2. Note that if $n \geq m_2$, then

$$\frac{|\eta^n(1)|}{|\eta^n(0)|} < \left(\frac{d}{a}\right)^n < \frac{\epsilon}{3c}.$$

Now denote $G = \max\{m_1, m_2, 1\}$. Let w be a prefix of $\eta^\omega(0)$ of length

$$|w| > \frac{3|\eta^G(0)|}{\epsilon}.$$

As in Lemma 6, we write

$$w = \eta^{n_1}\big(01^c0^{m_1}\big)\eta^{n_2}\big(01^c0^{m_2}\big)\cdots\eta^{n_r}\big(01^c0^{m_r}\big)x,$$

where $r \geq 0$, $n_1 > n_2 > \cdots > n_r \geq G$, $0 \leq m_1, m_2, \ldots, m_r \leq a-1$, and either

$$|x| \leq |\eta^G(0)| \qquad \text{or} \qquad \eta^{n_{r+1}}(0) \lhd x \trianglelefteq \eta^{n_{r+1}}(01^c) \tag{12}$$

with $n_r > n_{r+1} \geq G$. Then

$$\begin{aligned}
\Big||w|_0 - \alpha_0|w|\Big| &= \Big|\sum_{i=1}^{r}\big(|\eta^{n_i}(01^c0^{m_i})|_0 - \alpha_0|\eta^{n_i}(01^c0^{m_i})|\big) + \big(|x|_0 - \alpha_0|x|\big)\Big| \\
&= \Big|\sum_{i=1}^{r}(m_i+1)\big(|\eta^{n_i}(0)|_0 - \alpha_0|\eta^{n_i}(0)|\big) - \alpha_0\sum_{i=1}^{r}c|\eta^{n_i}(1)| + \big(|x|_0 - \alpha_0|x|\big)\Big| \\
&\leq \sum_{i=1}^{r}(m_i+1)\Big||\eta^{n_i}(0)|_0 - \alpha_0|\eta^{n_i}(0)|\Big| + \alpha_0\sum_{i=1}^{r}c|\eta^{n_i}(1)| + \big||x|_0 - \alpha_0|x|\big| \\
&\leq \sum_{i=1}^{r}a\Big||\eta^{n_i}(0)|_0 - \alpha_0|\eta^{n_i}(0)|\Big| + c\sum_{i=1}^{r}|\eta^{n_i}(1)| + \big||x|_0 - \alpha_0|x|\big| \\
&\leq \frac{\epsilon}{3}\sum_{i=1}^{r}|\eta^{n_i}(0)| + c\sum_{i=1}^{r}|\eta^{n_i}(1)| + \big||x|_0 - \alpha_0|x|\big|.
\end{aligned} \tag{13}$$

We analyze the two cases in (12) separately. First, assume that $|x| \leq |\eta^G(0)|$. Then

$$\begin{aligned}
\Big|\frac{|w|_0}{|w|} - \alpha_0\Big| &\leq \frac{\epsilon}{3}\frac{\sum_{i=1}^{r}|\eta^{n_i}(0)|}{|w|} + c\frac{\sum_{i=1}^{r}|\eta^{n_i}(1)|}{|w|} + \frac{\big||x|_0 - \alpha_0|x|\big|}{|w|} \\
&\leq \frac{\epsilon}{3}\frac{\sum_{i=1}^{r}|\eta^{n_i}(0)|}{\sum_{i=1}^{r}|\eta^{n_i}(0)|} + c\frac{\sum_{i=1}^{r}|\eta^{n_i}(1)|}{\sum_{i=1}^{r}|\eta^{n_i}(0)|} + \frac{|x|}{|w|} \\
&\leq \frac{\epsilon}{3} + c\frac{\epsilon}{3c} + \frac{|\eta^G(0)|}{|w|} \leq \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon.
\end{aligned}$$

Next, assume that

$$\eta^{n_{r+1}}(0) \lhd x \trianglelefteq \eta^{n_{r+1}}(01^c)$$

with $n_{r+1} \geq G$. Then since $n_{r+1} \geq G$,

$$\begin{aligned}
\big||x|_0 - \alpha_0|x|\big| &\leq \Big||\eta^{n_{r+1}}(0)|_0 - \alpha_0|\eta^{n_{r+1}}(0)|\Big| + \alpha_0|\eta^{n_r+1}(1^c)| \\
&\leq \frac{\epsilon}{3}|\eta^{n_{r+1}}(0)| + c|\eta^{n_{r+1}}(1)|,
\end{aligned}$$

and therefore by (13),

$$\Big||w|_0 - \alpha_0|w|\Big| \leq \frac{\epsilon}{3}\sum_{i=1}^{r+1}|\eta^{n_i}(0)| + c\sum_{i=1}^{r+1}|\eta^{n_i}(1)|.$$

Finally, as above, we get

$$\begin{aligned}\Big|\frac{|w|_0}{|w|}-\alpha_0\Big| &\le \frac{\epsilon}{3}\frac{\sum_{i=1}^{r+1}|\eta^{n_i}(0)|}{|w|}+c\frac{\sum_{i=1}^{r+1}|\eta^{n_i}(1)|}{|w|}\\ &\le \frac{\epsilon}{3}\frac{\sum_{i=1}^{r+1}|\eta^{n_i}(0)|}{\sum_{i=1}^{r+1}|\eta^{n_i}(0)|}+c\frac{\sum_{i=1}^{r+1}|\eta^{n_i}(1)|}{\sum_{i=1}^{r+1}|\eta^{n_i}(0)|}\\ &\le \frac{\epsilon}{3}+c\frac{\epsilon}{3c}<\epsilon.\end{aligned}$$

This concludes our proof.

Theorem 1. *The frequency of letters exists in $\varphi^\omega(0)$. Furthermore, the frequency of 0 is α_0, and the frequency of 1 is α_1, where*

$$\alpha_0=\begin{cases}0 & \text{if } a\le d,\\ \dfrac{a-d}{a-d+c} & \text{if } a>d,\end{cases}\qquad \text{and}\qquad \alpha_1=\begin{cases}1 & \text{if } a\le d,\\ \dfrac{c}{a-d+c} & \text{if } a>d.\end{cases}$$

Proof. We have proved the existence of the frequency in Proposition 1 when $a \le d$. In the proof of the proposition, we saw that the frequency of 0 is 0 and the frequency of 1 is 1.

Now assume $a > d$. Referring to Lemma 4, let

$$\alpha_0=\lim_{n\to\infty}\frac{|\varphi^n(0)|_0}{|\varphi^n(0)|}.$$

By Lemmas 2 and 1,

$$f_\eta(k)\le f_\varphi(k)\le f_\mu(k)$$

for all $k \ge 1$. Therefore, by Propositions 2 and 3,

$$\Big|\frac{f_\varphi(k)}{k}-\alpha_0\Big|\le\max\left\{\Big|\frac{f_\mu(k)}{k}-\alpha_0\Big|,\Big|\frac{f_\eta(k)}{k}-\alpha_0\Big|\right\}\longrightarrow 0$$

as $k \longrightarrow \infty$, and thus α_0 is the frequency of 0.

The formulas for α_0 and α_1 in the statement come from (6) and from the identity $\alpha_1 = 1 - \alpha_0$. This completes the proof.

Remark 2. Suppose φ is primitive and its associated incidence matrix is

$$M(\varphi)=\begin{pmatrix}a & b\\ c & d\end{pmatrix}.$$

The primitivity of φ implies that $b, c \neq 0$. If φ is prolongable in 0, then by the result of Michel, the frequency of letters 0 and 1 in $\varphi^\omega(0)$ exists and equals, respectively,

$$\frac{r-d}{r-d+c}\qquad\text{and}\qquad\frac{c}{r-d+c},\tag{14}$$

where

$$r = \frac{a + d + \sqrt{(a-d)^2 + 4bc}}{2}.$$

This formula can be found in more general form from [2, Theorem 8.4.7]. Using the formulas in Theorem 1, it is easy to see that (14) holds true even if φ is non-primitive.

Acknowledgments

The author is grateful to Prof. Jeffrey Shallit for his comments and support, and to the School of Computing at the University of Waterloo for the hospitality.

References

1. Allouche, J.-P.; Davison, J. L.; Queffélec, M.; Zamboni, L. Q. Transcendence of Sturmian or morphic continued fractions. *J. Number Theory* **91** (2001), no. 1, 39–66.
2. Allouche, J-P. and Shallit, J. *Automatic Sequences*, Cambridge, (2003).
3. Cobham, A. Uniform tag sequences. *Math. Systems Theory* **6** (1972), 164–192.
4. Michel, P. Sur les ensembles minimaux engendrés par les substitutions de longueur non constante. Ph. D. Thesis, Université de Rennes, (1975).
5. Michel, P. Stricte ergodicité d'ensembles minimaux de substitution. In J.-P. Conze and M. S. Keane, editors, *Théorie Ergodique: Actes des Journées Ergodiques, Rennes 1973/1974*, Vol. 532 of Lecture Notes in Mathematics, pp. 189-201. Springer-Verlag, (1976)

Author Index