Rudolf Freund, Michal Hospodár, Galina Jirásková,
and Giovanni Pighizzini (eds.)

Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)

Short Papers

Druck: Druckerei Riegelnik
1080 Wien, Piaristengasse 19

Preface

This volume contains the five short contributions of the *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)* held in Košice, Slovakia, on August 21st and 22nd, 2018. The NCMA workshop series was established in 2009 as an annual forum for researchers working on different aspects of non-classical and classical models of automata and grammars. The purpose of the NCMA workshop series is to provide an opportunity to exchange and develop novel ideas, and to stimulate research on non-classical and classical models of automata and grammar-like structures. Many models of automata and grammars are studied from different points of view in various areas, both as theoretical concepts and as formal models for applications. The goal of the NCMA workshop series is to motivate a deeper coverage of this particular area and in this way to foster new insights and substantial progress in computer science as a whole.

The previous workshops took place in the following places:

2009	Wrocław, Poland,
2010	Jena, Germany,
2011	Milano, Italy,
2012	Fribourg, Switzerland,
2013	Umeå, Sweden,
2014	Kassel, Germany,
2015	Porto, Portugal,
2016	Debrecen, Hungary, and
2017	Praha, Czech Republic.

The Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018) was organized by the Košice branch of the Mathematical Institute of the Slovak Academy of Sciences. Its scientific program consisted of invited lectures, regular contributions, and short presentations.

In addition to the two invited talks and the 11 regular contributions, NCMA 2018 also features five short presentations to emphasize its workshop character, each of them also having been evaluated by at least two members of the program committee. The extended abstracts of these short presentations are contained in this volume.

We are grateful to the Košice branch of the Mathematical Institute of the Slovak Academy of Sciences for the local organization and for the financial support of NCMA 2018.

August 2018

Rudolf Freund, Wien
Michal Hospodár, Košice
Galina Jirásková, Košice
Giovanni Pighizzini, Milano

Table of Contents

RECOGNITION OF UNCOUNTABLY MANY LANGUAGES WITH ONE COUNTER

Maksims Dimitrijevs[(A,B)] Abuzer Yakaryılmaz[(A,B)]

[(A)]University of Latvia, Faculty of Computing
Raiņa bulvāris 19, Rīga, LV-1586, Latvia

[(B)]University of Latvia, Center for Quantum Computer Science
Raiņa bulvāris 19, Rīga, LV-1586, Latvia

md09032@lu.lv abuzer@lu.lv

Abstract
Recently, we investigated the minimal cases for realtime probabilistic machines that can define uncountably many languages with bounded error, and, we left open whether one-counter is sufficient. In this short paper, we answer this question by providing a positive answer.

Keywords: *Probabilistic automata, realtime computation, counter automata, bounded error.*

1. Introduction

Probabilistic and quantum machines can recognize uncountably many languages with bounded error when using real number transitions [1, 2, 3, 9]. We have been systematically investigating different bounded-error probabilistic models with various restrictions on computational resources that can recognize uncountably many languages [2, 3]. Regarding the realtime reading mode, we have shown that the following probabilistic realtime machines can recognize uncountably many languages with bounded error [3, 5]:

- Unary logarithmic-space probabilistic Turing machines (PTMs).
- Unary probabilistic automata with two counters (P2CAs).
- Unary probabilistic automata with k counters (PkCAs) in $O(\sqrt[k-1]{n})$ space, where $k > 2$.
- Binary double-logarithmic-space PTMs.
- Binary probabilistic automata with many counters in $O(\sqrt[k]{\log n})$ space for any $k \geq 1$, where the required number of counters depends on k.
- Binary P2CAs in $O(\sqrt[k]{n})$ space for any integer $k > 1$.

We have also shown that most of the presented bounds are tight. Uncountably many unary languages cannot be recognized with one stack in realtime [8], and so, using two counters is the minimal requirement. Realtime $o(\log \log n)$-space PTMs can recognize only regular languages [7]. Realtime counter automata with $o(\sqrt[k]{\log n})$ space for some $k \geq 1$ also can recognize only regular languages [5].

It is a known fact that probabilistic automata can recognize only regular languages in polynomial time [6]. On the other hand, we have proved possible sublinear space bound for realtime P2CAs that recognize uncountably many languages with bounded error. The case of realtime probabilistic automata with one counter was left open. In this paper we show that in non-unary case for realtime probabilistic automata it is sufficient to have one counter to recognize uncountably many languages with bounded error.

In the next section, we present the notations and definitions to follow the rest of the paper. Then, we present our results in Section 3.

2. Background

We assume that the reader is familiar with the basics of complexity theory and automata theory. We denote the left and the right end-markers as ¢ and \$, and the blank symbol as #. Σ not containing symbols ¢ and \$ denotes the input alphabet, $\tilde{\Sigma}$ is the set $\Sigma \cup \{¢, \$\}$. Σ^* is set of all strings (including the empty string (ε)) defined over Σ. For any natural number $i > 0$, $bin(i)$ denotes the unique binary representation that always starts with digit 1.

Our realtime models operate in strict mode: any given input, say $w \in \Sigma^*$, is read as $\tilde{w} = ¢w\$$ from the left to the right and symbol by symbol without any pause on any symbol.

Formally, a realtime probabilistic counter automaton with k counters (PkCA) P is a 6-tuple

$$P = (S, \Sigma, \delta, s_1, s_a, s_r),$$

where S is the set of finite internal states, $s_1 \in S$ is the initial state, $s_a \in S$ and $s_r \in S$ ($s_a \neq s_r$) are the accepting and rejecting states, respectively, and δ is the transition function

$$\delta : S \times \tilde{\Sigma} \times \{0,1\}^k \times S \times \{-1,0,1\}^k \to [0,1]$$

that governs the behavior of P as follows: When P is in state $s \in S$, reads symbol $\sigma \in \tilde{\Sigma}$ on the input tape, and checks the status of each of k counters $c^k = \{0,1\}^k$, whether the value of the counter is zero ($c = 0$) or not ($c = 1$), it enters state $s' \in S$ and updates the value of each counter with value from $d^k = \{-1,0,1\}^k$ with probability

$$\delta(s, \sigma, c^k, s', d^k),$$

where $d = -1$ ($d = 0$ and $d = 1$) means the value of the counter is decreased by one (the value of the counter remains unchanged and the value of the counter is increased by one).

To be a well-formed $PkCA$, the following condition must be satisfied: for each triple $(s, \sigma, c^k) \in S \times \tilde{\Sigma} \times \{0,1\}^k$,

$$\sum_{s' \in S, d^k \in \{-1,0,1\}^k} \delta(s, \sigma, c^k, s', d^k) = 1.$$

The computation starts in state s_1, and any given input, say $w \in \Sigma^*$, is read as $\cent w\$$ symbol by symbol from the left to the right, and the computation is terminated and the given input is accepted (rejected) if P enters s_a (s_r). It must be guaranteed that the machine enters a halting state after reading \$.

Language $L \subseteq \Sigma^*$ is said to be recognized by a PkCA P with error bound ϵ if

- each member is accepted by P with probability at least $1 - \epsilon$, and,
- each non-member is rejected by P with probability at least $1 - \epsilon$.

We can also say that L is recognized by P with bounded error or recognized by bounded-error PkCA P.

A PkCA with a working tape instead of counters is called probabilistic Turing machine (PTM). The working tape contains only blank symbols at the beginning of the computation and it has a two-way read/write head. On the work tape, a PTM reads the symbol under the head as a part of a transition, and then, it overwrites the symbol under the head and updates the position of head by at most one square after the transition.

A language L is recognized by a bounded-error PkCA in space $s(n)$, if the maximum absolute value of any of the counters is not more than $s(n)$ for any input with length n. In case of PTM, $s(n)$ is the maximum space used by PTM on a given input - the number of all cells visited on the work tape during the computation with some non-zero probability.

We denote the set of integers $\mathbb{Z}$ and the set of positive integers $\mathbb{Z}^+$. The set $\mathcal{I} = \{I \mid I \subseteq \mathbb{Z}^+\}$ is the set of all subsets of positive integers and so it is an uncountable set (the cardinality is $\aleph_1$) like the set of real numbers ($\mathbb{R}$). The cardinality of $\mathbb{Z}$ or $\mathbb{Z}^+$ is $\aleph_0$ (countably many).

For $I \in \mathcal{I}$, the membership of each positive integer is represented as a binary probability value:

$$p_I = 0.x_1 01 x_2 01 x_3 01 \cdots x_i 01 \cdots, \quad x_i = 1 \leftrightarrow i \in I.$$

The coin landing on head with probability p_I is named $\mathfrak{coin}_I$.

3. Realtime Automata with One Counter

We use a fact presented in our previous paper [2].

Fact 1 *[2] Let $x = x_1 x_2 x_3 \cdots$ be an infinite binary sequence. If a biased coin lands on head with probability $p = 0.x_1 01 x_2 01 x_3 01 \cdots$, then the value x_k is determined correctly with probability* *at least $\frac{3}{4}$ after 64^k coin tosses, where x_k is guessed as the $(3k+3)$-th digit of the binary number representing the total number of heads after 64^k coin tosses.*

We proceed with the possibility to improve the error bound when compute the bit x_k. We present the explicit proof, which is similar to the proof of Fact 1.

Lemma 3.1 *Let $x = x_1x_2x_3\cdots$ be an infinite binary sequence. If a biased coin lands on head with probability $p = 0.x_101x_201x_301\cdots$, then the value x_k can be determined with probability at least $1 - \frac{1}{4\cdot 2^l}$ after $64^k \cdot 2^l$ coin tosses, where $l > 0$.*

Proof. Let X be the random variable denoting the number of heads after $64^k \cdot 2^l$ coin tosses. The expected value of X is $E[X] = p \cdot 64^k \cdot 2^l$. The value of x_k is equal to $(3\cdot k + l + 3)$-th bit in $E[X]$.

If $|X - E[X]| \leq 8^k \cdot 2^l$ we still have the correct x_k since in $E[X]$ $(= x_101x_201x_301\cdots x_k01\cdots)$ x_k01 is followed by $3k+l$ bits and if we add a number in the interval $[-8^k\cdot 2^l, 8^k \cdot 2^l]$ to $E[X]$, we can get a number between

$$x_101x_201x_301\cdots x_k00\cdots \text{ and } x_101x_201x_301\cdots x_k10\cdots.$$

By using this fact with Chebyshev's inequality, we can follow that

$$Pr[|X - E[X]| \geq 8^k \cdot 2^l] \leq \frac{p\cdot(1-p)\cdot 64^k \cdot 2^l}{(8^k\cdot 2^l)^2} = \frac{p\cdot(1-p)\cdot 64^k\cdot 2^l}{64^k \cdot 2^{2l}} = \frac{p\cdot(1-p)}{2^l},$$

where the function $p\cdot(1-p)$ is parabolic and its global maximum is $\frac{1}{4}$, i.e. $p\cdot(1-p) \leq \frac{1}{4}$ for any chosen probability p.

Therefore, by returning the $(3k+l+3)$-th digit of the counter value that keeps the number of heads after $64^k \cdot 2^l$ coin tosses, we can correctly guess x_k with the probability at least $1 - \frac{1}{4\cdot 2^l}$. □

We use the following nonregular binary language, a modified version of `DIMA` [2]:

$$\texttt{DIMA3}_l = \{0^{2^0}10^{2^1}10^{2^2}1\cdots 10^{2^{6k+l-2}}110^{2^{6k+l-1}}11^{2^{6k+l}}(0^{2^{3k+l}-1}1)^{2^{3k}} \mid k > 0\},$$

where $l > 0$ is a even constant that influences the error bound.

Theorem 3.2 *Realtime probabilistic automata with one counter can recognize uncountably many languages with bounded error.*

Proof. Let w_k be the k-th shortest member of $\texttt{DIMA3}_l$ for $k > 0$. For any $I \in \mathcal{I}$, we define the following language:

$$\texttt{DIMA3}_l(I) = \{w_k \mid k > 0 \text{ and } k \in I\}.$$

Now we proceed with the recognition of $\texttt{DIMA3}_l(I)$ for any $I \in \mathcal{I}$. Let P be the P1CA and w be the given input of the form

$$w = 0^{t_1}10^{t_2}1\cdots 10^{t_{m-1}}110^{t_m}11^{t'_0}0^{t'_1}10^{t'_2}1\cdots 10^{t'_n}1,$$

where $t_1 = 1$, m and n are positive integers, $m - l$ is divisible by 6, and $t_i, t'_j > 0$ for $1 \leq i \leq m$ and $0 \leq j \leq n$. (Otherwise, the input is rejected deterministically.)

P splits computation into five paths with equal probabilities. In the first path, with the help of the counter, P makes the following comparisons:

- for each $i \in \{1, \ldots, \frac{m}{2}\}$, whether $2t_{2i-1} = t_{2i}$,
- for each $j \in \{1, \ldots, \frac{n}{2}\}$, whether $t'_{2j-1} = t'_{2j}$.

In the second path, with the help of the counter, P makes the following comparisons:

- for each $i \in \{1, \ldots, \frac{m}{2} - 1\}$, whether $2t_{2i} = t_{2i+1}$,
- whether $2t_m = t'_0$ (this also helps to set the counter to 0 for the upcoming comparisons),
- for each $j \in \{1, \ldots, \frac{n}{2} - 1\}$, whether $t'_{2j} = t'_{2j+1}$.

In the third path, P checks whether $1 + \sum_{i=1}^{m} t_i = n + \sum_{j=1}^{n} t'_j$. In the fourth path P checks, whether $\frac{t'_1+1}{2^l} = n$.

It is easy to see that all comparisons are successful if and only if $w \in \mathtt{DIMA3}_l$. If check in the path is not successful, the input is rejected. Otherwise, P finishes to read the input and accepts it with probability $\frac{5}{9}$, and rejects it with the remaining probability $\frac{4}{9}$.

In the fifth path, P tosses $\mathtt{coin}_I$ $T = 1 + \sum_{i=1}^{m} t_i$ times by reading the part of the input $0^{t_1}10^{t_2}1 \cdots 10^{t_{m-1}}110^{t_m}1$. Remark that if $w \in \mathtt{DIMA3}_l$, T is $64^k \cdot 2^l$ for some $k > 0$. After each coin toss, if the result is a head, P increases the value of the counter by one. Let H be the total number of heads, therefore, the value of the counter is H. Then P reads H symbols from the part $w'_k = (0^{2^{3k+l}-1}1)^{2^{3k}}$ with the help of the counter. During attempt to read H symbols, if the input is finished, then P rejects the input in this path. Otherwise, P guesses the value x_k with probability at least $1 - \frac{1}{4 \cdot 2^l}$. If the guess is 1, P accepts the input with probability $\frac{5}{9}$, and rejects the input with probability $\frac{4}{9}$. If the guess is 0, P rejects the input.

When P reads H symbols from the part $w'_k = (0^{2^{3k+l}-1}1)^{2^{3k}}$, it guesses the value x_k. Here we use the analysis similar to one presented in [4]. We can write H as

$$H = i \cdot 8^{k+1} \cdot 2^l + j \cdot 8^k \cdot 2^l + q = (8i + j)8^k \cdot 2^l + q,$$

where $i \geq 0$, $j \in \{0, \ldots, 7\}$, and $q < 8^k \cdot 2^l$.

Due to Lemma 3.1, x_k is the $(3k + l + 3)$-th digit of $bin(H)$ with probability $1 - \frac{1}{4 \cdot 2^l}$. In other words, x_k is guessed as 1 if $j \in \{4, \ldots, 7\}$, and as 0, otherwise. P sets $j = 0$ at the beginning. We can say that for each head, it consumes a symbol from w'_k. After reading $8^k \cdot 2^l$ symbols, it updates j as $(j + 1) \mod 8$. When the value of the counter reaches zero, P guesses x_k by checking the value of j.

If $w \in \mathtt{DIMA3}_l(I)$, then the input accepted with probability at least $4 \cdot \frac{1}{5} \cdot \frac{5}{9} + \frac{1}{5} \cdot \frac{5}{9} \cdot (1 - \frac{1}{4 \cdot 2^l}) = \frac{5}{9} - \frac{1}{36 \cdot 2^l}$.

If $w \notin \mathtt{DIMA3}_l$, the input is rejected with probability at least $\frac{1}{5} + 4 \cdot \frac{1}{5} \cdot \frac{4}{9} = \frac{5}{9}$.

If $w \in \texttt{DIMA3}_l$ and $w \notin \texttt{DIMA3}_l(I)$, the input is rejected with probability at least $4 \cdot \frac{1}{5} \cdot \frac{4}{9} + \frac{1}{5} \cdot (1 - \frac{1}{4 \cdot 2^l}) = \frac{5}{9} - \frac{1}{20 \cdot 2^l}$.

Therefore, the input is recognized with error bound $\epsilon = \frac{4}{9} + \frac{1}{20 \cdot 2^l}$, where $\epsilon < \frac{1}{2}$ for any $l \geq 0$, and ϵ can be arbitrarily close to $\frac{4}{9}$ for sufficiently large l. Since the cardinality of set $\{I \mid I \in \mathcal{I}\}$ is uncountable, there are uncountably many languages in $\{\texttt{DIMA3}_l(I) \mid I \in \mathcal{I}\}$, each of which is recognized by a bounded-error realtime P1CA. □

We conclude that our bounds for the number of counters are tight, i.e., one counter in the case of binary alphabets and two counters in the case of unary alphabets, since binary realtime probabilistic automata and unary realtime probabilistic automata with one counter can recognize only regular languages with bounded error [6, 8].

Acknowledgements

Dimitrijevs is partially supported by University of Latvia projects AAP2016/B032 “Innovative information technologies” and ZD2018/20546 “For development of scientific activity of Faculty of Computing”. Yakaryılmaz is partially supported by ERC Advanced Grant MQC.

References

[1] L. M. ADLEMAN, J. DEMARRAIS, M.-D. A. HUANG, Quantum computability. *SIAM Journal on Computing* 26 (1997) 5, 1524–1540.

[2] M. DIMITRIJEVS, A. YAKARYILMAZ, Uncountable classical and quantum complexity classes. In: H. BORDIHN, R. FREUND, B. NAGY, GY. VASZIL (eds.), *Eighth Workshop on Non-Classical Models for Automata and Applications (NCMA 2016)*. books@ocg.at 321, Österreichische Computer Gesellschaft, Wien, 2016, 131–146. (Also see: arXiv:1608.00417).

[3] M. DIMITRIJEVS, A. YAKARYILMAZ, Uncountable realtime probabilistic classes. In: G. PIGHIZZINI, C. CÂMPEANU (eds.), *Descriptional Complexity of Formal Systems - 19th IFIP WG 1.02 International Conference, DCFS 2017, Milano, Italy, July 3-5, 2017, Proceedings*. Lecture Notes in Computer Science 10316, Springer, 2017, 102–113. (Also see: arXiv: 1705.01773).

[4] M. DIMITRIJEVS, A. YAKARYILMAZ, Probabilistic verification of all languages. 2018. Technical report, arXiv:1807.04735.

[5] M. DIMITRIJEVS, A. YAKARYILMAZ, Uncountable realtime probabilistic classes. 2018. (Extended version, submitted to International Journal of Foundations of Computer Science).

[6] C. DWORK, L. STOCKMEYER, A time complexity gap for two-way probabilistic finite-state automata. *SIAM Journal on Computing* 19 (1990) 6, 1011–1123.

[7] R. FREIVALDS, Space and reversal complexity of probabilistic one-way Turing Machines. In: M. KARPINSKI (ed.), *Fundamentals of Computation Theory, Proceedings of the 1983 International FCT-Conference, Borgholm, Sweden, August 21-27, 1983*. Lecture Notes in Computer Science 158, Springer, 1983, 159–170.

[8] J. KAŅEPS, D. GEIDMANIS, R. FREIVALDS, Tally languages accepted by Monte Carlo pushdown automata. In: J. D. P. ROLIM (ed.), *Randomization and Approximation Techniques in Computer Science, International Workshop, RANDOM'97, Bologna, Italy, July 11–12. 1997, Proceedings*. Lecture Notes in Computer Science 1269, Springer, 1997, 187–195.

[9] A. C. C. SAY, A. YAKARYILMAZ, Magic coins are useful for small-space quantum machines. *Quantum Information & Computation* 17 (2017) 11&12, 1027–1043.

PARSING LANGUAGES OF P COLONY AUTOMATA

Erzsébet Csuhaj-Varjú$^{(A)}$ Kristóf Kántor$^{(B)}$
György Vaszil$^{(B)}$

$^{(A)}$Department of Algorithms and Their Applications
Faculty of Informatics, ELTE Eötvös Loránd University,
Pázmány Péter sétány 1/c, 1117 Budapest, Hungary
csuhaj@inf.elte.hu

$^{(B)}$Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
{kantor.kristof, vaszil.gyorgy}@inf.unideb.hu

Abstract
In this paper a subclass of generalized P colony automata is defined that satisfies a property which resembles the LL(k) property of context-free grammars The possibility of parsing the characterized languages using a k symbol lookahead, as in the LL(k) parsing method for context-free languages, is examined.

1. Introduction

The computational model called P colony is similar to tissue-like membrane systems. In P colonies, multisets of objects are used to describe the contents of cells and the environment. These multisets are processed by the cells in the corresponding colony using rules which enable the evolution of the objects present in the cells or the exchange of objects between the environment and the cells. These cells or computing agents have a very restricted functionality: they can store a limited amount of objects at a given time (the capacity of the cell) and thus they can process a limited amount of information. For more information on P colonies, consult summaries [12, 3].

P colony automata were introduced in [2]. They are called automata, since these variants of P colonies accept string languages by assuming an initial input tape with an input string in the environment. The available types of rules are extended by so-called tape rules. These types of rules in addition to processing the objects as their non-tape counterparts, also read the processed objects from the input tape.

Generalized P colony automata were introduced in [9] to overcome the difficulty that different tape rules can read different symbols in the same computational step. The main idea of this computational model was to get the process of input reading closer to other kinds of membrane systems, in particular to antiport P systems and P automata. The latter, introduced in [6] (see also [5]) are P systems using symport and antiport rules (see [13]), describing string languages. Generalized P colony automata were studied further in [11, 10].

A computation in this model defines accepted multiset sequences that are transformed into accepted symbol sequences/ strings. Generalized P colony automata have no input string, but there are tape rules and non-tape rules equally for evolution and communication rules. In a single computational step, this system is able to read more than one symbol, thus reading a multiset. This way generalized P colony automata are able to avoid the conflicts present in P colony automata, where simultaneous usage of tape rules in a single computational step can arise problems. After getting the result of a computation, that is, the accepted sequence of multisets, the sequence is mapped to a string in a similar way as shown in P automata.

In [9], some basic variants of the model were introduced and studied from the point of view of their computational power. In [11, 10] the investigations were continued by structuring the previous results around the capacity of the systems, and different types of restrictions imposed on the use of tape rules in the programs.

Since P colony automata variants accept languages, different types of descriptions of their language classes are of interest. One possible research direction is to investigate their parsing properties in terms of programs and rules of the (generalized) P colony automata. In this paper, we study the possibility of deterministically parsing the languages characterized by these devices. We define the so-called LL(k) condition for these types of automata, which enables deterministic parsing with a k symbol lookahead as in the case of context-free LL(k) languages. As an initial result, we show that using generalized P colony automata we can deterministically parse context-free languages that are not LL(k) in the "original" sense.

An extended version of this short paper has been submitted for publication, see [4].

2. Preliminaries and Definitions

Let V be a finite alphabet, let the set of all words over V be denoted by V^*, and let ε be the empty word. We denote the cardinality of a finite set S by $|S|$, and the number of occurrences of a symbol $a \in V$ in w by $|w|_a$.

A multiset over a set V is a mapping $M : V \to \mathbb{N}$ where $\mathbb{N}$ denotes the set of non-negative integers. This mapping assigns to each object $a \in V$ its multiplicity $M(a)$ in M. The set $supp(M) = \{a \mid M(a) \geq 1\}$ is the support of M. If V is a finite set, then M is called a finite multiset. A multiset M is empty if its support is empty, $supp(M) = \emptyset$. The set of finite multisets over the alphabet V is denoted by $\mathcal{M}(V)$. A finite multiset M over V will also be represented by a string w over the alphabet V with $|w|_a = M(a)$, $a \in V$, the empty multiset will be denoted by $\emptyset$.

A *genPCol automaton* of capacity k and with n cells, $k, n \geq 1$, is a construct

$$\Pi = (V, e, w_E, (w_1, P_1), \ldots, (w_n, P_n), F)$$

where

- V is an *alphabet*, the alphabet of the automaton, its elements are called *objects*;
- $e \in V$ is the *environmental object* of the automaton, the only object which is assumed to be available in an arbitrary, unbounded number of copies in the environment;
- $w_E \in (V - \{e\})^*$ is a string representing a multiset from $\mathcal{M}(V - \{e\})$, the multiset of objects different from e which is found in the environment initially;
- $(w_i, P_i), 1 \leq i \leq n$, specifies the i-th *cell* where w_i is (the representation of) a multiset over V, it determines the initial contents of the cell, and its cardinality $|w_i| = k$ is called the *capacity* of the system. P_i is a set of *programs*, each program is formed from k rules of the following types (where $a, b \in V$):
  - *tape rules* of the form $a \overset{T}{\rightarrow} b$, or $a \overset{T}{\leftrightarrow} b$, called rewriting tape rules and communication tape rules, respectively; or
  - *nontape rules* of the form $a \rightarrow b$, or $a \leftrightarrow b$, called rewriting (nontape) rules and communication (nontape) rules, respectively.

  A program is called a *tape program* if it contains at least one tape rule.
- F is a set of *accepting configurations* of the automaton which we will specify in more detail below.

A genPCol automaton reads an input word during a computation. A part of the input (possibly consisting of more than one symbol) is read during each configuration change: the processed part of the input corresponds to the multiset of symbols introduced by the tape rules of the system.

A *configuration* of a genPCol automaton is an $(n+1)$-tuple $(u_E, u_1, \ldots, u_n)$, where $u_E \in \mathcal{M}(V - \{e\})$ is the multiset of objects different from e in the environment, and $u_i \in \mathcal{M}(V)$, $1 \leq i \leq n$, are the contents of the i-th cell. The *initial configuration* is given by $(w_E, w_1, \ldots, w_n)$, the initial contents of the environment and the cells. The elements of the set F of *accepting configurations* are given as configurations of the form $(v_E, v_1, \ldots, v_n)$, where $v_E \in \mathcal{M}(V - \{e\})$ denotes a multiset of objects different from e being in the environment, and $v_i \in \mathcal{M}(V)$, $1 \leq i \leq n$, is the contents of the i-th cell.

Let $c = (u_E, u_1, \ldots, u_n)$ be a configuration of a genPCol automaton Π, and let $U_E = u_E \cup \{e, e, \ldots\}$, thus, the multiset of objects found in the environment (together with the infinite number of copies of e, denoted as $\{e, e, \ldots\}$, which are always present). The *sequence of programs*

$$(p_1, \ldots, p_n) \in (P_1 \cup \{\#\}) \times \ldots \times (P_n \cup \{\#\})$$

is *applicable in configuration* c, if the following conditions hold:

- The selected programs are applicable in the cells,

- the symbols to be brought inside the cells by the programs are present in the environment,
- the set of selected programs is maximal.

Let us denote the applicable sequences of programs in the configuration $c = (u_E, u_1, \dots, u_n)$ by App_c, that is,

$$\begin{aligned} App_c = \{P_c = (p_1, \dots, p_n) \in (P_1 \cup \{\#\}) \times \dots \times (P_n \cup \{\#\}) \mid \text{ where } P_c \\ \text{is a sequence of applicable programs in the configuration } c\}. \end{aligned}$$

A configuration c is called *a halting configuration* if the set of applicable sequences of programs is the singleton set $App_c = \{(p_1, \dots, p_n) \mid p_i = \# \text{ for all } 1 \le i \le n\}$.

Let $c = (u_E, u_1, \dots, u_n)$ be a configuration of the genPCol automaton. By applying a sequence of applicable programs $P_c \in App_c$, the configuration c is *changed* to a configuration $c' = (u'_E, u'_1, \dots, u'_n)$, denoted by $c \overset{P_c}{\Longrightarrow} c'$, if the following properties hold. (For a program p, we denote by $create(p)$, $import(p)$, and $export(p)$ the multisets of objects created by the program through rewriting, brought inside the cell from the environment, and sent out to the environment, respectively.)

- If $(p_1, \dots, p_n) = P_c \in App_c$ and $p_i \in P_i$, then $u'_i = create(p_i) \cup import(p_i)$, otherwise, if $p_i = \#$, then $u'_i = u_i$, $1 \le i \le n$. Moreover,
- $U'_E = U_E - \bigcup_{p_i \neq \#, 1 \le i \le n} import(p_i) \cup \bigcup_{p_i \neq \#, 1 \le i \le n} export(p_i)$ (where U'_E again denotes $u'_E \cup \{e, e, \dots\}$ with an infinite number of copies of e).

Thus, in genPCol automata, we apply the programs in the maximally parallel way, that is, in each computational step, every component cell nondeterministically applies one of its applicable programs. Then we collect all the symbols that the tape rules "read": this is the multiset read by the system in the given computational step.

For any P_c sequence of applicable programs in a configuration c, let us denote the multiset of objects read by the tape rules of the programs of P_c by $read(P_c)$. Then we can also define the set of multisets which can be read in any configuration of the genPCol automaton Π as

$$in_c(\Pi) = \{read(P_c) \mid P_c \in App_c\}.$$

Remark 2.1 *Although the set of configurations of a genPCol automaton Π can be infinite (because the multiset corresponding to the contents of the environment is not necessarily finite), the set $in_c(\Pi)$ is always finite.*

A successful computation defines this way an accepted sequence of multisets: $u_1 u_2 \dots u_s$, $u_i \in in_{c_{i-1}}(\Pi)$, for $1 \le i \le s$, that is, the sequence of multisets entering the system during the steps of the computation.

Let $\Pi = (V, e, w_E, (w_1, P_1), \dots, (w_n, P_n), F)$ be a genPCol automaton. The *set of input se-*

quences accepted by Π is defined as

$$A(\Pi) = \{u_1u_2\dots u_s \mid u_i \in in_{c_{i-1}}(\Pi),\ 1 \le i \le s,\ \text{and there is a configuration sequence } c_0,\dots,c_s,\ \text{with } c_0=(w_E,w_1,\dots,w_n),\ c_s \in F,\ c_s \text{ halting, and } c_i \overset{P_{c_i}}{\Longrightarrow} c_{i+1} \text{ with } u_{i+1} = read(P_{c_i}) \text{ for all } 0 \le i \le s-1\}.$$

Let Π be a genPCol automaton, and let $f : \mathcal{M}(V) \to 2^{\Sigma^*}$ be a mapping, such that $f(u) = \{\varepsilon\}$ if and only if u is the empty multiset.

The *language accepted by* Π with respect to f is defined as

$$L(\Pi, f) = \{f(u_1)f(u_2)\dots f(u_s) \in \Sigma^* \mid\ u_1u_2\dots u_s \in A(\Pi)\}.$$

Let V and Σ be two alphabets, and let $\mathcal{M}_{FIN}(V) \subseteq \mathcal{M}(V)$ denote the set of finite subsets of the set of finite multisets over an alphabet V. Consider a mapping $f : D \to 2^{\Sigma^*}$ for some $D \subseteq \mathcal{M}_{FIN}(V)$. We say that $f \in \mathcal{F}_{\text{TRANS}}$, if for any $v \in D$, we have $|f(v)| = 1$, and we can obtain $f(v) = \{w\}$, $w \in \Sigma^*$ by applying a deterministic finite transducer to any string representation of the multiset v (as w is unique, the transducer must be constructed in such a way that all string representations of the multiset v as input result in the same $w \in \Sigma^*$ as output, and moreover, as f should be nonerasing, the transducer produces a result with $w \neq \varepsilon$ for any nonempty input).

Besides the above defined class of mappings, we also use the so-called permutation mapping. Let $f_{perm} : \mathcal{M}(V) \to 2^{\Sigma^*}$ where $V = \Sigma$ be defined as follows. For all $v \in \mathcal{M}(V)$, we have

$$f_{perm}(v) = \{a_{\sigma(1)}a_{\sigma(2)}\dots a_{\sigma(s)} \mid v = a_1a_2\dots a_s \text{ for some permutation } \sigma\}.$$

3. P Colony Automata and the LL(k) Condition

Let $U \subset \Sigma^*$ be a finite set of strings over some alphabet Σ. Let us denote for some $k \ge 1$, the set of length k prefixes of the elements of U by $\text{FIRST}_k(U)$, that is, let

$$\text{FIRST}_k(U) = \{pref_k(u) \in \Sigma^* \mid u \in U\}$$

where $pref_k(u)$ denotes the string of the first k symbols of u if $|u| \ge k$, or $pref_k(u) = u$ otherwise.

Definition 3.1 Let $\Pi = (V, e, w_E, (w_1, P_1), \dots, (w_n, P_n), F)$ be a genPCol automaton, let $f : \mathcal{M}(V) \to 2^{\Sigma^*}$ be a mapping as above, and let $c_0, c_1, \dots, c_s$ be a sequence of configurations with $c_i \Longrightarrow c_{i+1}$ for all $0 \le i \le s-1$.

We say that the P colony Π is LL(k) for some $k \ge 1$ with respect to the mapping f, if for any two distinct sets of programs applicable in configuration c_s, $P_{c_s}, P'_{c_s} \in Acc_{c_s}$ with $P_{c_s} \neq P'_{c_s}$, the next k symbols of the input string that is being read determines which of the two sequences are to be applied in the next computational step, that is, the following holds.

Consider two computations

$$c_s \overset{P_{c_s}}{\Longrightarrow} c_{s+1} \overset{P_{c_{s+1}}}{\Longrightarrow} \dots \overset{P_{c_{s+m}}}{\Longrightarrow} c_{s+m+1}, \text{ and } c_s \overset{P'_{c_s}}{\Longrightarrow} c'_{s+1} \overset{P'_{c_{s+1}}}{\Longrightarrow} \dots \overset{P'_{c_{s+m'}}}{\Longrightarrow} c'_{s+m'+1}$$

where $u_{c_s} = read(P_{c_s})$ and $u_{c_s+i} = read(P_{c_s+i})$ for $1 \le i \le m$, and similarly $u'_{c_s} = read(P'_{c_s})$ and $u'_{c_s+i} = read(P_{c_s+i'})$ for $1 \le i \le m'$, thus, the two sequences of input multisets are

$$u_{c_s} u_{c_s+1} \dots u_{c_s+m} \text{ and } u'_{c_s} u'_{c_s+1} \dots u'_{c_s+m'}.$$

Assume that these sequences are long enough to "consume" the next k symbols of the input string, that is, for w and w' with

$$w \in f(u_{c_s}) f(u_{c_s+1}) \dots f(u_{c_s+m}) \text{ and } w' \in f(u'_{c_s}) f(u'_{c_s+1}) \dots f(u'_{c_s+m'}),$$

either $|w| \ge k$ and $|w'| \ge k$, or if $|w| < k$ (or $|w'| < k$), then c_{s+m+1} (or $c_{s+m'+1}$) is a halting configuration.

The P colony Π is LL(k), if for any two computations as above,

$$\mathrm{FIRST}_k(w) \cap \mathrm{FIRST}_k(w') = \emptyset.$$

Let us illustrate the above definition with an example.

Example 3.2 *Let* $\Pi = (\{a, b, c, d, f, g, e\}, e, \emptyset, (ea, P_1), F)$ *where*

$$\begin{aligned} P_1 = \{&\langle e \to b, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to e, b \overset{T}{\leftrightarrow} a\rangle, \langle e \to c, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to f, a \overset{T}{\leftrightarrow} e\rangle, \\ &\langle e \to d, c \overset{T}{\leftrightarrow} b\rangle, \langle b \to c, d \overset{T}{\leftrightarrow} e\rangle, \langle e \to g, f \overset{T}{\leftrightarrow} b\rangle, \langle b \to f, g \overset{T}{\leftrightarrow} e\rangle\} \text{ and} \\ F = \{&(v, ce), (v, fe) \mid v \in V^*, b \notin v\}. \end{aligned}$$

The language characterized by Π *is*

$$L(\Pi, f_{perm}) = \{a\} \cup \{(ab)^n a (cd)^n \mid n \ge 1\} \cup \{(ab)^n a (fg)^n \mid n \ge 1\}.$$

To see this, consider the possible computations of Π*. The initial configuration is* $(\emptyset, ea)$ *and there are three possible configurations that can be reached. Two of these are non-accepting states, but the derivations cannot be continued, so let us consider the third one (we denote by* $\Rightarrow_u$ *a configuration change during which the multiset of symbols* u *was read by the automaton).*

$$(a, be) \Rightarrow_b (b, ea) \Rightarrow_a (ba, be) \Rightarrow_b (bb, ea) \Rightarrow_a \dots \Rightarrow_b (b^i, ea).$$

At this point, the computation can follow two different paths again, either

$$(b^i, ae) \Rightarrow_a (b^i a, ec) \Rightarrow_c (b^{i-1} ac, db) \Rightarrow_d (b^{i-1} acd, ce) \Rightarrow_c \dots \Rightarrow_d (ac^i d^i, ce),$$

or

$$(b^i, ae) \Rightarrow_a (b^i a, ef) \Rightarrow_f (b^{i-1}af, gb) \Rightarrow_g (b^{i-1}afg, fe) \Rightarrow_f \ldots \Rightarrow_g (af^i g^i, fe).$$

In the first phase of the computation, the system produces copies of b and sends them to the environment, then in the second phase these copies of b are exchanged to copies of cd or copies of fg. The system can reach an accepting state when all the copies of b are used, that is, when an equal number of copies of ab and either of cd or of fg were produced.

Note that the system satisfies the LL(1) property, the symbol that has to be read, in order to accept a desired input word, determines the set of programs that has to be used in the next computational step.

Let us denote the class of context-free LL(k) languages by $\mathcal{L}$(CF,LL(k)) (see for example the monograph [1] for more details) and the languages characterized by genPCol automata satisfying the above defined condition with input mapping of type f_{perm} or $f \in TRANS$, as $\mathcal{L}_X$(genPCol,LL(k)), $X \in \{perm, TRANS\}$.

The following statement can be presented.

Theorem 3.3 *There are context-free languages in $\mathcal{L}_X$ (genPCol,LL(1)), $X \in \{perm, TRANS\}$, which are not in $\mathcal{L}$(CF,LL(k)) for any $k \geq 1$.*

Proof. The language $L(\Pi, f_{perm}) \in \mathcal{L}_{perm}$(genPCol,LL(1)) from Example 3.2 is not in $\mathcal{L}$(CF,LL(k)) for any $k \geq 1$. If we consider the mapping $f_1 \in TRANS$, $f_1 : \{a,b,c,d,f,g\} \to \{a,b,c,d,f,g\}$ with $f_1(x) = x$ for all $x \in \{a,b,c,d,f,g\}$, then $L(\Pi, f_1) = L(\Pi, f_{perm})$, thus, $\mathcal{L}_{TRANS}$(genPCol,LL(1)) also contains the non-LL(k) context-free language. □

4. Conclusions

P systems and their variants are able to describe powerful language classes, thus their applicability in the theory of parsing or analyzing syntactic structures are of particular interest, see, for example [7, 8]. In [7], so-called active P automata (P automata with dynamically changing membrane structure) were used for parsing, utilizing the dynamically changing membrane structure of the P automaton for analyzing the string.

In this paper we studied the possibility of deterministically parsing languages characterized by P colony automata. We provided the definition of an LL(k)-like property for (generalized) P colony automata, and showed that languages which are not LL(k) in the "original" context-free sense for any $k \geq 1$ can be characterized by LL(1) P colony automata with different types of input mappings. The properties of these language classes for different values of k and different types of input mappings are open to further investigations.

Acknowledgments

The work of E. Csuhaj-Varjú was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558. The work of K. Kántor and Gy. Vaszil was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558 and also by the construction EFOP-3.6.3-VEKOP-16-2017-00002, a project financed by the European Union, co-financed by the European Social Fund.

References

[1] A. V. AHO, J. D. ULMANN, *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Englewood Cliffs, N.J., 1973.

[2] L. CIENCIALA, L. CIENCIALOVÁ, E. CSUHAJ-VARJÚ, GY. VASZIL, PCol automata: Recognizing strings with P colonies. In: M. A. MARTÍNEZ DEL AMOR, GH. PĂUN, I. PÉREZ HURTADO, A. RISCOS NUÑEZ (eds.), *Eighth Brainstorming Week on Membrane Computing, Sevilla, February 1–5, 2010*. Fénix Editora, 2010, 65–76.

[3] L. CIENCIALA, L. CIENCIALOVÁ, E. CSUHAJ-VARJÚ, P. SOSÍK, P colonies. *Bulletin of the International Membrane Computing Society* 1 (2016) 2, 119–156.

[4] E. CSUHAJ-VARJÚ, K. KÁNTOR, GY. VASZIL, Deterministic parsing with P colony automata. Submitted.

[5] E. CSUHAJ-VARJÚ, M. OSWALD, GY. VASZIL, P automata. In: GH. PĂUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., 2010, 144–167.

[6] E. CSUHAJ-VARJÚ, GY. VASZIL, P automata or purely communicating accepting P systems. In: GH. PĂUN, G. ROZENBERG, A. SALOMAA, C. ZANDRON (eds.), *Membrane Computing, International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, August 19–23, 2002, Revised Papers*. Lecture Notes in Computer Science 2597, Springer, 2003, 219–233.

[7] G. B. ENGUIX, R. GRAMATOVICI, Parsing with active P automata. In: C. MARTÍN-VIDE, G. MAURI, GH. PĂUN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17–22, 2003, Revised Papers*. Lecture Notes in Computer Science 2933, Springer, 2004, 31–42.

[8] G. B. ENGUIX, B. NAGY, Modeling syntactic complexity with P systems: A preview. In: *Unconventional Computation and Natural Computation – 13th International Conference, UCNC 2014, London, ON, Canada, July 14-18, 2014, Proceedings*. Lecture Notes in Computer Science 8553, Springer, 2014, 54–66.

[9] K. KÁNTOR, GY. VASZIL, Generalized P colony automata. *Journal of Automata, Languages and Combinatorics (JALC)* 19 (2014) 1–4, 145–156.

[10] K. KÁNTOR, GY. VASZIL, Generalized P colony automata and their relation to P automata. In: M. GHEORGHE, G. ROZENBERG, A. SALOMAA, C. ZANDRON (eds.), *Membrane*

Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers. Lecture Notes in Computer Science 10725, Springer, 2018, 167–182.

[11] K. KÁNTOR, GY. VASZIL, On the classes of languages characterized by generalized P colony automata. *Theoretical Computer Science* 724 (2018), 35–44.

[12] A. KELEMENOVÁ, P colonies. In: GH. PĂUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., 2010, 584–593.

[13] A. PĂUN, GH. PĂUN, The power of communication: P systems with symport/antiport. *New Generation Computing* 20 (2002) 3, 295–306.

THE COMPLEXITY OF LANGUAGES RESULTING FROM THE CUT OPERATION IN THE UNARY CASE

Markus Holzer$^{(B)}$ Michal Hospodár$^{(A)}$

$^{(B)}$ Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
`holzer@informatik.uni-giessen.de`

$^{(A)}$ Mathematical Institute, Slovak Academy of Sciences,
Grešákova 6, 040 01 Košice, Slovakia
`hosmich@gmail.com`

Abstract
We investigate the state complexity of languages resulting from the cut operation of two unary regular languages represented by minimal deterministic finite automata with m and n states. We show that only complexities up to $2m-1$ and between n and $m+n-2$ can be attained, while if $2m \leq n-1$, then the complexities from $2m$ up to $n-1$ cannot be attained.

1. Introduction

It is well known that for every n-state nondeterministic finite automaton (NFA), there exists a language equivalent deterministic finite automaton (DFA) with at most 2^n states [23]. This bound is tight in the sense that for an arbitrary integer n there is always some n-state NFA which cannot be simulated by any DFA with less than 2^n states [19, 20, 21, 26].

Nearly two decades ago a very fundamental question on determinization was raised by Iwama, Kambayashi, and Takaki [10]: does there always exist a minimal n-state NFA whose equivalent minimal DFA has α states for all n and α with $n \leq \alpha \leq 2^n$? Iwama, Matsuura, and Paterson [11] called a number α in the range from n to 2^n *magic* if no minimal n-state NFA has an equivalent minimal α-state DFA. The simple question whether for every n no number is magic turned out to be harder than expected. In a series of papers, non-magic (attainable) numbers were identified [6, 12, 13] until the problem was solved in [15] showing that for ternary languages *no* magic numbers exist. On the contrary, magic numbers do exist for unary languages [5]. For binary languages, the original problem from [10] is still open.

$^{(A)}$Research supported by VEGA grant 2/0084/15 and grant APVV-15-0091. This work was conducted during a research visit at the Institut für Informatik, Universität Giessen, Germany, funded by the DAAD short-term grant ID 57314022.

The idea behind the magic number problem is not limited to the determinization of NFAs. In fact every (regularity preserving) formal language operation can be used to define a magic number problem for the operation in question. For instance, consider the intersection operation on languages. Let A and B be minimal finite automata with m and n states, respectively. Then the size of the minimal automaton for the intersection of $L(A)$ and $L(B)$ is between 1 and mn. The value one is induced by the intersection of disjoint languages and the value mn by the standard cross-product construction for the intersection operation. Thus, in a similar way as for the determinization, one may now ask, whether every α within the range between 1 and mn can be attained by the size of minimal automaton for intersection of languages given by two minimal automata with m and n states, respectively? In other words, is the outcome of the intersection operation in terms of the number of states contiguous or are there any gaps, hence magic numbers? In [9] it was shown that for the intersection on DFAs *no* number from 1 up to mn is magic—this already holds for binary automata. Besides intersection, also other formal language operations, for example, union [9], concatenation [14, 17], square [3], star [2, 16], and reversal [24] were investigated from the "magic number" perspective. It turned out that magic numbers are quite rare, and most of them occur in the unary case.

Geffert [5] investigated the state complexity of languages accepted by n-state unary NFAs. He proved that most of the numbers in the range from n up to $F(n) + n^2$, where $F(n)$ is the Landau function, is not attainable as the state complexity of a language accepted by a minimal unary n-state NFA. However, his proof is existential, and no specific value is known to be unattainable.

Van Zijl [25] examined the magic number problem for symmetric difference NFAs. She proved that in the range from 2^{n-1} up to 2^n, no value except for values $\operatorname{lcm}(2^{n_1}-1, 2^{n_2}-1, \ldots, 2^{n_k}-1)$ where $n = n_1 + n_2 + \cdots + n_k$, can be attained by the state complexity of a language accepted by a minimal unary n-state symmetric difference NFA.

Čevorová [2] studied the complexity of languages resulting from the Kleene star operation in the unary case. In such a case, the known upper bound is $(n-1)^2 + 1$ [27]. She proved that the values from 1 to n, as well as the values $n^2 - 2n + 2$ and $n^2 - 3n + 3$, are attainable, while the value $n^2 - 3n + 2$ is attainable if n is odd and it is not attainable otherwise. Moreover, she showed that all the values from $n^2 - 3n + 4$ up to $n^2 - 2n + 1$ and from $n^2 - 4n + 7$ up to $n^2 - 3n + 1$ cannot be attained by the state complexity of the Kleene star of any language accepted by minimal unary DFA with n states.

We contribute to the list of magic number problems for formal language operations by studying the cut operation in the unary case. The cut operation was introduced in [1] as a machine implementation of "concatenation" on UNIX text processors which behaves greedy like in its left term of concatenation. Tight upper bounds for the state complexity of the cut and iterated cut operations on DFAs were obtained in [4]. While the state complexity of concatenation is growing linearly with the number of states of the first automaton and exponentially with the number of states of the second automaton, the state complexity of the cut operation is only linearly growing with both parameters. In the unary case, the known tight upper bound is given by the function $f(m, n)$ such that $f(1, n) = 1$, $f(m, 1) = m$, $f(m, n) = 2m-1$ if $m, n \geq 2$ and $m \geq n$, and $f(m, n) = m + n - 2$ if $m, n \geq 2$ and $m < n$ [4].

In this paper, we show for every value from 1 up to $f(m, n)$ whether or not it can be attained by the state complexity of the cut of two languages accepted by minimal unary DFAs with m and n states. We show that only complexities up to $2m - 1$ and between n and $m + n - 2$ can be attained, while complexities from $2m$ up to $n - 1$ turn out to be magic. To get these results, the tail-loop structure of minimal unary DFAs is very valuable in the proofs. To the best of our knowledge, this is the first operation where for unary alphabet, every value in the range of possible complexities is known to be either attainable or not, and not all values are attainable.

2. Preliminaries

We recall some definitions on finite automata as contained in [7]. Let Σ^* denote the set of all words over a finite alphabet Σ. The *empty word* is denoted by ε. If u, v, w are words over Σ such that $w = uv$, then u is a *prefix* of w. Further, we denote the set $\{i, i+1, \ldots, j\}$ by $[i, j]$.

A *deterministic finite automaton* (DFA) is a quintuple $A = (Q, \Sigma, \delta, s, F)$ where Q is a finite nonempty set of *states*, Σ is a finite nonempty set of *input symbols*, $s \in Q$ is the *initial* state, $F \subseteq Q$ is the set of *final* (or *accepting*) states, and $\delta\colon Q \times \Sigma \to Q$ is the *transition function* which can be extended to the domain $Q \times \Sigma^*$ in the natural way. The *language accepted* (or *recognized*) by the DFA A is defined as $L(A) = \{\, w \in \Sigma^* \mid \delta(s, w) \in F \,\}$.

Two DFAs A and B are *equivalent* if they accept the same language, that is, if $L(A) = L(B)$. An automaton is *minimal* if it admits no smaller equivalent automaton with respect to the number of states. For DFAs this property can be verified by showing that all states are reachable from the initial state and all states are pairwise distinguishable. It is well known that every regular language has a unique, up to isomorphism, minimal DFA. The *state complexity* of a regular language is the number of states in the minimal DFA accepting this language.

In [1] the *cut operation* on languages K and L, denoted by $K \,!\, L$, is defined as

$$K \,!\, L = \{\, uv \mid u \in K,\ v \in L, \text{ and } uv' \notin K \text{ for every nonempty prefix } v' \text{ of } v \,\}.$$

The above defined cut operation preserves regularity as shown in [1]. Since we are interested in the descriptional complexity of this operation we briefly recall the construction of a DFA for the cut operation; we slightly deviate from the presentation of the construction given in [4].

Let $A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ and $B = (Q_B, \Sigma, \delta_B, s_B, F_B)$ be two DFAs. Let $\bot \notin Q_B$. Define the cut automaton $A \,!\, B = (Q, \Sigma, \delta, s, F)$ with the state set $Q = (Q_A \times \{\bot\}) \cup (Q_A \times Q_B)$, the initial state $s = (s_A, \bot)$ if $\varepsilon \notin L(A)$ and $s = (s_A, s_B)$ otherwise, the set of final states $F = Q_A \times F_B$, and for each state (p, q) in Q and each input a in Σ we have

$$\delta((p, \bot), a) = \begin{cases} (\delta_A(p, a), \bot), & \text{if } \delta_A(p, a) \notin F_A; \\ (\delta_A(p, a), s_B), & \text{otherwise;} \end{cases}$$

$$\delta((p, q), a) = \begin{cases} (\delta_A(p, a), \delta_B(q, a)), & \text{if } \delta_A(p, a) \notin F_A; \\ (\delta_A(p, a), s_B), & \text{otherwise.} \end{cases}$$

Then $L(A \,!\, B) = L(A) \,!\, L(B)$.

Consider the function f from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ defined by

$$f(m,n) = \begin{cases} 1, & \text{if } m = 1; \\ m, & \text{if } m \geq 2 \text{ and } n = 1; \\ 2m-1, & \text{if } m,n \geq 2 \text{ and } m \geq n; \\ m+n-2, & \text{if } m,n \geq 2 \text{ and } m < n. \end{cases} \tag{1}$$

It was proven in [4, Theorem 3.2] that if A and B are unary DFAs with m and n states, respectively, then $f(m,n)$ states are sufficient and necessary in the worst case for any DFA accepting the language $L(A)\,!\,L(B)$.

3. The Descriptional Complexity of the Cut Operation

In this section we investigate the range of attainable complexities for the cut operation on unary languages. We show that depending on m and n some values may be unattainable.

When working with unary DFAs, we use the notational convention proposed by Nicaud in [22]. Every unary DFA consists of a tail path, which starts from the initial state, followed by a loop of one or more states. Let $A = (Q, \{a\}, \delta, q_0, F)$ be a unary DFA with $|Q| = n$. We can identify the states of A with integers from $[0, n-1]$ via $q \mapsto \min\{\, i \mid \delta(q_0, a^i) = q \,\}$. In particular the initial state q_0 is mapped to 0. Let $\ell = \delta(q_0, a^n)$. Then the unary DFA A with n states, loop number ℓ ($0 \leq \ell \leq n-1$), and set of final states F ($F \subseteq [0, n-1]$) is referred to as $A = (n, \ell, F)$.

The following characterization of minimal unary DFAs is known.

Lemma 3.1 [22, Lemma 1] *A unary* DFA $A = (n, \ell, F)$ *is minimal if and only if*

(1) its loop is minimal, and

(2) if $\ell \neq 0$, then states $n-1$ and $\ell - 1$ do not have the same finality, that is, exactly one of them is final.

Now we are ready for our results on the cut operation of unary regular languages represented by DFAs. In a series of lemmata we consider the state complexity α of the resulting language in increasing order of α. The first interval we are going to discuss is $[1, m]$.

Lemma 3.2 *Let $m, n \geq 1$ and $1 \leq \alpha \leq m$. There exist minimal unary* DFA*s A and B with m and n states, respectively, such that the minimal* DFA *for $L(A)\,!\,L(B)$ has α states.*

Proof. The proof has five cases: (1) Let $m = 1$, so we must have $\alpha = 1$. Let A be the one-state DFA accepting the empty language and B be the minimal n-state DFA for $a^{n-1}a^*$.nThen $L(A)\,!\,L(B) = \emptyset$ which is accepted by a minimal one-state DFA.

(2) Let $m \geq 2$ and $n = 1$. Let A be the minimal m-state DFA for $a^{\alpha-1}(a^m)^*$ and B be one-state DFA for a^*. The reachable part of the cut automaton $A \,!\, B$ consists of the tail of non-final states $(i, \perp)$ with $0 \leq i \leq \alpha - 2$ and the loop of final states $(i, 0)$ with $0 \leq i \leq m - 1$. Since all the final states are equivalent, the minimal DFA for $L(A) \,!\, L(B)$ has α states.

(3) Let $m, n \geq 2$ and $\alpha = 1$. Consider the unary languages $a^{m-1}a^*$ and $a^{n-1}a^*$ accepted by minimal DFAs A and B of m and n states, respectively. Then the reachable part of the cut automaton $A \,!\, B$ consists of the tail of non-final states $(i, \perp)$ with $1 \leq i \leq m - 2$, and the loop consisting of a single non-final state $(m - 1, 0)$; notice that 0 is a non-final state in B. Hence $L(A) \,!\, L(B)$ is the empty language accepted by a one-state DFA.

(4) Let $m \geq 2$, $n = 2$, and $2 \leq \alpha \leq m$. Consider the unary languages K and L defined as follows. If $m-\alpha$ is even, then $K = \{\, a^{\alpha-2}, a^{m-2} \,\}$ and $L = a(aa)^*$, otherwise, $K = \{\, a^{\alpha-1}, a^{m-2} \,\}$ and $L = (aa)^*$. The minimal DFAs for K and L have m and 2 states, respectively. We have $K \,!\, L = a^{\alpha-1}(aa)^*$, which is accepted by a minimal α-state DFA.

(5) Let $m \geq 2$, $n \geq 3$, and $2 \leq \alpha \leq m$. Consider the unary deterministic finite automata $A = (m, \alpha-2, [\alpha - 1, m - 1])$ and $B = (n, n - 1, [0, n - 2])$. By Lemma 3.1, the DFAs A and B are minimal. The reachable part of the cut automaton consists of the tail of $\alpha - 1$ non-final states and of the loop of $m - \alpha + 2$ final states. Hence the minimal DFA for $L(A) \,!\, L(B)$ has α states. □

Our next interval is $[m + 1, 2m - 1]$; cf. $f(m, n)$ defined by (1) on page 28.

Lemma 3.3 *Let $m, n \geq 2$ and $m+1 \leq \alpha \leq 2m-1$. There exist minimal unary* DFA*s A and B with m and n states, respectively, such that the minimal* DFA *for $L(A) \,!\, L(B)$ has α states.*

Proof. We have $\alpha = m + \beta$ for some integer β with $1 \leq \beta \leq m - 1$. Consider the unary DFA $A = (m, 0, \{\beta\})$. Define the unary DFA B as follows:

$$B = \begin{cases} (n, 0, \{m - 1\}), & \text{if } m < n; \\ (n, n - 1, \{n - 1\}), & \text{otherwise.} \end{cases}$$

By Lemma 3.1, the DFAs A and B are minimal. If $m < n$, then $L(A) \,!\, L(B)$ is accepted by the DFA $(\alpha, \beta, \{\alpha - 1\})$, otherwise, it is accepted by the DFA $(\alpha, \beta, \{\, i \mid n + \beta - 1 \leq i \leq \alpha - 1 \,\})$. The resulting DFA is minimal by Lemma 3.1. □

The last interval we are considering in this series of lemmata is $[n, m + n - 2]$.

Lemma 3.4 *Let $m, n \geq 2$, $\alpha \geq m$, and $n \leq \alpha \leq m + n - 2$. There exist minimal unary* DFA*s A and B with m and n states, respectively, such that the minimal* DFA *for $L(A) \,!\, L(B)$ has α states.*

Proof. Consider the DFAs $A = (m, m - 1, \{m - 2\})$ and $B = (n, 0, \{\alpha - m + 1\})$ which are minimal by Lemma 3.1; notice that $1 \leq \alpha - m + 1 \leq n - 1$. In the cut automaton $A \,!\, B$, the states $(m-2, 0)$ and $(m-1, 0)$ are non-final and both of them are sent to $(m-1, 1)$ on a, hence they are equivalent. Next, for each i with $1 \leq i \leq n - \alpha + m - 2$, the states $(m - 2 - i, \perp)$

and $(m-1, n-i)$ are equivalent as well; notice that $n-\alpha+m-2 \geq 0$ since $\alpha \leq m+n-2$. To get the minimal DFA for the cut, we redirect the out-transition from the state $(m-1, \alpha-m+1)$ to the state $(\alpha-n, \perp)$ if $\alpha \leq m+n-3$ and to the state $(\alpha-n, 0)$ if $\alpha = m+n-2$. Since $m-1+(\alpha-m+1) = \alpha$, the resulting minimal DFA for $L(A) \,!\, L(B)$ has α states. □

For certain values of m and n the intervals stated in the previous lemmata may not be contiguous. For instance, if we choose $m = 2$ and $n = 5$, then the intervals from Lemmata 3.2, 3.3, and 3.4 cover $\{1, 2, 3, 5\}$. Hence the value 4, which comes from the interval $[2m, n-1]$, is missing. In fact, we show that whenever this interval is nonempty, these values cannot be obtained by an application of the cut operation on minimal DFAs with an appropriate number of states.

Lemma 3.5 *Let $m, n \geq 2$ and $2m \leq \alpha \leq n-1$. There exist* no *minimal unary m-state* DFA A *and minimal unary n-state* DFA B *such that the minimal* DFA *for $L(A) \,!\, L(B)$ has α states.*

Proof. We discuss two cases depending on whether the language $L(A)$ is infinite or finite.

If $L(A)$ is infinite, then A must have a final state in its loop. Denote the size of loop in A by ℓ and the smallest final state in the loop of A by j. Consider the cut automaton $A \,!\, B$. Notice that its initial state is sent to the state $(j, 0)$ by the word a^j. Next, the state $(j, 0)$ is sent to itself by the word a^ℓ. It follows that $A \,!\, B$ is equivalent to a DFA $(j+\ell, j, F)$ for some set $F \subseteq [0, j+l-1]$. Since $j \leq m-1$ and $\ell \leq m$, the minimal DFA for $L(A) \,!\, L(B)$ has at most $2m-1$ states.

If $L(A)$ is finite, then A has a loop in the non-final state $m-1$and the state $m-2$ is final. Let $A = (m, m-1, F)$ and $B = (n, k, F')$ be minimal unary DFAs for some sets $F \subseteq [0, m-1]$ and $F' \subseteq [0, n-1]$. It follows that in the cut automaton $A \,!\, B$, the state $(m-2, 0)$ and the states $(m-1, j)$ with $1 \leq j \leq n-1$ are reachable. Two distinct states $(m-1, j)$ and $(m-1, j')$ are distinguishable by the same word as the states j and j' in B, and the state $(m-2, 0)$ and a state $(m-1, j)$ are distinguishable by the same word as 0 and j are distinguishable in B. It follows that the cut automaton has at least n reachable and pairwise distinguishable states, and the theorem follows. □

Now let us summarize our results; recall that the state complexity of the cut operation on unary languages is given by the function $f(m, n)$ defined by (1) on page 28 such that $f(1, n) = 1$, $f(m, 1) = m$, $f(m, n) = 2m-1$ if $m, n \geq 2$ and $m \geq n$ and $f(m, n) = m+n-2$ if $m, n \geq 2$ and $m < n$.

Theorem 3.6 *For every $m, n, \alpha \geq 1$ such that $\alpha = 1$ if $m = 1$, $1 \leq \alpha \leq m$ if $m \geq 2$ and $n = 1$, or $1 \leq \alpha \leq 2m-1$ or $n \leq \alpha \leq m+n-2$ if $m, n \geq 2$, there* exist *minimal unary* DFA*s A and B with m and n states, respectively, such that the minimal* DFA *for $L(A) \,!\, L(B)$ has α states. In the case of $m, n \geq 2$ and $2m \leq \alpha \leq n-1$, there* do not exist *minimal unary m-state and n-state* DFA*s A and B such that the minimal* DFA *for $L(A) \,!\, L(B)$ has α states.*

Proof. The cases of $m = 1$ and $m \geq 2$, $n = 1$ are given by Lemma 3.2 (1) and Lemma 3.2 (2), respectively. Let $m, n \geq 2$. The case of $1 \leq \alpha \leq m$ is given by Lemma 3.2 (3)-(5). The case of $m+1 \leq \alpha \leq 2m-1$ is covered by Lemma 3.3, and the case of $n \leq \alpha \leq m+n-2$ by Lemma 3.4. We proved that no value from $2m$ to $n-1$ is attainable in Lemma 3.5. □

4. Conclusions

We examined the state complexity of languages resulting from the cut operation on minimal unary deterministic finite automata with m and n states. We proved that no value from $2m$ up to $n-1$ is attainable by the state complexity of the cut of two unary languages represented by minimal deterministic finite automata with m and n states. All the remaining values up to the known upper bound are attainable. This means that the problem of finding all attainable complexities for the cut operation is completely solved for unary alphabet. To the best of our knowledge, the cut operation is the first operation where the problem of finding all possible complexities in the unary case is completely solved.

Acknowledgements

We thank Juraj Šebej and Jozef Jirásek Jr. for their help on border values in our theorems. Moreover, also thanks to Galina Jirásková for her support and to all who helped us to improve the presentation of the paper.

References

[1] M. BERGLUND, H. BJÖRKLUND, F. DREWES, B. VAN DER MERWE, B. WATSON, Cuts in regular expressions. In: M. BÉAL, O. CARTON (eds.), *Developments in Language Theory - 17th International Conference, DLT, 2013, Marne-la-Vallée, France, June 18-21, 2013. Proceedings*. Lecture Notes in Computer Science 7907, Springer, 2013, 70–81.

[2] K. ČEVOROVÁ, Kleene star on unary regular languages. In: JÜRGENSEN and REIS [18], 2013, 277–288.

[3] K. ČEVOROVÁ, G. JIRÁSKOVÁ, I. KRAJŇÁKOVÁ, On the square of regular languages. In: HOLZER and KUTRIB [8], 2014, 136–147.

[4] F. DREWES, M. HOLZER, S. JAKOBI, B. VAN DER MERWE, Tight bounds for cut-operations on deterministic finite automata. *Fundamenta Informaticae* 155 (2017) 1-2, 89–110.

[5] V. GEFFERT, Magic numbers in the state hierarchy of finite automata. *Information and Computation* 205 (2007) 11, 1652–1670.

[6] V. GEFFERT, State hierarchy for one-way finite automata. *Journal of Automata, Languages and Combinatorics (JALC)* 12 (2007) 1-2, 139–145.

[7] M. A. HARRISON, *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[8] M. HOLZER, M. KUTRIB (eds.), *Implementation and Application of Automata - 19th International Conference, CIAA 2014, Giessen, Germany, July 30 - August 2, 2014. Proceedings*. Lecture Notes in Computer Science 8587, Springer, 2014.

[9] M. HRICKO, G. JIRÁSKOVÁ, A. SZABARI, Union and intersection of regular languages and descriptional complexity. In: C. MEREGHETTI, B. PALANO, G. PIGHIZZINI, D. WOTSCHKE (eds.), *7th International Workshop on Descriptional Complexity of Formal Systems – DCFS 2005, Como, Italy, June 30 - July 2, 2005. Proceedings*. Università degli Studi di Milano, 2005, 170–181.

[10] K. IWAMA, Y. KAMBAYASHI, K. TAKAKI, Tight bounds on the number of states of DFAs that are equivalent to n-state NFAs. *Theoretical Computer Science* 237 (2000) 1–2, 485–494.

[11] K. IWAMA, A. MATSUURA, M. PATERSON, A family of NFA's which need $2^n - \alpha$ deterministic states. In: M. NIELSEN, B. ROVAN (eds.), *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 - September 1, 2000, Proceedings*. Lecture Notes in Computer Science 1893, Springer, 2000, 436–445.

[12] J. JIRÁSEK, G. JIRÁSKOVÁ, A. SZABARI, Deterministic blow-ups of minimal nondeterministic finite automata over a fixed alphabet. *International Journal of Foundations of Computer Science* 19 (2008) 3, 617–631.

[13] G. JIRÁSKOVÁ, Deterministic blow-ups of minimal NFA's. *RAIRO – Theoretical Informatics and Applications (RAIRO: ITA)* 40 (2006) 3, 485–499.

[14] G. JIRÁSKOVÁ, Concatenation of regular languages and descriptional complexity. *Theory of Computing Systems* 49 (2011) 2, 306–318.

[15] G. JIRÁSKOVÁ, Magic numbers and ternary alphabet. *International Journal of Foundations of Computer Science* 22 (2011) 2, 331–344.

[16] G. JIRÁSKOVÁ, M. PALMOVSKÝ, J. ŠEBEJ, Kleene closure on regular and prefix-free languages. In: HOLZER and KUTRIB [8], 2014, 226–237.

[17] G. JIRÁSKOVÁ, A. SZABARI, J. ŠEBEJ, The complexity of languages resulting from the concatenation operation. In: C. CÂMPEANU, F. MANEA, J. SHALLIT (eds.), *Descriptional Complexity of Formal Systems - 18th IFIP WG 1.2 International Conference, DCFS 2016, Bucharest, Romania, July 5-8, 2016. Proceedings*. Lecture Notes in Computer Science 9777, Springer, 2016, 153–167.

[18] H. JÜRGENSEN, R. REIS (eds.), *Descriptional Complexity of Formal Systems - 15th International Workshop, DCFS 2013, London, ON, Canada, July 22-25, 2013. Proceedings*. Lecture Notes in Computer Science 8031, Springer, 2013.

[19] O. B. LUPANOV, A comparison of two types of finite automata. *Problemy Kibernetiki* 9 (1963), 321–326. (in Russian) German translation: Über den Vergleich zweier Typen endlicher Quellen. Probleme der Kybernetik 6, 328–335 (1966).

[20] A. R. MEYER, M. J. FISCHER, Economy of description by automata, grammars, and formal systems. In: *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*. IEEE Computer Society, 1971, 188–191.

[21] F. R. MOORE, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transaction on Computing* C–20 (1971), 1211–1219.

[22] C. NICAUD, Average state complexity of operations on unary automata. In: M. KUTYŁOWSKI, L. PACHOLSKI, T. WIERZBICKI (eds.), *Mathematical Foundations of Computer Science 1999, 24th International Symposium, MFCS'99, Szklarska Poręba, Poland, September 6-10, 1999, Proceedings*. Lecture Notes in Computer Science 1672, Springer, 1999, 231–240.

[23] M. O. RABIN, D. SCOTT, Finite automata and their decision problems. *IBM Journal of Research and Development* 3 (1959), 114–125.

[24] J. ŠEBEJ, Reversal on regular languages and descriptional complexity. In: JÜRGENSEN and REIS [18], 2013, 265–276.

[25] L. VAN ZIJL, Magic numbers for symmetric difference NFAs. *International Journal of Foundations of Computer Science* 16 (2005) 5, 1027–1038.

[26] YU. L. YERSHOV, On a conjecture of V. A. Uspenskii. *Algebra i Logika* 1 (1962), 45–48. (in Russian).

[27] S. YU, Q. ZHUANG, K. SALOMAA, The state complexities of some basic operations on regular languages. *Theoretical Computer Science* 125 (1994) 2, 315–328.

NONDETERMINISTIC COMPLEXITY OF POWER AND POSITIVE CLOSURE ON SUBCLASSES OF CONVEX LANGUAGES

Michal Hospodár   Matúš Palmovský

Mathematical Institute, Slovak Academy of Sciences,
Grešákova 6, 040 01 Košice, Slovakia
`hosmich@gmail.com`   `matp93@gmail.com`

Abstract
We study the nondeterministic state complexity of the k-th power and positive closure operations on the classes of prefix-, suffix-, factor-, and subword-free, -closed, and -convex regular languages, and on the classes of right, left, two-sided, and all-sided ideal languages. We show that the upper bound kn on the complexity of the k-th power in the class of regular languages is tight for closed and convex classes, while in the remaining classes, the tight upper bound is $k(n-1)+1$. Next we show that the upper bound n on the complexity of the positive closure operation in the class of regular languages is tight in all considered classes except for classes of factor-closed and subword-closed languages, where the complexity is one. All our worst-case examples are described over a unary or binary alphabet, except for witnesses for the k-th power on subword-closed and subword-convex languages which are described over a ternary alphabet. Moreover, whenever a binary alphabet is used for describing a worst-case example, it is optimal in the sense that the corresponding upper bounds cannot be met by a language over a unary alphabet. The most interesting result is the description of a binary factor-closed language meeting the upper bound kn for the k-th power. To get this result, we use a method which enables us to avoid tedious descriptions of fooling sets.

1. Introduction

The nondeterministic state complexity of a regular language is the smallest number of states in any nondeterministic finite automaton (with a unique initial state) recognizing this language. The nondeterministic state complexity of a regular operation is the number of states that are sufficient and necessary in the worst case to accept the language resulting from this operation, considered as a function of the nondeterministic state complexities of the operands.

The nondeterministic state complexity of basic regular operations such as union, intersection, concatenation, and positive closure, has been investigated by Holzer and Kutrib [8].

Research supported by VEGA grant 2/0084/15 and grant APVV-15-0091.

The binary witnesses for complementation and reversal were described by Jirásková [12]. The k-th power operation on nondeterministic automata was studied by Domaratzki and Okhotin [5]. The nondeterministic state complexity of operations on prefix-free and suffix-free languages was examined by Han et al. [6, 7] and by Jirásková et al. [13, 15]. The results of these papers were improved and new results on nondeterministic complexity were obtained in a series of papers by Mlynárčik et al. In [14], complementation on prefix-free, suffix-free, and non-returning languages was investigated. Complementation on factor-free, subword-free, and ideal languages was considered in [17], basic operations (intersection, union, concatenation, Kleene star, reversal, complementation) on closed and ideal languages in [10], and basic operations on free and convex languages in [11]. Let us mention that the (deterministic) state complexity of basic operations on all above mentioned classes were considered by Brzozowski et al. [2, 3, 4].

In this paper, we investigate the nondeterministic state complexity of the k-th power and positive closure operations on subclasses of convex languages. For both operations and all considered subclasses, we provide a tight upper bound on its nondeterministic state complexity. Except for two cases in which our witnesses are ternary, all the witnesses are described over a binary or unary alphabet. Moreover, whenever a binary alphabet is used, it is always optimal in the sense that the corresponding upper bound cannot be met by any unary language.

2. Preliminaries

We assume that the reader is familiar with basic notions in formal languages and automata theory. For details and all the unexplained notions, the reader may refer to [9, 19, 20]. Let Σ be a finite non-empty alphabet of symbols. Then Σ^* denotes the set of strings over the alphabet Σ including the empty string ε. A language is any subset of Σ^*. The *concatenation* of two languages K and L is the language $KL = \{uv \mid u \in K \text{ and } v \in L\}$. The *k-th power* of a language L is the language $L^k = LL^{k-1}$ where $L^0 = \{\varepsilon\}$. The *Kleene star* of a language L is the language $L^* = \bigcup_{i\geq 0} L^i$. The *positive closure* of a language L is the language $L^+ = \bigcup_{i\geq 1} L^i$.

A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \cdot, s, F)$ where Q is a finite non-empty set of *states*, Σ is a finite non-empty *input alphabet*, $s \in Q$ is the *initial* state, $F \subseteq Q$ is the set of *final* (or *accepting*) states, and $\cdot \colon Q \times \Sigma \to 2^Q$ is the *transition function* which can be extended to the domain $2^Q \times \Sigma^*$ in the natural way.

The *language accepted* (or *recognized*) by the NFA A is defined as $L(A) = \{w \in \Sigma^* \mid s \cdot w \cap F \neq \emptyset\}$. An NFA is a (partial) *deterministic finite automaton* (DFA) if $|q \cdot a| \leq 1$ for each q in Q and each a in Σ.

We say that (p, a, q) is a transition in NFA A if $q \in p \cdot a$. We also say that the state q has an *in-transition* on symbol a, and the state p has an *out-transition* on symbol a. An NFA is *non-returning* if its initial state does not have any in-transitions, and it is *non-exiting* if each its final state does not have any out-transitions. To *omit* a state in an NFA means to remove it from the set of states and to remove all its in-transitions and out-transitions from the transition function. To *merge* two states means to replace them by a single state with all in-transitions and out-transitions of the original states.

The *reverse of a string* w in Σ^* is defined by $\varepsilon^R = \varepsilon$ and $(wa)^R = aw^R$ for each a in Σ and each w in Σ^*. The *reverse of a language* L is the language $L^R = \{w^R \mid w \in L\}$. The *reverse of an* NFA $A = (Q, \Sigma, \cdot, s, F)$ is the NFA $A^R = (Q, \Sigma, \cdot^R, F, \{s\})$ with possibly multiple initial states where $q \cdot^R a = \{p \in Q \mid q \in p \cdot a\}$; notice that A^R is obtained from A by reversing all the transitions, and by swapping the roles of the initial and final states. Let $A = (Q, \Sigma, \cdot, s, F)$ be an NFA and $X, Y \subseteq Q$. We say that X is *reachable* in A if there is a string w in Σ^* such that $X = s \cdot w$. Next, we say that Y is *co-reachable* in A if Y is reachable in A^R.

The nondeterministic state complexity of a regular language L, denoted nsc(L), is the smallest number of states in any NFA for L. To provide lower bounds on nondeterministic state complexity, we use the fooling set method described below.

Definition 2.1 *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 each i, j in $\{1, 2, \ldots, n\}$, $x_i y_i \in L$, and if $i \neq j$, then $x_i y_j \notin L$ or $x_j y_i \notin L$.*

Lemma 2.2 (cf. [1, Lemma 1]) *Let $\mathcal{F}$ be a fooling set for a regular language L. Then every* NFA *for L has at least $|\mathcal{F}|$ states.*

The next lemma provides a useful way to prove the minimality of a given NFA.

Lemma 2.3 *Let $n \geq 2$. Let A be an* NFA *with the state set $Q = \{1, 2, \ldots, n\}$ and let $\{(X_i, Y_i) \mid i \in Q\}$ be a set of pairs of subsets of Q such that for each i in Q*

(1) X_i is reachable and Y_i is co-reachable in A,

(2) $i \in X_i \cap Y_i$, and

(3) $X_i \subseteq \{i, i+1, \ldots, n\}$ and $Y_i \subseteq \{1, 2, \ldots, i\}$.

Then every NFA *for $L(A)$ has at least n states.*

Proof. Since X_i is reachable, there is a string x_i which sends the initial state of A to the set X_i. Since Y_i is co-reachable, there is a string y_i which is accepted by A from every state in Y_i and rejected from every other state. Since $X_i \cap Y_i = \{i\}$, the string $x_i y_i$ is in $L(A)$. Let $i \neq j$. Without loss of generality, we have $i > j$. Then $X_i \cap Y_j = \emptyset$, so $x_i y_j$ is not in $L(A)$. It follows by Definition 2.1 that the set $\{(x_i, y_i) \mid i \in Q\}$ is a fooling set for $L(A)$, so every NFA for $L(A)$ has at least n states by Lemma 2.2. □

If $u, v, w, x \in \Sigma^*$ and $w = uxv$, then u is a *prefix* of w, x is a *factor* of w, and v is a *suffix* of w. If $w = u_0 v_1 u_1 \cdots v_n u_n$, where $u_i, v_i \in \Sigma^*$, then $v_1 v_2 \cdots v_n$ is a *subword* of w. A prefix v (suffix, factor, subword) of w is *proper* if $v \neq w$. A language L is *prefix-free* if $w \in L$ implies that no proper prefix of w is in L; it is *prefix-closed* if $w \in L$ implies that each prefix of w is in L; and it is *prefix-convex* if $u, w \in L$ and u is a prefix of w imply that each string v such that u is a prefix of v and v is a prefix of w is in L. Suffix-, factor-, and subword-free, -closed, and -convex languages are defined analogously. A language is a right (respectively, left, two-sided, all sided) *ideal* if $L = L\Sigma^*$ (respectively, $L = \Sigma^* L, L = \Sigma^* L \Sigma^*, L = L \sqcup\!\sqcup \Sigma^*$ where $L \sqcup\!\sqcup \Sigma^*$ is the language obtained from L by inserting any number of symbols to any string in L). Notice that the classes of free, closed, and ideal languages are subclasses of convex languages.

It is known that if a language is prefix-free, then every minimal NFA for it is non-exiting [18, Proposition 4.2], and if a language is suffix-free, then every minimal NFA for it is non-returning [18, Proposition 4.3]. Next, if a language is a right (left) ideal, then it is accepted by a minimal NFA such that its unique final (initial) state has a loop on each symbol and no other out-transitions (in-transitions) [10, Proposition 12], [18, Proposition 6.1]. Finally, an NFA with all states final accepts a prefix-closed language [18, Proposition 5.1], an NFA with all states initial accepts a suffix-closed language, and if a language is prefix-closed and suffix-closed, then it is factor-closed [18, Proposition 5.3].

3. Results

In this section, we examine the nondeterministic state complexity of the k-th power and positive closure on subclasses of convex languages. To get upper bounds, we use automata characterizations of languages in considered classes. To get lower bounds, we use the fooling set method given by Lemma 2.2 or, in the case of binary factor-closed languages, its simplification given by Lemma 2.3.

The nondeterministic state complexity of the k-th power on regular languages is kn if $k \geq 2$ and $n \geq 2$ [5, Theorem 3]. The next theorem shows that the complexity of the k-th power on all the classes of free, ideal, and unary convex languages is $k(n-1)+1$, while in all the remaining classes, it is kn. To describe a subword-closed witness, we use a ternary alphabet. All the remaining witnesses are described over binary or unary alphabets, and moreover, the binary alphabet is always optimal.

Theorem 3.1 (k-th Power) *Let $k \geq 2$ and $n \geq 2$. Let L be a language with* $\operatorname{nsc}(L) \leq n$.

(1) If L is prefix- or suffix-free, then $\operatorname{nsc}(L^k) \leq k(n-1)+1$, and this upper bound is met by a unary subword-free language.

(2) If L is right or left ideal, then $\operatorname{nsc}(L^k) \leq k(n-1)+1$, and this upper bound is met by a unary all-sided ideal language.

(3) If L is a unary convex language, then $\operatorname{nsc}(L^k) \leq k(n-1)+1$, and this upper bound is met by a unary subword-closed language.

(4) If L is prefix- or suffix-closed, then $\operatorname{nsc}(L^k) \leq kn$, and this upper bound is met by a binary factor-closed language and by a ternary subword-closed language.

Proof. (1) We may assume that a minimal NFA N for a prefix-free language L is non-exiting and has a unique final state. To get an NFA for L^k, we take k copies of N and we merge the final state in the j-th copy with the initial state in the $(j+1)$-th copy if $1 \leq j \leq k-1$. The initial state of the resulting NFA is the initial state in the first copy, and its final state is the final state in the k-th copy. If L is suffix-free, then we may assume that a minimal NFA N for L is non-returning. To get an NFA for L^k, we take k copies of N. For every symbol a and every final state p in the j-th copy with $1 \leq j \leq k-1$, we make the state p non-final and add the transitions (p, a, q) whenever there is a transition on a to q from the initial state in the $(j+1)$-th copy. Next, we omit the unreachable initial state in the $(j+1)$-th copy.

For tightness, let $L = \{a^{n-1}\}$. Then L is subword-free and it is accepted by an n-state NFA. We have $L^k = \{a^{k(n-1)}\}$ and the set $\{(a^i, a^{k(n-1)-i}) \mid 0 \le i \le k(n-1)\}$ is a fooling set for L^k of size $k(n-1)+1$. By Lemma 2.2, every NFA for L^k has at least $k(n-1)+1$ states. Hence $\mathrm{nsc}(L^k) = k(n-1)+1$.

(2) We may assume that a minimal NFA for a right ideal language L has a loop on every input symbol in its unique final state which has no other out-transitions. The construction of an NFA for L^k is the same as for prefix-free languages in case (1). Next, we may assume that a minimal NFA for a left ideal language L has a loop on every input symbol in its initial state which has no other in-transitions. The construction of an NFA for L^k is the same as for suffix-free languages in case (1), except that we add a loop on every symbol on p.

For tightness, let $L = \{a^i \mid i \ge n-1\}$. Then L is an all-sided ideal language and L is accepted by an n-state NFA. We have $L^k = \{a^i \mid i \ge k(n-1)\}$ and the same fooling set as in case (1) is a fooling set for L^k.

(3) Let L be a unary convex language accepted by a minimal n-state NFA. If L is infinite, then $L = \{a^i \mid i \ge n-1\}$, so $L^k = \{a^i \mid i \ge k(n-1)\}$ and $\mathrm{nsc}(L^k) = k(n-1)+1$. If L is finite, then the length of the longest string in L is $n-1$, so the length of the longest string in L^k is $k(n-1)$ and $\mathrm{nsc}(L^k) = k(n-1)+1$. This upper bound is met by the unary subword-closed language $\{a^i \mid 0 \le i \le n-1\}$.

(4) The upper bound is the same as in the general case of regular languages. Let us describe binary factor-closed and ternary subword-closed witnesses.

Let L be the language accepted by the NFA A shown in Figure 1. Since A has all states final

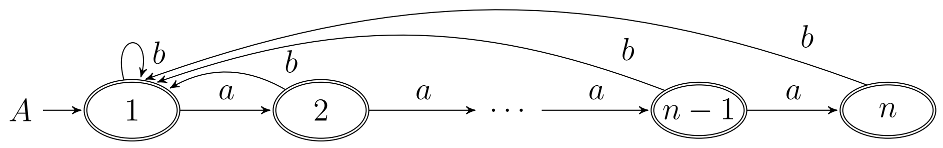

Figure 1: A binary factor-closed witness language meeting the upper bound kn for the k-th power.

and $L = L^R$, the language L is prefix-closed and suffix-closed, and therefore also factor-closed. The reader can verify that the language L^k is accepted by the kn-state partial DFA D consisting of k copies of A connected through the transitions on a going from the last state of the j-th copy to the second state of the $(j+1)$-th copy if $1 \le j \le k-1$.

For $i = 1, 2, \dots, kn$, let $X_i = \{i\}$ and $Y_i = \{1, 2, \dots, i\}$. Notice that

- each set X_i with $i \notin \{jn+1 \mid 1 \le j \le k-1\}$ is reachable in D by a word in a^*;
- each set X_i with $i \in \{jn+1 \mid 1 \le j \le k-1\}$ is reachable in D by a word in a^*b;
- each set Y_i with $i \notin \{jn \mid 1 \le j \le k-1\}$ is co-reachable in D since it is reachable in D^R by a word in a^*;
- each set Y_i with $i \in \{jn \mid 1 \le j \le k-1\}$ is co-reachable in D since it is reachable in D^R by a word in a^*b.

Moreover, $i \in X_i \cap Y_i$, $X_i \subseteq \{i, i+1, \ldots, kn\}$, and $Y_i \subseteq \{1, 2, \ldots, i\}$, so the sets X_i and Y_i satisfy the conditions of Lemma 2.3. Hence every NFA for L^k has at least kn states, which together with the upper bound gives $\operatorname{nsc}(L^k) = kn$.

Next, let $L = \{b^*a^ic^* \mid 0 \le i \le n-1\}$, which is accepted by an n-state NFA. Since every subword of a string $b^\ell a^i c^m$ in L is of the form $b^{\ell'} a^{i'} c^{m'}$ where $i' \le i \le n-1$, the language L is subword-closed. For each j with $1 \le j \le k$, consider the set of pairs of strings

$$\mathcal{F}_j = \{((ba^{n-1}c)^{j-1}ba^i, a^{n-1-i}c(ba^{n-1}c)^{k-j}) \mid 0 \le i \le n-1\}.$$

We have $(ba^{n-1}c)^k \in L^k$. Next, we have $L^k \subseteq (b^*a^*c^*)^k$, and moreover, no string with more than $k(n-1)$ occurrences of a is in L^k. It follows that the set $\bigcup_{j=1}^k \mathcal{F}_j$ is a fooling set for L^k of size kn, so every NFA for L^k has at least kn states by Lemma 2.2. □

Notice that in the theorem above, we must have $n \ge 2$ since for every positive integer k, the k-th power of a language accepted by a 1-state NFA is the same language. The theorem also shows that two symbols are necessary to meet the bound kn for the k-th power on closed languages.

Now we consider the operation of positive closure. The upper bound on nondeterministic state complexity of positive closure on regular languages is n [8, Theorem 9] since we can get an NFA for L^+ from an NFA for L by adding the transition (q, a, s) whenever there is a transition (q, a, f) for a final state f. The next theorem showsthat this upper bound is tight in all the classes of free and ideal, so also convex languages, and on the classes of prefix-closed and suffix-closed languages. It also proves that the positive closure of every factor-closed language is of complexity one.

Theorem 3.2 (Positive Closure) *Let n be a positive integer.*

(1) There exists a unary subword-free language L with $\operatorname{nsc}(L) \le n$ and $\operatorname{nsc}(L^+) = n$.
(2) There exists a unary all-sided ideal language L with $\operatorname{nsc}(L) \le n$ and $\operatorname{nsc}(L^+) = n$.
(3) There exists a binary prefix-closed language L with $\operatorname{nsc}(L) \le n$ and $\operatorname{nsc}(L^+) = n$.
(4) There exists a binary suffix-closed language L with $\operatorname{nsc}(L) \le n$ and $\operatorname{nsc}(L^+) = n$.
(5) If L is factor-closed, then $\operatorname{nsc}(L^+) = 1$.

Proof. (1) Let $L = \{a^{n-1}\}$, which is accepted by an n-state NFA. Then $L^+ = \{a^{k(n-1)} \mid k \ge 1\}$ and the set $\{(a^i, a^{n-1-i}) \mid 0 \le i \le n-1\}$ is a fooling set for L^+ of size n. By Lemma 2.2, every NFA for L^+ has at least n states. Hence $\operatorname{nsc}(L^+) = n$.

(2) Let $L = \{a^i \mid i \ge n-1\}$, which is accepted by an n-state NFA. We have $L^+ = L$ and the same set as above is a fooling set for L^+ of size n.

(3) Let L be the language accepted by the NFA shown in Figure 2. Notice that each state of this NFA is final, hence L is prefix-closed. Consider the set $\mathcal{F} = \{(a^i, a^{n-1-i}b) \mid 0 \le i \le n-1\}$. We have $a^ia^{n-1-i}b = a^{n-1}b$, which is in L^+. Let $0 \le i < j \le n-1$. Then $a^ia^{n-1-j}b$ is not in L^+. Hence the set $\mathcal{F}$ is a fooling set for L^+ of size n.

(4) Let L be the language accepted by the NFA shown in Figure 3. Notice that if we make all states of this NFA initial, then we get an equivalent finite automaton. Hence L is suffix-closed.

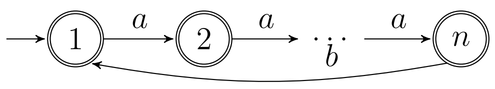

Figure 2: A binary prefix-closed witness language meeting the upper bound n for positive closure.

Moreover, $L^+ = L$ since the initial state is the unique final state. Consider the set of pairs of strings $\mathcal{F} = \{(ba^i, a^{n-1-i}) \mid 0 \le i \le n-1\}$. Since $ba^{n-1} \in L$, while ba^k with $k \le n-2$ is not in L, the set $\mathcal{F}$ is a fooling set for L, so also for L^+, of size n.

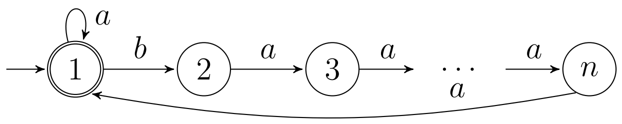

Figure 3: A binary suffix-closed witness language meeting the upper bound n for positive closure.

(5) Let Γ be the set of symbols present in at least one string of L. Then $L \subseteq \Gamma^*$, and since L is factor-closed, $\Gamma \cup \{\varepsilon\} \subseteq L$. It follows that $L^+ = \Gamma^*$, which is accepted by a one-state NFA. □

Notice that two symbols are necessary to meet the bound n for positive closure on prefix- and suffix-closed languages since every unary prefix- or suffix-closed language is also factor-closed.

4. Conclusions

We investigated the nondeterministic state complexity of the k-th power and positive closure in the subclasses of convex languages. We considered the classes of prefix-, suffix-, factor-, and subword-free, -closed, and -convex languages, and the classes of right, left, two-sided, and all-sided ideals. We found the exact complexities of both operations in each of the above mentioned classes. For describing witness languages for the k-th power on subword-closed and subword-convex languages, we used a ternary alphabet. All the remaining witness languages are described over a binary or unary alphabet. Moreover, if a binary alphabet is used, it is optimal in the sense that the corresponding upper bound cannot be met by any unary language.

References

[1] J. BIRGET, Intersection and union of regular languages and state complexity. *Information Processing Letters* 43 (1992) 4, 185–190.

[2] J. BRZOZOWSKI, G. JIRÁSKOVÁ, B. LI, J. SMITH, Quotient complexity of bifix, factor, and subword-free regular languages. *Acta Cybernetica* 21 (2014) 4, 507–527.

[3] J. A. BRZOZOWSKI, G. JIRÁSKOVÁ, B. LI, Quotient complexity of ideal languages. *Theoretical Computer Science* 470 (2013), 36–52.

[4] J. A. BRZOZOWSKI, G. JIRÁSKOVÁ, C. ZOU, Quotient complexity of closed languages. *Theory of Computing Systems* 54 (2014) 2, 277–292.

[5] M. DOMARATZKI, A. OKHOTIN, State complexity of power. *Theoretical Computer Science* 410 (2009) 24-25, 2377–2392.

[6] Y. HAN, K. SALOMAA, Nondeterministic state complexity for suffix-free regular languages. In: MCQUILLAN and PIGHIZZINI [16], 2010, 189–196.

[7] Y. HAN, K. SALOMAA, D. WOOD, Nondeterministic state complexity of basic operations for prefix-free regular languages. *Fundamenta Informaticae* 90 (2009) 1–2, 93–106.

[8] M. HOLZER, M. KUTRIB, Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science* 14 (2003) 6, 1087–1102.

[9] J. E. HOPCROFT, J. D. ULLMAN, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[10] M. HOSPODÁR, G. JIRÁSKOVÁ, P. MLYNÁRČIK, Nondeterministic complexity of operations on closed and ideal languages. In: Y. HAN, K. SALOMAA (eds.), *Implementation and Application of Automata - 21st International Conference, CIAA 2016, Seoul, South Korea, July 19-22, 2016, Proceedings*. Lecture Notes in Computer Science 9705, Springer, 2016, 125–137.

[11] M. HOSPODÁR, G. JIRÁSKOVÁ, P. MLYNÁRČIK, Nondeterministic complexity of operations on free and convex languages. In: A. CARAYOL, C. NICAUD (eds.), *Implementation and Application of Automata - 22nd International Conference, CIAA 2017, Marne-la-Vallée, France, June 27-30, 2017, Proceedings*. Lecture Notes in Computer Science 10329, Springer, 2017, 138–150.

[12] G. JIRÁSKOVÁ, State complexity of some operations on binary regular languages. *Theoretical Computer Science* 330 (2005) 2, 287–298.

[13] G. JIRÁSKOVÁ, M. KRAUSOVÁ, Complexity in prefix-free regular languages. In: MCQUILLAN and PIGHIZZINI [16], 2010, 197–204.

[14] G. JIRÁSKOVÁ, P. MLYNÁRČIK, Complement on prefix-free, suffix-free, and non-returning NFA languages. In: H. JÜRGENSEN, J. KARHUMÄKI, A. OKHOTIN (eds.), *Descriptional Complexity of Formal Systems - 16th International Workshop, DCFS 2014, Turku, Finland, August 5-8, 2014. Proceedings*. Lecture Notes in Computer Science 8614, Springer, 2014, 222–233.

[15] G. JIRÁSKOVÁ, P. OLEJÁR, State complexity of intersection and union of suffix-free languages and descriptional complexity. In: H. BORDIHN, R. FREUND, M. HOLZER, M. KUTRIB, F. OTTO (eds.), *Workshop on Non-Classical Models for Automata and Applications (NCMA 2009)*. books@ocg.at 256, Österreichische Computer Gesellschaft, 2009, 151–166.

[16] I. MCQUILLAN, G. PIGHIZZINI (eds.), *Proceedings Twelfth Annual Workshop on Descriptional Complexity of Formal Systems, DCFS 2010, Saskatoon, Canada, 8-10th August 2010*. EPTCS 31, 2010.

[17] P. MLYNÁRČIK, Complement on free and ideal languages. In: J. SHALLIT, A. OKHOTIN (eds.), *Descriptional Complexity of Formal Systems – 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25–27, 2015. Proceedings*. Lecture Notes in Computer Science 9118, Springer, 2015, 185–196.

[18] P. MLYNÁRČIK, Nondeterministic state complexity in subregular classes. *Dissertation thesis.* Faculty of Mathematics, Physics and Informatics of the Comenius University, Bratislava (2017). `http://im.saske.sk/%7Ejiraskov/students/phd_thesis_mlynarcik.pdf`

[19] M. SIPSER, *Introduction to the Theory of Computation*. Cengage Learning, Boston, 2012.

[20] S. YU, Chapter 2: Regular Languages. In: G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. Vol. 1, Springer, Heidelberg, 1997, 41–110.

EXTENDED FINITE AUTOMATA AND DECISION PROBLEMS FOR MATRIX SEMIGROUPS

Özlem Salehi   Ahmet Celal Cem Say

Boğaziçi University, Department of Computer Engineering,
Bebek 34342 İstanbul, Turkey
{ozlem.salehi,say}@boun.edu.tr

Abstract
We make a connection between the subgroup membership and identity problems for matrix groups and extended finite automata. We provide an alternative proof for the decidability of the subgroup membership problem for 2×2 integer matrices. We show that the emptiness problem for extended finite automata over 4×4 integer matrix semigroups is undecidable. We prove that the decidability of the universe problem for extended finite automata is a sufficient condition for the decidability of the subgroup membership and identity problems.

1. Introduction

Among the various extensions of classical finite state automata, extended finite automata over a monoid M or M-automata have been investigated both implicitly and explicitly by many researchers [2, 5, 9]. An M-automaton is a nondeterministic finite automaton equipped with a register that is multiplied by an element of the monoid M at each step. The register is initialized with the identity element of the monoid and a successful computation is the one which ends in an accept state with the register being equal to the identity element. In this paper, our aim is to make a connection between the theory of extended finite automata and the subgroup membership and identity problems for matrix semigroups. Matrices play an important role in various areas of computation, which makes it interesting to study decision problems on matrices. Even for integer matrices of low dimension, many decision problems become non-trivial for finitely generated infinite semigroups.

Let S be a matrix semigroup finitely generated by a generating set of square matrices F. The *membership problem* is to decide whether or not a given matrix Y belongs to the matrix semigroup S [7]. Equivalently, given a finite set of matrices $F = \{Y_1, Y_2, \ldots, Y_n\}$ and a matrix Y, the problem is to determine if there exists an integer $k \geq 1$ and $i_1, i_2, \ldots, i_k \in \{1, \ldots, n\}$ such that $Y_{i_1} Y_{i_2} \cdots Y_{i_k} = Y$. The identity problem is a special case of the membership problem

Özlem Salehi is partially supported by TÜBİTAK (Scientific and Technological Research Council of Turkey).

where Y is restricted to be the identity matrix. Introduced by Mihailova [8], the subgroup membership problem is one of the classical decision problems in group theory. Given elements $h_1, h_2, \dots, h_n$ and g of a group G, the subgroup membership problem for H in G asks whether g belongs to the subgroup H generated by $h_1, h_2, \dots, h_n$. Note that the subgroup membership problem for matrix groups is a special case of the membership problem. For 2×2 integer matrices, the decidability of the membership problem is proved in [10]. For 3×3 matrices, both the identity and membership problems are still open. Undecidability of the membership problem for 4×4 integer matrices is known for a long time due to a result by Mihailova [8] whereas the undecidability of the identity problem has been proved recently in [1, 6].

For our purposes, we define S-automata or extended finite automata over semigroups, generalizing the notion of M-automata from monoids to semigroups. The emptiness problem is defined as the problem of deciding whether a given machine accepts any string. For 2×2 integer matrices, by using the decidability of the emptiness problem of the corresponding extended finite automata, we provide an alternative proof for the decidability of subgroup membership problem. We show that the undecidability of the identity problem for 4×4 integer matrices yields the undecidability of the emptiness problem for extended finite automata over semigroups of 4×4 integer matrices. We also prove some results on the the decidability of the universe problem for extended finite automata, the problem of deciding whether a given machine accepts every string.

2. Background

2.1. Preliminaries

We denote by $\mathbb{Z}^{n \times n}$ the set of $n \times n$ matrices with integer entries. $GL(n, \mathbb{Z})$ denotes the general linear group of degree n over the ring of integers, equivalently the group of $n \times n$ invertible matrices with integer entries. Note that these matrices have determinant ± 1. Restricting the matrices in $GL(n, \mathbb{Z})$ to those that have determinant 1, we obtain the special linear group of degree n over the ring of integers, $SL(n, \mathbb{Z})$. We denote the *free group* over r generators by $\mathbf{F}_r$.

Word problem for G is the subgroup membership problem for the trivial group generated by 1. In other words, given an element $g \in G$, the problem is to decide whether g represents the identity element. The word problem language of G is the language $W(G, X)$ over $A = X \cup X^{-1}$ and consists of all words that represent the identity element of G. Most of the time, the statements about word problem are independent of the generating set and in these cases the word problem language is denoted by $W(G)$.

2.2. S-Automaton

Let S be a semigroup. Let Q be the set of states, where $q_0 \in Q$ denotes the initial state, $Q_a \subseteq Q$ denotes the set of accepting states, and let Σ be the input alphabet where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

An S-*automaton* (extended finite automaton over S) is a 6-tuple $V = (Q, \Sigma, S, \delta, q_0, Q_a)$ where the transition function δ is defined as $\delta : Q \times \Sigma_\varepsilon \to \mathbb{P}(Q \times S)$. $\delta(q, \sigma) \ni (q', m)$ means that when V reads the symbol (or empty string) $\sigma \in \Sigma_\varepsilon$ in state q, it will move to state q', and write xm in the register, where x is the old content of the register.

An S-automaton is in fact an extended finite automaton or a group/monoid automaton [3, 5] where the group/monoid condition is loosened to a semigroup. In order to define the initialization and acceptance steps, we need an identity element. If S is a monoid or a group, then an identity element already exists and belongs to S. Otherwise, we define 1 to be the identity element of S. The register of V is initialized with the identity element 1 and an input string is accepted if, after completely reading the string, V enters an accept state with the content of the register being equal to the identity element. Note that when S is not a monoid nor a group, then V can accept only the empty string. Nevertheless, we define the concept of S-automaton so that the machines in the proofs of Theorem 4.1 and 5.2 are constructed properly.

We denote by $L(V)$ the set of accepted strings by V. $\mathfrak{L}(S)$ denotes the class of languages recognized by S-automata.

Alternatively, monoid automata can be defined through rational subsets as in [4, 5] which we discuss next.

A *finite automaton* F over a monoid M is a finite directed graph whose edges are labeled by elements from M. F consists of a vertex labeled as the initial vertex and a set of vertices labeled as the terminal vertices such that an element of M is accepted by F if it is the product of the labels on a path from the initial vertex to a terminal vertex. A subset of M is called *rational* if its elements are accepted by some finite automaton over M.

When M is a free monoid (such as Σ^*), then the accepted elements are words over Σ and the set of accepted words is a language over Σ. Rational subsets of a free monoid are called rational (regular) languages. Note that when $M = \Sigma^*$, then the definition coincides with the definition of a finite state automaton.

An M-automaton V recognizing a language over alphabet Σ is a finite automaton F over the monoid $\Sigma^* \times M$ such that the accepted elements are $(w, 1)$ where $w \in \Sigma^*$. This is stated explicitly in the following proposition by Corson ([2], Proposition 2.2). The proof involves constructing an M-automaton from a finite automaton over $\Sigma^* \times M$ and vice versa.

Fact 2.1 [2] *Let L be a language over an alphabet Σ. Then $L \in \mathfrak{L}(M)$ if and only if there exists a rational subset $R \subseteq \Sigma^* \times M$ such that $L = \{w \in \Sigma | wR1\}$.*

3. Decidability of the Subgroup Membership Problem for $\mathbb{Z}^{2\times 2}$

It is proved that the membership problem for subsemigroups of $\mathbb{Z}^{2\times 2}$ is decidable in [10]. In this section, we provide an alternative automata theoretic proof for the decidability of the subgroup membership problem for $\mathbb{Z}^{2\times 2}$.

For a finite index subgroup H of some finitely generated group G, it is known that $\mathfrak{L}(H) = \mathfrak{L}(G)$ [2]. We will go over the proof details and use Fact 2.1 to show that given a G-automaton, one can construct an H-automaton recognizing the same language.

Lemma 3.1 *Let G be a finitely generated group and let H be a subgroup of finite index. Any G-automaton can be converted into an H-automaton recognizing the same language.*

Proof. Let X be the generator set for G and let $A = X \cup X^{-1}$. Let V be a G-automaton recognizing language L over alphabet Σ. Then there exists a rational subset $R \subseteq \Sigma^* \times G$ such that $L = \{w \in \Sigma^* | wR1\}$. One can define the elements of G in terms of A to obtain a rational subset $R_0 \subseteq \Sigma^* \times A^*$.

Since H has finite index in G, $W(G) \in \mathfrak{L}(H)$ ([2] Lemma 2.4). It follows that there exists a rational subset $S \subseteq A^* \times H$ such that $W(G) = \{w \in A^* | wS1\}$.

Then the composition $R_0 \circ S$ is a rational subset of $\Sigma^* \times H$ and it follows that $L = \{w \in \Sigma^* | w(R_0 \circ S)1\}$ ([2], Theorem 3.1). The detailed construction of the finite automaton recognizing the composition is given in ([4], Theorem 5.3). Hence a finite automaton F over $\Sigma^* \times H$ recognizing L exists, from which an H-automaton V' recognizing L can be constructed. □

The following construction of a pushdown automaton simulating an $\mathbf{F}_2$-automaton is left as an exercise in [5]. We present here some details of the construction.

Lemma 3.2 *Any $\mathbf{F}_2$-automaton can be converted into a pushdown automaton recognizing the same language.*

Proof. Let V be an $\mathbf{F}_2$-automaton recognizing language L over Σ with the state set Q and let $X = \{a, b\}$ be the generator set for $\mathbf{F}_2$. Let us construct a pushdown automaton V' recognizing the same language with the stack alphabet X. Let $(q', f) \in \delta(q, \sigma)$ be a transition of V where $q, q' \in Q$, $\sigma \in \Sigma_\varepsilon$ and $f \in \mathbf{F}_2$ such that $f = f_1 f_2 \dots f_n$ where $f_i \in A = X \cup X^{-1}$ for $i = 1 \dots n$. In V', we need an extra n states $q_1 \dots q_n \notin Q$ to mimic each given transition of V. If $f_i = a$ or $f_i = b$, then this corresponds to pushing a or b to the stack, respectively. Similarly, if $f_i = a^{-1}$ or $f_i = b^{-1}$, then V' pops a or b from the stack. Each single transition of V is accomplished by the pushdown automaton V' by going through the extra states and pushing and popping symbols. Initially, the register of V is initialized with the identity element of $\mathbf{F}_2$, which corresponds to the stack of V' being empty. The acceptance condition of V, which is ending in an accept state with the register being equal to the identity element is realized in V' by starting with an empty stack and accepting with an empty stack in an accept state. We conclude that V' recognizes language L. □

Theorem 3.3 *Let H be a finitely generated subgroup of G. If the emptiness problem for G-automata is decidable, then the subgroup membership problem for H in G is decidable.*

Proof. The subgroup membership problem for H in G is the problem of deciding whether a given element $g \in G$ belongs to H. We are going to construct a G-automaton V_1 and show that $g \in H$ iff $L(V_1)$ is nonempty. V_1 has two states: the initial state q_1 and the accept state

q_2. The transition function of V_1 is defined as $\delta(q_1, a) = (q_2, g)$ and $\delta(q_2, a) = (q_2, h_i)$ for each $i = 1 \ldots n$ where the set $\{h_1, \ldots, h_n\}$ generates H.

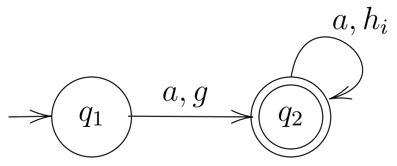

Figure 1: State transition diagram of V_1

If $g \in H$, then it is also true that $g^{-1} \in H$ since H is a group. There exists an integer $k \geq 1$ and $i_1, i_2, \ldots, i_k \in \{1, \ldots, n\}$ such that $h_{i_1} h_{i_2} \cdots h_{i_k} = g^{-1}$. The string a^k is accepted by V_1 as the register is initially multiplied by g and there exists a product of elements yielding g^{-1}, from which we can conclude that the identity element can be obtained through a series of transitions of the machine V_1. Hence, we can conclude that $L(V_1)$ is nonempty.

For the other direction, assume that $L(V_1)$ is nonempty, which means that some input string is accepted by V_1. Since the acceptance condition requires that the product of the elements multiplied by the register of V_1 is equal to the identity element and the register is initially multiplied by g, we can conclude that H contains g^{-1}. Since H is a group, $g \in H$ as well.

Now suppose that the emptiness problem for G-automaton is decidable. Then one can check if g is an element of H by constructing V_1 and checking if $L(V_1)$ is nonempty. Hence, the subgroup membership problem for H is also decidable. □

Theorem 3.4 *Given a matrix Y from $\mathbb{Z}^{2\times 2}$ and a subgroup H of $\mathbb{Z}^{2\times 2}$, it is decidable whether Y belongs to H.*

Proof. We are going to show that the emptiness problem for $\mathbb{Z}^{2\times 2}$-automaton is decidable and use Theorem 3.3 to conclude the result.

Suppose that a $\mathbb{Z}^{2\times 2}$-automaton V is given. When V processes an input string, its register is initialized by the identity matrix and multiplied by matrices from $\mathbb{Z}^{2\times 2}$. Suppose that in a successful computation leading to acceptance, the register is multiplied by some non-invertible matrix Y. Since Y is non-invertible, the register can not be equal to the identity matrix again and such a computation can not be successful. Any such edges labeled by a non-invertible matrix can be removed from V, without changing the accepted language. We can conclude that the matrices multiplied by the register are invertible and belong to $GL(2, \mathbb{Z})$ and V is in fact a $GL(2, \mathbb{Z})$-automaton.

Since $\mathbf{F}_2$ has finite index in $GL(2, \mathbb{Z})$, one can construct an $\mathbf{F}_2$-automaton recognizing $L(V)$ by Lemma 3.1. The $\mathbf{F}_2$-automaton can be converted to a pushdown automaton V' using the procedure described in Lemma 3.2. Since the emptiness problem for pushdown automata is known to be decidable, we conclude that the emptiness problem for $\mathbb{Z}^{2\times 2}$-automata is also decidable since a $\mathbb{Z}^{2\times 2}$-automaton can be converted to a pushdown automaton. Then by Theorem 3.3, the result follows. □

4. Undecidability of the Emptiness Problem for $\mathbb{Z}^{4\times 4}$-Automata

In [6], it is shown that the identity problem is undecidable for a semigroup generated by eight 4×4 integer matrices. Using this fact, we prove that the emptiness problem is undecidable for the corresponding semigroup automaton.

Theorem 4.1 *Let S be a finitely generated semigroup. If the emptiness problem for S-automata is decidable, then the identity problem for S is decidable.*

Proof. We are going to construct an S-automaton V_2 and show that S contains the identity element iff $L(V_2)$ is nonempty. V_2 has two states: the initial state q_1 and the accept state q_2. The transition function of V_2 is defined as $\delta(q_1, a) = (q_2, s_i)$ and $\delta(q_2, a) = (q_2, s_i)$ for each $i = 1 \dots n$ where $\{s_1, s_2, \dots s_n\}$ is the generator set for S.

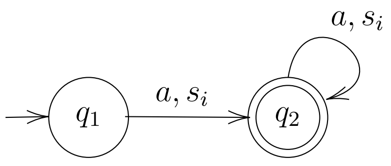

Figure 2: State transition diagram of V_2

If S contains the identity element, then there exists an integer $k \geq 1$ and $i_1, i_2, \dots, i_k \in \{1, \dots, n\}$ such that $s_{i_1} s_{i_2} \cdots s_{i_k} = 1$. Then the string a^k is accepted by V_2 as there exists a product of elements yielding the identity element and this product can be obtained by a series of transitions. Hence, we can conclude that $L(V_2)$ is nonempty. For the converse, suppose that $L(V_2)$ is nonempty, which means that some input string is accepted by V_2. Since the acceptance condition requires that the product of the elements multiplied by the register of V_2 is equal to the identity element, we can conclude that S contains the identity element.

Now suppose that the emptiness problem for S-automaton is decidable. Then one can check if S contains the identity element by constructing V_2 and checking if $L(V_2)$ is nonempty. Hence, the identity problem for S is also decidable. □ The identity problem for $\mathbb{Z}^{4\times 4}$ is shown to be undecidable for a semigroup of 48 matrices in [1]. Later on, the result is improved to eight matrices in [6].

Fact 4.2 [6] *Given a semigroup S generated by eight 4×4 integer matrices, determining whether the identity matrix belongs to S is undecidable.*

Using this result, we obtain the following corollary about the emptiness problem for extended finite automata over semigroups of $\mathbb{Z}^{4\times 4}$.

Corollary 4.3 *Let S be a subsemigroup of $\mathbb{Z}^{4\times 4}$ generated by eight matrices. The emptiness problem for S-automaton is undecidable.*

Proof. By Fact 4.2 we know that the identity problem for S is undecidable. By Theorem 4.1, the result follows. □

5. Universe Problem for S-Automata

In this section we prove some results connecting the universe problem for S-automata and the subgroup membership and identity problems for S.

Theorem 5.1 *Let H be a finitely generated subgroup of G. If the universe problem for G-automata is decidable, then the subgroup membership problem for H in G is decidable.*

Proof. We are going to construct a G-automaton V_2 and such that $g \in H$ iff $L(V_3) = \Sigma^*$ where $\Sigma = \{a\}$. $\{h_1, h_2, \dots h_n\}$ is the generator set for H.

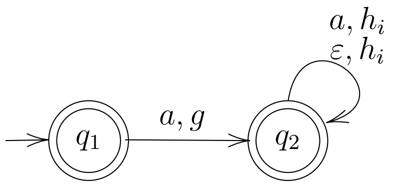

Figure 3: State transition diagram of V_3

The rest of the proof is similar to the proof of Theorem 3.3 and omitted here. □

Theorem 5.2 *Let S be a finitely generated semigroup. If the universe problem for S-automata is decidable, then the identity problem for S is decidable.*

Proof. We are going to construct an S-automaton V_4 such that S contains the identity element iff $L(V_4) = \Sigma^*$ where $\Sigma = \{a\}$. $\{s_1, s_2, \dots s_n\}$ is the generator set for S. The rest of the proof

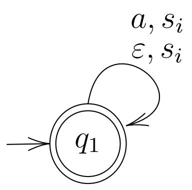

Figure 4: State transition diagram of V_4

is similar to the proof of Theorem 4.1 and omitted here. □

Let us note that the converses of Theorem 5.1 and 5.2 are not true. For a given pushdown automaton, an $\mathbf{F}_2$-automaton recognizing the same language can be constructed [5]. It is a well known fact that the universe problem for pushdown automata is undecidable from which we can conclude that the universe problem for $\mathbf{F}_2$-automaton is undecidable. On the other hand, $\mathbf{F}_2$ is a subgroup of $SL(2, \mathbb{Z})$ and the membership problem for $SL(2, \mathbb{Z})$ and thus the identity problem are known to be decidable [6].

6. Future Work

It is still not known whether the membership and identity problems are decidable for semigroups of 3×3 integer matrices. Recent results from [6] suggest that these problems are more likely to be decidable. We propose that investigating the decidability of the emptiness and universe problems for extended finite automata defined over 3×3 integer matrices is one possible way for obtaining results about the decision problems on these matrix semigroups.

References

[1] P. C. BELL, I. POTAPOV, On the undecidability of the identity correspondence problem and its applications for word and matrix semigroups. *International Journal of Foundations of Computer Science* 21 (2010) 06, 963–978.

[2] J. M. CORSON, Extended finite automata and word problems. *International Journal of Algebra and Computation* 15 (2005) 03, 455–466.

[3] J. DASSOW, V. MITRANA, Finite automata over free groups. *International Journal of Algebra and Computation* 10 (2000) 06, 725–737.

[4] R. H. GILMAN, Formal languages and infinite groups. *Geometric and computational perspectives on infinite groups* (1996), 27–51.

[5] M. KAMBITES, Word problems recognisable by deterministic blind monoid automata. *Theoretical Computer Science* 362 (2006) 1, 232–237.

[6] S.-K. KO, R. NISKANEN, I. POTAPOV, On the identity problem for the special linear group and the Heisenberg group. In: I. CHATZIGIANNAKIS, C. KAKLAMANIS, D. MARX, D. SANNELLA (eds.), *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), July 9–13, 2018, Prague, Czech Republic*. Leibniz International Proceedings in Informatics (LIPIcs) 107, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 132:1–15.

[7] A. MARKOV, On certain insoluble problems concerning matrices. In: *Doklady Akad. Nauk SSSR*. 57, 1947, 539–542.

[8] K. A. MIHAILOVA, The occurrence problem for free products of groups. *Mathematics of the USSR-Sbornik* 4 (1968) 2, 181.

[9] V. MITRANA, R. STIEBE, The accepting power of finite automata over groups. In: GH. PĂUN, A. SALOMAA (eds.), *New Trends in Formal Languages*. Lecture Notes in Computer Science 1218, Springer, 1997, 39–48.

[10] I. POTAPOV, P. SEMUKHIN, Decidability of the membership problem for 2×2 integer matrices. In: P. N. KLEIN (ed.), *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, January 16–19, 2017, Barcelona, Spain*. SIAM, 2017, 170–186.

Author Index