FACHGRUPPE
AFS

31. Theorietag "Automaten und Formale Sprachen"

Tagungsband

Arbeitstreffen der GI Fachgruppe "Automaten und Formale Sprachen"

Leipzig, 20.–21. September 2021

Vorwort

Dieser Tagungsband enthält die Zusammenfassungen der Beiträge zum 31. Theorietag der GI Fachgruppe "Automaten und Formale Sprachen". Der Theorietag findet aufgrund der geltenden Corona-Beschränkungen rein online vom 20.–21. September 2021 statt. Neben den Beiträgen enthält das Tagungsprogramm auch die eingeladenen Vorträge von Prof. KEVIN KNIGHT (Didi Labs, USA) und Prof. SEBASTIAN MANETH (Universität Bremen). Prof. KNIGHT trägt zum Thema "Are Automata Important for Machine Translation?" vor und wird Anwendungen der Automatentheorie diskutieren. Prof. MANETH trägt zu einem Kernthema der Automatentheorie vor und wird uns neueste Ergebnisse zu Baumübersetzern in seinem Vortrag "Definability Results for Top-Down Tree Transducers" vorstellen.

Selbst eine kleine Tagung wie der Theorietag benötigt die Hilfe und Unterstützung vieler Beteiligter. Der Organisator möchte hiermit explizit allen Autoren für ihr Interesse und ihre Beiträge danken. Diese Beiträge bilden das wissenschaftliche Kernprogramm des Treffens und geben den Teilnehmern einen Überblick über die Forschungsaktivitäten, die von der GI Fachgruppe abgedeckt werden. Die Zusammenarbeit mit dem Leitungsgremium der GI Fachgruppe war jederzeit produktiv und ermöglichte das Arbeitstreffen trotz der ungewöhnlichen Umstände. Ein Vor-Ort-Treffen mit dem gewohnten Kegeln ist sicherlich in Folgejahren wieder möglich. Abschließend gilt mein Dank allen Teilnehmern des Theorietags. Ich wünsche Ihnen eine interessante und erfolgreiche Tagung.

13. September 2021 Andreas Maletti
Leipzig

Inhaltsverzeichnis

New Derivation Modes for Catalytic P Systems with One Catalyst

Artiom Alhazov[X] Rudolf Freund[X] Sergiu Ivanov[X]
Sergey Verlan[X]

[X]Vladimir Andrunachievici Institute of Mathematics and Computer Science
Acdemiei 5, Chișinău, MD-2028, Moldova
artiom@math.md

[X] Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Wien, Austria
rudi@emcc.at

[X]Université Paris-Saclay, Université Évry,
IBISC, 91020, Évry-Courcouronnes, France
sergiu.ivanov@ibisc.univ-evry.fr

[X]Univ. Paris Est Creteil, LACL, 94010, Creteil, France
verlan@u-pec.fr

Abstract

We consider new variants of derivation modes for catalytic P systems with one catalyst which yield computational completeness: we only take those non-extendable multisets whose application yields the maximal number of generated objects or else those non-extendable multisets whose application yields the maximal difference in the number of objects between the newly generated configuration and the current configuration. Similar results can even be obtained if we do not require the multisets of rules to be non-extendable.

1. Introduction

One basic feature of P systems already presented in [6] is the maximally parallel derivation mode, i.e., using non-extendable multisets of rules in every derivation step. The result of a computation can be extracted when the system halts, i.e., when no rule is applicable any more. Catalysts are special symbols which allow only one object to evolve in its context and in their basic variant never evolve themselves, i.e., a catalytic rule is of the form $ca \to cv$, where c is a catalyst, a is a single object and v is a multiset of objects. In contrast, non-catalytic rules in catalytic P systems are non-cooperative rules of the form $a \to v$.

From the beginning, the question how many catalysts are needed for obtaining computational completeness has been one of the most intriguing challenges regarding (catalytic) P systems. Without catalysts only regular (semi-linear) sets can be generated when using the standard maximal derivation mode and the standard halting mode, i.e., a result is extracted when the system halts with no rule being applicable any more. In [4] it has already been shown that two catalysts are enough for generating any recursively enumerable set of multisets. Last year we could show computational completeness for P systems with only one catalyst which are using the derivation mode $max_{objects}$, i.e., only those multisets of rules which affect the maximal number of objects in the underlying configuration are taken, see [1]. In this extended abstract we exhibit our main results proved in [2] and show the computational completeness for P systems with only one catalyst if we take only multisets whose application yields the maximal number of generated objects or else those multisets whose application yields the maximal difference in the number of objects between the newly generated configuration and the current configuration.

2. Definitions

For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation, i.e., containing all possible strings over V. The *empty string* is denoted by λ. The set of all multisets over V is denoted by V°. The cardinality of a set or multiset M is denoted by $|M|$. For further notions and results in formal language theory we refer to textbooks like [3] and [8]. For an introduction to the area of membrane computing, see [7].

Register machines are well-known universal devices for computing on (or generating or accepting) sets of vectors of natural numbers.

Definition 2.1 *A* register machine *is a construct $M = (m, B, l_0, l_h, P)$ where m is the number of registers, P is the set of instructions bijectively labeled by elements of B, $l_0 \in B$ is the initial label, and $l_h \in B$ is the final label. The instructions of M can be of the following forms:*

- $p : (ADD(r), q, s)$, *with $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.*
  Increase the value of register r by one, and non-deterministically jump to instruction q or s.
- $p : (SUB(r), q, s)$, *with $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.*
  *If the value of register r is not zero then decrease the value of register r by one (*decrement *case) and jump to instruction q, otherwise jump to instruction s (*zero-test *case).*
- $l_h : HALT$. *Stop the execution of the register machine.*

A configuration *of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed. M is called* deterministic *if the ADD-instructions all are of the form $p : (ADD(r), q)$.*

Taking into account the well-known flattening process, e.g., see [5], in this paper we only consider simple catalytic P systems, i.e., with the simplest membrane structure of only one membrane, and with only one catalyst:

Definition 2.2 *A* simple catalytic P system *is a construct $\Pi = (V, \{c\}, T, w, \mathcal{R})$ where V is the alphabet of objects, $c \in V$ is the single catalyst, $T \subseteq (V \setminus \{c\})$, $w \in V^\circ$ is the multiset of objects initially present in the membrane region; $\mathcal{R}$ is a finite set of evolution rules over V; these evolution rules are of the forms $ca \to cv$ or $a \to v$, where c is a catalyst, a is an object from $V \setminus \{c\}$, and v is a multiset over $V \setminus \{c\}$.*

The multiset in the single membrane region of Π constitutes a *configuration* of the P system. The *initial configuration* is given by the initial multiset w. A transition between configurations is governed by the application of the evolution rules, which is done in a given derivation mode.

Given a P system $\Pi = (V, \{c\}, T, w, \mathcal{R})$, the standard parallel derivation mode used in P systems is the *maximally parallel derivation mode* (max for short); in the derivation modes $max_{rules}[max]$ and $max_{objects}[max]$ only a [non-extendable] multiset of rules using the maximal number of rules and objects, respectively, is allowed to be applied.

In addition to these well-known derivation modes, in this paper we also consider several new variants of derivation modes: in the derivation modes $max_{GENobjects}[max]$ as well as $max_{\Delta objects}[max]$ only [non-extendable] multisets R of rules are taken for which the number of objects generated by the application of the rules in R to the configuration C respectively the difference $\Delta C = |C'| - |C|$ between the number of objects in the configuration C' obtained by the application of R and the number of objects in the underlying configuration C is maximal.

3. Results

Theorem 3.1 *For any register machine with at least two decrementable registers we can construct a simple catalytic P system working in the derivation modes $max_{\Delta objects}[max]$ or in the derivation modes $max_{GENobjects}[max]$, which can simulate every step of the register machine in n steps where n is the number of decrementable registers.*

Proof. Given an arbitrary register machine $M = (m, B, l_0, l_h, P)$ we will construct a corresponding catalytic P system with one membrane and one catalyst $\Pi = (V, \{c\}, T, c(l_0, 1), \mathcal{R})$ simulating M. The main idea behind our construction is that all the objects except the catalyst c and the output objects (representing the contents of the output registers) go through a cycle of length n where n is the number of decrementable registers of the simulated register machine. When the objects are traversing the r-th section of the n sections, they "know" that they are to probably simulate a *SUB*-instruction on register r of the register machine M. The set of labels of *SUB*-instructions on register r is denoted by $B_{SUB(r)}$, the set of labels of all *ADD*-instructions by B_{ADD}. Without loss of generality, we assume that the first instruction labeled l_0 is an *ADD*-instruction on register 1.

$$\begin{aligned} V = & \{a_r \mid n+1 \leq r \leq m\} \cup \{(a_r, i) \mid 1 \leq r \leq n, 1 \leq i \leq n\} \\ & \cup \{(p,i) \mid p \in B_{ADD}, 1 \leq i \leq n\} \cup \{(p,i) \mid p \in B_{SUB(r)}, 1 \leq i \leq r+1\} \\ & \cup \{(p,i)^-, (p,i)^0 \mid p \in B_{SUB(r)}, r+2 \leq i \leq n\} \cup \{c, e, d\}. \end{aligned}$$

The construction includes the dummy object d which is erased by the rule $d \to \lambda$. The objects a_r, $n+1 \leq r \leq m$, represent the output registers. For the decrementable registers, we use the objects (a_r, i), $1 \leq r \leq n, 1 \leq i \leq n$, which go through a loop of n steps. The main idea now is that the only case when such an object can be used to decrement register r is when $i = r$, i.e., in the r-th step of the simulation cycle. In the same way as the register objects a_r, the program objects (p, i) representing the label p from B undergo the same cycle of length n.

$$(a_r, i) \to (a_r, i+1), 1 \leq r < n;\ (a_r, n) \to (a_r, 1).$$

For simulating an *ADD*-instruction $p : (ADD(r), q, s)$ we need the following rules:

$c(p,i) \to c(p,i+1)d,\ 1 \leq i < n,$ as well as
$c(p,n) \to c(q,1)(a_r,1),\ c(p,n) \to c(s,1)(a_r,1)$ for a decrementable register r and
$c(p,n) \to c(q,1)a_r,\ c(p,n) \to c(s,1)a_r$ for an output register r.

For simulating a *SUB*-instruction $p : (SUB(r), q, s)$ we need the following rules:

$c(p,i) \to c(p,i+1)d,\ 1 \leq i < r;\ \ (p,r) \to (p,r+1),\ c(a_r,r) \to ced.$
If $r < n-1$:
$ce \to cdddd,\ (p,r+1) \to (p,r+2)^-, c(p,r+1) \to c(p,r+2)^0 dd;$
$c(p,i)^- \to c(p,i+1)^- d,\ \ r+2 \leq i < n,\ \ c(p,n)^- \to c(q,1)d,$
$c(p,i)^0 \to c(p,i+1)^0 d,\ \ r+2 \leq i < n,\ \ c(p,n)^0 \to c(s,1)d.$
If $r = n-1$: $ce \to cdddd,\ (p,n) \to (q,1),\ c(p,n) \to c(s,1)dd.$
If $r = n$: $ce \to cdddd,\ (p,n+1) \to (q,2),\ c(p,n+1) \to c(s,2)dd.$

To implement the final $HALT$-instruction $l_h : HALT$, we introduce d instead of $(l_h,1)$ or $(l_h,2)$ as done for other labels. We finally observe that the proof construction given above is even deterministic if the underlying register machine to be simulated is deterministic.

To mimick the non-extendability of the multisets of rules to be applied in the derivation modes $max_{\Delta objects}$ and $max_{GEN objects}$, we simply add one more dummy object d on the right-hand side of every rule constructed above, except for the erasing rule $d \to \lambda$. $\square$

References

[1] A. ALHAZOV, R. FREUND, S. IVANOV, When Catalytic P Systems With One Catalyst Can Be Computationally Complete. *Journal of Membrane Computing* (2021).

[2] A. ALHAZOV, R. FREUND, S. IVANOV, S. VERLAN, Variants of Simple P Systems With One Catalyst Being Computationally Complete. In: GY. VASZIL, C. ZANDRON, G. ZHANG (eds.), *International Conference on Membrane Computing ICMC 2021, Proceedings*. 2021, 21–38.

[3] J. DASSOW, GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. Springer, 1989.

[4] R. FREUND, L. KARI, M. OSWALD, P. SOSÍK, Computationally Universal P Systems Without Priorities: Two Catalysts Are Sufficient. *Theoretical Computer Science* **330** (2005) 2, 251–266.

[5] R. FREUND, A. LEPORATI, G. MAURI, A. E. PORRECA, S. VERLAN, C. ZANDRON, Flattening in (Tissue) P Systems. In: A. ALHAZOV, S. COJOCARU, M. GHEORGHE, YU. ROGOZHIN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing*. Lecture Notes in Computer Science 8340, Springer, 2014, 173–188.

[6] GH. PĂUN, Computing With Membranes. *Journal of Computer and System Sciences* **61** (2000) 1, 108–143.

[7] GH. PĂUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.

[8] G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. Springer, 1997.

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 9–12.

New Derivation Modes for Purely Catalytic P Systems

Artiom Alhazov$^{(A)}$ Rudolf Freund$^{(B)}$ Sergiu Ivanov$^{(C)}$
Marion Oswald$^{(B)}$

$^{(A)}$Vladimir Andrunachievici Institute of Mathematics and Computer Science
Acdemiei 5, Chișinău, MD-2028, Moldova
artiom@math.md

$^{(B)}$ Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Wien, Austria
{rudi,marion}@emcc.at

$^{(C)}$Université Paris-Saclay, Université Évry,
IBISC, 91020, Évry-Courcouronnes, France
sergiu.ivanov@ibisc.univ-evry.fr

Abstract

We consider new variants of derivation modes for purely catalytic P systems which can simulate register machines with n decrementable registers with only n catalysts: we only take those non-extendable multisets whose application yields the maximal number of generated objects or else those non-extendable multisets whose application yields the maximal difference in the number of objects between the newly generated configuration and the current configuration. Similar results can even be obtained if we do not require the multisets of rules to be non-extendable.

1. Introduction

One basic feature of P systems already presented in [5] is the maximally parallel derivation mode, i.e., using non-extendable multisets of rules in every derivation step. The result of a computation can be extracted when the system halts, i.e., when no rule is applicable any more. Catalysts are special symbols which allow only one object to evolve in its context and in their basic variant never evolve themselves, i.e., a catalytic rule is of the form $ca \to cv$, where c is a catalyst, a is a single object and v is a multiset of objects. In purely catalytic P systems only such catalytic rules are used.

With only one catalyst only regular (semi-linear) sets can be generated. In [3] it has already been shown that three catalysts are enough for generating any recursively enumerable set of multisets. In this extended abstract we exhibit our main results proved in [1] and show that P systems with only n catalysts can simulate register machines with n decrementable registers if we take only multisets whose application yields the maximal number of generated objects or else those multisets whose application yields the maximal difference in the number of objects between the newly generated configuration and the current configuration.

2. Definitions

For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation, i.e., containing all possible strings over V. The *empty string* is denoted by λ. The set of all multisets over V is denoted by V°. The cardinality of a set or multiset M is denoted by $|M|$. For further notions and results in formal language theory we refer to textbooks like [2] and [7]. For an introduction to the area of membrane computing, see [6].

Register machines are well-known universal devices for computing on (or generating or accepting) sets of vectors of natural numbers.

Definition 2.1 *A* register machine *is a construct $M = (m, B, l_0, l_h, P)$ where m is the number of registers, P is the set of instructions bijectively labeled by elements of B, $l_0 \in B$ is the initial label, and $l_h \in B$ is the final label. The instructions of M can be of the following forms:*
– $p : (ADD(r), q, s)$*, with $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.*
Increase the value of register r by one, and non-deterministically jump to instruction q or s.
– $p : (SUB(r), q, s)$*, with $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.*
*If the value of register r is not zero then decrease the value of register r by one (*decrement *case) and jump to instruction q, otherwise jump to instruction s (*zero-test *case).*
– $l_h : HALT$*. Stop the execution of the register machine.*
A configuration *of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed. M is called* deterministic *if the ADD-instructions all are of the form $p : (ADD(r), q)$.*

Taking into account the well-known flattening process, which means that computations in a P system with an arbitrary membrane structure can be simulated in a P system with only one membrane, e.g., see [4], in this paper we only consider simple purely catalytic P systems, i.e., with the simplest membrane structure of only one membrane:

Definition 2.2 *A* simple purely catalytic P system *is a construct $Pi = (V, C, T, w, \mathcal{R})$ where V is the alphabet of objects, $C \subset V$ is the set of catalysts, $T \subseteq (V \setminus C)$ is the alphabet of* terminal *objects, $w \in V^\circ$ is the multiset of objects initially present in the membrane region, $\mathcal{R}$ is a finite set of catalytic rules of the forms $ca \to cv$.*

The multiset in the single membrane region of Π constitutes a *configuration* of the P system. The *initial configuration* is given by the initial multiset w. A transition between configurations is governed by the application of the evolution rules, which is done in a given derivation mode.

Given a P system $\Pi = (V, C, T, w, \mathcal{R})$, the standard parallel derivation mode used in P systems is the *maximally parallel derivation mode* (max for short); in the derivation modes $max_{rules}[max]$ and $max_{objects}[max]$ only a [non-extendable] multiset of rules using the maximal number of rules and objects, respectively, is allowed to be applied.

In addition to these well-known derivation modes, in this paper we also consider several new variants of derivation modes: in the derivation modes $max_{GENobjects}[max]$ as well as $max_{\Delta objects}[max]$ only [non-extendable] multisets R of rules are taken for which the number of objects generated by the application of the rules in R to the configuration C' respectively the difference $\Delta C' = |C''| - |C'|$ between the number of objects in the configuration C'' obtained by the application of R and the number of objects in the underlying configuration C' is maximal.

3. Results

Theorem 3.1 *For any register machine with $n \geq 2$ decrementable registers we can construct a simple purely catalytic P system with only n catalysts, working in one of the derivation modes $max_{\Delta objects}max$, $max_{GENobjects}max$, $max_{\Delta objects}$, or $max_{GENobjects}$, which can simulate any computation of the register machine.*

Proof. Given an arbitrary register machine $M = (m, B, l_0, l_h, P)$ with n decrementable registers we construct a corresponding simple purely catalytic P system with n catalysts

$$\Pi = (V, \{c_k \mid 1 \leq k \leq n\}, T, w_0, \mathcal{R})$$

simulating M. The main part of the proof is to show how to simulate the instructions of M in Π; in all cases we have to take care that the n catalysts are kept busy – using corresponding dummy objects d_k with the catalysts $c_k, 1 \leq k \leq n$, The priority between different rules for a catalyst is guarded by the number of objects on the right-hand side of the rules. We use the catalyst c_r – which has to be left free for decrementing or for zero-checking in the first step of the simulation – and its "coupled" catalyst $c_{r \oplus_n 1}$ throughout all the simulation steps. Here $r \oplus_n 1$ for $r < n$ simply is $r+1$, whereas for $r = n$ we define $n \oplus_n 1 = 1$. Moreover, the notation $[1..n]$ is used for the set (interval) of natural numbers $\{1, \dots, n\}$. For every $p \in B$ we define $Reg(p)$ to be the register affected by the instruction labeled by p; in addition, we define $Reg(l_h) = 1$. The set of labels of *SUB*-instructions on register r is denoted by $B_{SUB(r)}$, the set of labels of all *ADD*-instructions by B_{ADD}. During the simulation of all instructions, we use the following multisets:

$$D'_{n,r} = \prod_{i \in [1..n] \setminus \{r, r \oplus_n 1\}} d_i, \ 1 \leq r \leq n.$$

Without loss of generality, we assume that the first instruction labeled l_0 is an *ADD*-instruction on register 1; hence, we start with the initial multiset

$$w_0 = l_0 l'_0 D'_{n,1} \prod_{i \in [1..n]} c_i.$$

$$\begin{aligned}
V &= \{a_r \mid 1 \leq r \leq m\} \cup \{\hat{a}_r \mid 1 \leq r \leq n\} \cup \{p, p' \mid p \in B_{ADD} \cup \{l_h\}\} \\
&\cup \{p, p', \bar{p}, \hat{p} \mid p \in B_{SUB(r)}, 1 \leq r \leq n\} \cup \{c_k, d_k \mid 1 \leq k \leq n\} \cup \{d\}, \\
T &= \{a_r \mid n+1 \leq r \leq m\}.
\end{aligned}$$

The dummy objects d_i, $1 \leq i \leq n$, are used to keep the corresponding catalyst c_i busy whenever it is not needed during the simulation of a *SUB*-instruction, which is accomplished by the following rule erasing d_i, but instead introducing the necessary amount of objects d to keep the catalyst c_i away from erasing a register object a_r using the rule $c_i d_i \to c_i d^4$, $1 \leq i \leq n$. Moreover, for erasing d we use the rules $c_k d \to c_k$, $1 \leq k \leq n$. In the derivation mode $max_{\Delta objects}$ these erasing rules can only be used at the end of a computation when no other rules can be applied anymore.

An *ADD*-instruction $p : (ADD\,(r)\,, q, s)$, with $p \in B_{ADD}$, $q, s \in B$, $1 \leq r \leq m$ can be simulated in one step by letting every catalyst make one evolution step:

$c_{Reg(p)} p \to c_{Reg(p)} q q' a_r d D'_{n,Reg(q)}$ or $c_{Reg(p)} p \to c_{Reg(p)} s s' a_r d D'_{n,Reg(s)}$, $c_{Reg(p) \oplus_n 1} p' \to c_2 d^4$.

We recall that all other catalysts c_i with $i \in [1..n] \setminus \{Reg(p), Reg(p) \oplus_n 1\}$ are forced to apply the rule $c_i d_i \to c_i d^4$. The dummy objects d are used to guarantee that the rules given above, with in sum at least 5 objects on their right-hand sides, have priority over the rules $c_r a_r \to c_r \hat{a}_r d^2$, $1 \le r \le n$, with in sum only 4 objects on their right-hand sides.

A *SUB*-instruction $p : (SUB(r), q, s)$, with $p \in B_{SUB}$, $q, s \in B$, $1 \le r \le n$ is simulated as follows; we emphasize that the simulation is *deterministic*.

step	$\lvert reg(r)\rvert$	rule for c_r	rule for $c_{r\oplus_n 1}$	resulting objects
first	>0	$c_r a_r \to c_r \hat{a}_r d^2$	$c_{r\oplus_n 1} p' \to c_{r\oplus_n 1} \bar{p} d^{10} D'_{n,Reg(p)}$	$p, \bar{p}, \hat{a}_r$
	$=0$	$c_r p \to c_r d^2$	$c_{r\oplus_n 1} p' \to c_{r\oplus_n 1} \bar{p} d^{10} D'_{n,Reg(p)}$	$\bar{p}$
second	>0	$c_r \bar{p} \to c_r \hat{p} d^3$	$c_{r\oplus_n 1} p \to c_{r\oplus_n 1} d^9 D'_{n,Reg(p)}$	$\hat{a}_r, \hat{p}$
	$=0$	$c_r d \to c_r \ (^*)$	$c_{r\oplus_n 1} \bar{p} \to c_{r\oplus_n 1} s s' d^6 D'_{n,Reg(s)}$	s, s'
third	>0	$c_r \hat{p} \to c_1 q q' d^2 D'_{n,Reg(q)}$	$c_{r\oplus_n 1} \hat{a}_r \to c_{r\oplus_n 1} d^4$	q, q'

The rule $c_r d \to c_r$ marked with $(^*)$ is not applied in the derivation mode $max_{\Delta objects}$.

$l_h : HALT$ is simulated by $c_1 l_h \to c_1 dd$ and $c_2 l'_h \to c_2 dd$.

Finally all dummy objects are deleted by using the rules $c_i d \to c_i,\ 1 \le i \le n$. □

References

[1] A. ALHAZOV, R. FREUND, S. IVANOV, M. OSWALD, Variants of Simple Purely Catalytic P Systems With Two Catalysts. In: GY. VASZIL, C. ZANDRON, G. ZHANG (eds.), *International Conference on Membrane Computing ICMC 2021, Proceedings*. 2021, 39–53.

[2] J. DASSOW, GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. Springer, 1989.

[3] R. FREUND, L. KARI, M. OSWALD, P. SOSÍK, Computationally Universal P Systems Without Priorities: Two Catalysts Are Sufficient. *Theoretical Computer Science* **330** (2005) 2, 251–266.

[4] R. FREUND, A. LEPORATI, G. MAURI, A. E. PORRECA, S. VERLAN, C. ZANDRON, Flattening in (Tissue) P Systems. In: A. ALHAZOV, S. COJOCARU, M. GHEORGHE, YU. ROGOZHIN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing*. Lecture Notes in Computer Science 8340, Springer, 2014, 173–188.

[5] GH. PĂUN, Computing With Membranes. *Journal of Computer and System Sciences* **61** (2000) 1, 108–143.

[6] GH. PĂUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.

[7] G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. Springer, 1997.

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 13–16. UNIVERSITÄT LEIPZIG

Quantum Automata and Basic Varieties of Languages

Fabian Birkmann$^{(A)}$ Stefan Milius$^{(A)}$ Henning Urbat$^{(A)}$

$^{(A)}$Friedrich-Alexander-Universität Erlangen-Nürnberg
{fabian.birkmann,stefan.milius,henning.urbat}@fau.de

Abstract

Eilenberg's variety theorem marked a milestone in the algebraic theory of regular languages by establishing a formal correspondence between properties of regular languages and properties of finite monoids recognizing them. Motivated by classes of languages accepted by quantum finite automata, we introduce *basic varieties of regular languages*, a weakening of Eilenberg's original concept that does not require closure under any boolean operations, and prove a variety theorem for them. To do so, we investigate the algebraic recognition of languages by *lattice bimodules*, generalizing Klíma and Polák's lattice algebras, and we utilize the duality between algebraic completely distributive lattices and posets.

1. Introduction

The introduction of algebraic methods into the study of regular languages provides a convenient classification system that allows to study finite automata and their languages in terms of associated finite algebraic structures. A celebrated example is Schützenberger's theorem [17] stating that a language is star-free iff its syntactic monoid is aperiodic, thus proving the decidability of star-freeness. Eilenberg's *variety theorem* [8] formalizes this type of correspondence as a bijection between *varieties of regular languages* (i.e. classes of regular languages closed under the set-theoretic boolean operations, word derivatives and preimages of monoid homomorphisms) and *pseudovarieties of monoids* (i.e. classes of finite monoids closed under finite products, submonoids and quotient monoids).

Numerous extensions and generalizations of Eilenberg's theorem have been discovered over the past four decades, differing from the original one by either changing the type of languages under consideration, e.g. from regular languages to ω-regular languages [19], or by considering notions of varieties with relaxed closure properties. On the algebraic side, such a relaxation requires to replace monoids by more complex algebraic structures. For instance, Pin [13] studied *positive varieties of regular languages*, where the closure under complement is dropped, and proved them to biject with pseudovarieties of *ordered* monoids. Subsequently, Polák [14] introduced *disjunctive varieties of regular languages*, where in addition to closure under complement also the closure under intersection is dropped, and related them to pseudovarieties of idempotent semirings.

One item is conspicuously missing from this list: a variety theorem for classes of languages that need not be closed under any boolean operations, i.e. in which only closure under word derivatives and preimages of monoid homomorphisms is required. In our recent work [5], we close this gap by developing the theory of such *basic varieties*. As the corresponding algebraic structure we introduce *reduced lattice bimodules*. A reduced lattice bimodule (M,D) is given by a monoid M and a complete distributive lattice D such that

(1) M acts on D from the left and right, i.e. there are binary operations $M \times D \to D$ and $D \times M \to D$ respecting the monoid and lattice structure;

(2) M embeds into D via an injective unary operation $\iota\colon M \hookrightarrow D$;

(3) the image $\iota[M] \subseteq D$ generates D as a complete lattice.

Lattice bimodules form a two-sorted generalization of the *lattice algebras* recently studied by Klíma and Polák [11]. A *pseudovariety of reduced lattice bimodules* is a class of finite reduced lattice bimodules closed under reduced homomorphic images and reduced sub-bimodules of finite products. As our main result, we establish the following algebraic classification of basic varieties:

Basic Variety Theorem. Basic varieties of regular languages correspond bijectively to pseudovarieties of reduced lattice bimodules.

This answers the open problem of Klíma and Polák [11] about an Eilenberg-type correspondence.

Our proof of the theorem is inspired by the recently developed duality-theoretic perspective on algebraic language theory [1, 9, 16, 18], which provides the insight that correspondences between language varieties and pseudovarieties of algebraic structures can be understood in terms of an underlying dual equivalence of categories. More specifically, we show that pseudovarieties of reduced lattice bimodules can be interpreted as *profinite equational theories* of lattice bimodules in the category **AlgCDL** of algebraic completely distributive lattices (equivalently, profinite distributive lattices), while basic varieties give rise to *basic cotheories* of regular languages in the category **Pos** of posets and monotone maps. Our Eilenberg correspondence for basic varieties then boils down to an application of the well-known dual equivalence

$$\mathbf{AlgCDL} \simeq^{\mathsf{op}} \mathbf{Pos}.$$

Our Basic Variety Theorem is not an instance of previous category-theoretic generalizations of Eilenberg's theorem [1, 6, 16, 18] since the two-sorted nature of lattice bimodules requires to introduce the novel concept of *reduced* structures, which makes the ensuing notion of pseudovariety more intricate than the ones studied in *op. cit.* However, much of the methodology developed there applies smoothly, which can be seen as further evidence of its scope and flexibility.

2. Towards an Algebraic Characterization of Quantum Finite Automata

Basic varieties naturally arise in several areas of automata theory, most notably in the study of languages accepted by *reversible finite automata* (*RFA*) [10] or *quantum finite automata* (*QFA*) [4]. Several different notions of QFA have been proposed and studied, varying in their expressive power. In this section, we focus on *Kondacs-Watrous quantum finite automata (KWQFA)* [12], also known in the literature as *measure-many quantum finite automata*.

A KWQFA $M = (Q, \Sigma, T, q_0, Q_{\mathsf{acc}}, Q_{\mathsf{rej}}, Q_{\mathsf{non}})$ is given by a finite set Q of *basis states*, an input alphabet Σ not containing the end markers κ and \$, an initial state $q_0 \in Q$ and a partition $Q_{\mathsf{acc}} \,\dot\cup\, Q_{\mathsf{rej}} \,\dot\cup\, Q_{\mathsf{non}}$ of Q into accepting, rejecting and non-halting states. The transitions are specified by a family of unitary linear transformations $T_\sigma\colon \mathcal{H}_Q \to \mathcal{H}_Q$ ($\sigma \in \Sigma \cup \{\kappa, \$\}$) on the complex Hilbert space $\mathcal{H}_Q$ with orthonormal basis Q. Thus, denoting the basis vectors by $|q\rangle$ ($q \in Q$), every element $|\psi\rangle$ of $\mathcal{H}_Q$ can be uniquely expressed as a linear combination

$$|\psi\rangle = \sum_{q\in Q_{\mathsf{acc}}} \alpha_q\,|q\rangle + \sum_{q\in Q_{\mathsf{rej}}} \alpha_q\,|q\rangle + \sum_{q\in Q_{\mathsf{non}}} \alpha_q\,|q\rangle \tag{2.1}$$

where $\alpha_q \in \mathbb{C}$. The *states* of M are those $|\psi\rangle \in \mathcal{H}_Q$ with norm $\sum_{q\in Q} |\alpha_q|^2 = 1$. Note that a unitary transformation T_σ maps states to states. A *measurement* collapses the state $|\psi\rangle$ to its projection onto one of the subspaces $\langle Q_{\mathsf{acc}}\rangle$, $\langle Q_{\mathsf{rej}}\rangle$, $\langle Q_{\mathsf{non}}\rangle$ generated by Q_{acc}, Q_{rej}, Q_{non}, viz.

- the projection $\sum_{q\in Q_{\mathsf{acc}}} \alpha_q\,|q\rangle$ onto $\langle Q_{\mathsf{acc}}\rangle$ with probability $\sum_{q\in Q_{\mathsf{acc}}} |\alpha_q|^2$;
- the projection $\sum_{q\in Q_{\mathsf{rej}}} \alpha_q\,|q\rangle$ onto $\langle Q_{\mathsf{rej}}\rangle$ with probability $\sum_{q\in Q_{\mathsf{rej}}} |\alpha_q|^2$;
- the projection $\sum_{q\in Q_{\mathsf{non}}} \alpha_q\,|q\rangle$ onto $\langle Q_{\mathsf{non}}\rangle$ with probability $\sum_{q\in Q_{\mathsf{non}}} |\alpha_q|^2$.

Initially, the automaton is in the basis state $|q_0\rangle$. An input $w \in \Sigma^\star$ is processed by first adding the left (κ) and right ($) end markers. Then, for every successive symbol σ in $\tilde{w} = \kappa w\$$ the corresponding transformation T_σ is applied and a measurement is performed. The automaton halts and accepts if the resulting state lies in $\langle Q_{\mathsf{acc}}\rangle$, halts and rejects if it lies in $\langle Q_{\mathsf{rej}}\rangle$, and continues with processing the next input letter if it lies in $\langle Q_{\mathsf{non}}\rangle$. Thus, if the QFA is in the state $|\psi\rangle$ of (2.1) after reading the current input symbol but before making the measurement, it accepts with probability $\sum_{q\in Q_{\mathsf{acc}}} |\alpha_q|^2$, rejects with probability $\sum_{q\in Q_{\mathsf{rej}}} |\alpha_q|^2$ and continues processing the input with probability $\sum_{q\in Q_{\mathsf{non}}} |\alpha_q|^2$. This yields an overall probability $p \in [0,1]$ that the input word w is accepted, i.e. that at any stage of the computation the automaton is observed in a state from $\langle Q_{\mathsf{acc}}\rangle$.

We say that M *accepts* the language $L \subseteq \Sigma^\star$ (with bounded error) if there exists a real number $p > 1/2$ such that M accepts every input word in L with probability $\geq p$ and rejects every input word not in L with probability $\geq p$. The class of languages accepted by KWQFA is denoted by **RMM**. It is known to be a proper subclass of the class of all regular languages; for instance, $\{a,b\}^\star a \notin$ **RMM** [12, Proposition 7]. Subsequent work has identified certain "forbidden configurations" in the minimal deterministic finite automaton of a regular language making it unrecognizable by a KWQFA [3, 7]. In this way, it was shown that **RMM** is not closed under union and intersection [3, Corollary 3.2]. However, **RMM** is closed under preimages of monoid homomorphisms and derivatives [7, Theorem 4.1], and thus forms a basic variety regular languages that is not captured by any of the previously known Eilenberg-type correspondences.

It is an open problem in the theory of quantum automata whether the class **RMM** is decidable or whether it admits an algebraic characterization [2]. Our Basic Variety Theorem provides strong evidence that such a characterization must exist: it asserts that **RMM** corresponds to a pseudovariety of reduced lattice bimodules, which by the correspondence between pseudovarieties and theories admits an equational presentation using *profinite equations* over free lattice bimodules, much analogous to Reiterman's [15] description of pseudovarieties of finite monoids in terms of profinite equations over free monoids $\Sigma^\star$. A concrete profinite axiomatization of the pseudovariety induced by **RMM** might pave the way towards the decidability of that class: deciding whether a given regular language lies in **RMM** reduces to checking whether its syntactic lattice bimodule satisfies the equational axioms.

References

[1] J. ADÁMEK, S. MILIUS, R. MYERS, H. URBAT, Generalized Eilenberg Theorem: Varieties of Languages in a Category. *ACM Trans. Comput. Log.* **20** (2019) 1, 3:1–3:47.

[2] A. AMBAINIS, M. BEAUDRY, M. GOLOVKINS, A. ĶIKUSTS, M. MERCER, D. THÉRIEN, Algebraic Results on Quantum Automata. In: *Proc. STACS*. LNCS 2996, Springer, 2004, 93–104.

[3] A. AMBAINIS, A. ĶIKUSTS, M. VALDATS, On the Class of Languages Recognizable by 1-Way Quantum Finite Automata. In: *Proc. STACS*. LNCS 2010, Springer, 2001, 75–86.

[4] A. AMBAINIS, A. YAKARYILMAZ, *Automata and Quantum Computing*, 2018. Preprint: `https://arxiv.org/abs/1507.01988`.

[5] F. BIRKMANN, S. MILIUS, H. URBAT, On Language Varieties Without Boolean Operations. In: *LATA*. LNCS 12638, Springer, 2021, 3–15.

[6] M. BOJAŃCZYK, Recognisable Languages over Monads. In: *Proc. DLT*. 9168, Springer, 2015, 1–13.

[7] A. BRODSKY, N. PIPPENGER, Characterizations of 1-Way Quantum Finite Automata. *SIAM J. Comput.* **31** (1999), 73–91.

[8] S. EILENBERG, *Automata, Languages, and Machines*. Academic Press, 1974.

[9] M. GEHRKE, S. GRIGORIEFF, J.-E. PIN, Duality and equational theory of regular languages. In: *Proc. ICALP*. LNCS 5126, Springer, 2008, 246–257.

[10] M. GOLOVKINS, J.-E. PIN, Varieties Generated by Certain Models of Reversible Finite Automata. In: *Proc. COCOON*. LNCS 4112, Springer, 2006, 83–93.

[11] O. KLÍMA, L. POLÁK, Syntactic structures of regular languages. *Theoret. Comput. Sci.* **800** (2019), 125 – 141.

[12] A. KONDACS, J. WATROUS, On the power of quantum finite state automata. In: *Proc. FOCS*. IEEE, 1997, 66–75.

[13] J.-E. PIN, A Variety Theorem Without Complementation. *Russ. Math.* **39** (1995), 80–90.

[14] L. POLÁK, Syntactic Semiring of a Language. In: *Proc. MFCS*. LNCS 2136, Springer, 2001, 611–620.

[15] J. REITERMAN, The Birkhoff theorem for finite algebras. *Algebra Universalis* **14** (1982) 1, 1–10.

[16] J. SALAMANCA, *Unveiling Eilenberg-type correspondences: Birkhoff's Theorem for (finite) algebras + duality*, 2017. Preprint: `https://arxiv.org/abs/1702.02822`.

[17] M.-P. SCHÜTZENBERGER, On finite monoids having only trivial subgroups. *Inform. and Control* **8** (1965) 2, 190–194.

[18] H. URBAT, J. ADÁMEK, L.-T. CHEN, S. MILIUS, Eilenberg Theorems for Free. In: *Proc. MFCS*. LIPIcs 83, 2017, 43:1–43:14.

[19] T. WILKE, An EILENBERG theorem for infinity-languages. In: *Proc. ICALP*. LNCS 510, Springer, 1991, 588–599.

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 17–19. UNIVERSITÄT LEIPZIG

Zur parameterisierten Komplexität formalsprachlicher Probleme

Henning Fernau[A]

[A]Universität Trier, Theoretische Informatik
fernau@uni-trier.de

Zusammenfassung

Wir beobachten, dass parameterisierte formalsprachliche (und algebraische) Probleme, sofern sie nicht trivialerweise in FPT liegen, zumeist schwer oder vollständig für 'recht hohe' Komplexitätsklassen sind, für die man aus anderen Bereichen selten Beispiele findet.

1. Einleitung

Die parameterisierte Komplexitätstheorie wurde in den 1990er Jahren von Rod Downey und Mike Fellows begründet, siehe [5]. Die Idee ist, einer Instanz nicht nur in ihrer Bitlänge zu messen, sondern (auch) in anderen Größen, die u.U. viel kleiner sind. Diese Größen nennt man *Parameter*. Auch wenn man nun bei NP-schweren Problemen keine deterministischen Polynomzeitalgorithmen erwarten kann, so könnte man versuchen, die scheinbar unvermeidliche kombinatorische Explosion auf besagten Parameter einzuschränken. Gelingt dies, liegt das Problem in FPT. Genauer wird gefordert, dass es eine berechenbare Funktion f gibt, sowie eine berechenbare Funktion κ, die Instanzen ihren Parameter zuordnet, sodass sich die Laufzeit bei Eingabe x abschätzen lässt durch $f(\kappa(x))p(|x|)$, wobei p ein Polynom ist. Ganz entsprechend lassen sich auch 'FPT-Reduktionen' definieren.

Die bekanntesten Komplexitätsklassen 'jenseits von FPT' lassen sich gut durch 'parameterisierte Halteprobleme' kennzeichnen, z.B. entspricht in diesem Sinne W[1] der Frage , bei vorgelegter nichtdeterministischer Einband-Turingmaschine, ob diese Maschine in höchstens k Schritten bei leerer Eingabe hält; hierbei ist k der Parameter. W[2] entspricht derselben Problemstellung bei Mehrband-Turingmaschinen.

Im Laufe der Jahre wurden sehr viele Probleme gefunden, die vollständig für W[1] bzw. für W[2] sind. Diese Probleme entstammen vornehmlich der Graphentheorie und der Logik. Viel weniger ist bekannt über andere parameterisierte Komplexitätsklassen.

[A]Viele der vorgestellten Ergebnisse entstanden in Zusammenarbeit mit anderen Autoren, was durch entsprechende Zitate klar werden sollte.

2. W[sync]: eine weitere Komplexitätsklasse?

Eines der Themen, mit denen wir uns in letzter Zeit häufiger beschäftigt haben, sind synchronisierende Wörter, ein Konzept, das, wie Mischa Volkov im Übersichtsartikel [10] ausführt, schon vor der berühmten Arbeit von Ján Černý [4] in der Literatur immer wieder auftaucht. Hierzu passt folgendes kombinatorisches Problem: Gegeben sei ein DEA A sowie eine Zahl k, und gefragt ist, ob A ein synchronisierendes Wort der Länge höchstens k besitzt, also ein Wort, welches A in denselben Zustand treibt, gleich wo das Lesen des Wortes beginnt. Igor Rystsov und David Eppstein haben unabhängig voneinander fast denselben NP-Vollständigkeitsbeweise hierfür geliefert; siehe [6, 9].

Es gibt verschiedene natürlich erscheinende Parameterisierungen für dieses Problem, z.B. Zustandszahl oder Eingabealphabetgröße, oder auch die Zahl k selbst. Wenn wir mit der Zustandszahl parameterisieren, erhalten wir ein Problem in FPT, wie man leicht durch eine Potenzautomatenkonstruktion einsieht. Umgekehrt ist die Eingabealphabetgröße alleine ein nicht hilfreicher Parameter, weil die schon erwähnte NP-Schwere-Konstruktion von Eppstein und Rystsov benötigt nur binäre Eingabewörter. Ähnlich war für den Parameter k alleine schon länger klar, siehe [7], dass das Problem W[2]-schwer ist. Hingegen bietet die Parameterisierung nach einer Kombination aus Eingabealphabetgröße und Längenbeschränkung ein triviales FPT-Problem, denn man kann sich schlicht erlauben, alle genügend kleinen Wörter auszuprobieren.

Die Einordnung für das mit (nur) k parameterisierte Problem bleibt jedoch unbefriedigend. Im Lauf der Zeit haben wir [1, 2] für einige weitere (zumeist formalsprachliche oder algebraische) kombinatorische Probleme gezeigt, dass sie äquivalent zum längenparameterisierten Synchronisierungsproblem für DEAs sind. Beispielsweise trifft das zu auf die Frage, ob es zu einer vorgelegten Menge von DEAs ein Wort der Länge höchstens k gibt, das von allen vorgelegten Automaten akzeptiert wird; zuvor war nur die W[2]-Schwere dieses Problems bekannt, siehe [11]. Ein weiteres zu diesen beiden parameterisierten Problemen FPT-äquivalentes Problem, eingeführt in [3] als Monoidfaktorisierungsproblem, fragt danach, ob bei vorgelegten Abbildungen $f_0, f_1, \dots, f_n$, jeweils von X nach X, es eine Auswahl $i_1, i_2, \dots, i_{k'} \in \{1, 2, \dots, n\}$ mit $k' \leq k$ gibt, sodass $f_0 = f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}}$ gilt. Die Tatsache, dass es doch mehrere Probleme gibt, die äquivalent zu dem eingangs diskutierten, mit k parameterisierten Synchronisierungsproblem sind, hat uns dazu ermutigt, W[Sync] als neue parameterisierte Komplexitätsklasse einzuführen. Wir wissen, dass W[2] eine Teilklasse von W[Sync] ist, über W[Sync] aber Klassen wie A[2] oder W[P] liegen, die wir hier nicht formal eingeführt haben.

3. Weitere Klassen und Probleme

Wenn man das oben auch diskutierte DEA-Schnittproblem nach der Zahl der vorgelegten Automaten parameterisiert, erhält man ein vollständiges Problem für eine weitere interessante, in der Literatur aber weitgehend vernachlässigte parameterisierte Komplexitätsklasse, nämlich WNL, siehe [8]. Auch diese Klasse kennt etliche vollständige parameterisierte formalsprachliche Probleme. Einige davon liefern bei anderen Parameterisierungen auch Beispiele für Probleme 'zwischen' W[2] und W[Sync].

Das Studium formalsprachlicher Probleme im Zusammenhang mit parameterisierter Komplexität könnte weitere interessante Ergebnisse auch für die Welt der parameterisierten Komplexität liefern und möglicherweise auch zur Einführung weiterer Komplexitätsklassen beitragen.

Literatur

[1] J. BRUCHERTSEIFER, H. FERNAU, Synchronizing Words and Monoid Factorization: A Parameterized Perspective. In: J. CHEN, Q. FENG, J. XU (eds.), *Theory and Applications of Models of Computation, 16th International Conference, TAMC*. LNCS 12337, Springer, 2020, 352–364.

[2] J. BRUCHERTSEIFER, H. FERNAU, Synchronizing series-parallel deterministic automata with loops and related problems. *RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications* **55** (2021) 7, 1–24.

[3] L. CAI, J. CHEN, R. DOWNEY, M. FELLOWS, On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic* **36** (1997), 321–337.

[4] J. ČERNÝ, Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis* **14** (1964) 3, 208–216.

[5] R. G. DOWNEY, M. R. FELLOWS, *Parameterized Complexity*. Springer, 1999.

[6] D. EPPSTEIN, Reset sequences for monotonic automata. *SIAM Journal on Computing* **19** (1990) 3, 500–510.

[7] H. FERNAU, P. HEGGERNES, Y. VILLANGER, A multi-parameter analysis of hard problems on deterministic finite automata. *Journal of Computer and System Sciences* **81** (2015) 4, 747–765.

[8] S. GUILLEMOT, Parameterized complexity and approximability of the Longest Compatible Sequence problem. *Discrete Optimization* **8** (2011) 1, 50–60.

[9] I. K. RYSTSOV, On minimizing the length of synchronizing words for finite automata. In: *Theory of Designing of Computing Systems*. Institute of Cybernetics of Ukrainian Acad. Sci., 1980, 75–82. (in Russian).

[10] M. V. VOLKOV, Synchronizing Automata and the Černý Conjecture. In: C. MARTÍN-VIDE, F. OTTO, H. FERNAU (eds.), *Language and Automata Theory and Applications, Second International Conference, LATA*. LNCS 5196, Springer, 2008, 11–27.

[11] H. T. WAREHAM, The parameterized complexity of intersection and composition operations on sets of finite-state automata. In: S. YU, A. PĂUN (eds.), *Implementation and Application of Automata, 5th CIAA 2000*. LNCS 2088, Springer, 2001, 302–310.

Matching Patterns with Variables under Hamming Distance

Paweł Gawrychowski[A] Florin Manea[B] Stefan Siemer[B]

[A]University of Wrocław,
Faculty of Mathematics and Computer Science, Poland
gawry@cs.uni.wroc.pl

[B]Göttingen University,
Computer Science Department and Campus-Institut Data Science, Germany
{florin.manea,stefan.siemer}@cs.informatik.uni-goettingen.de

Abstract

A pattern α is a string of variables and terminal letters. We say that α matches a word w, consisting only of terminal letters, if w can be obtained by replacing the variables of α by terminal words. The matching problem, i.e., deciding whether a given pattern matches a given word, was heavily investigated: it is NP-complete in general, but can be solved efficiently for classes of patterns with restricted structure. In this talk, we approach this problem in a generalized setting, by considering approximate pattern matching under Hamming distance. More precisely, we are interested in what is the minimum Hamming distance between w and any word u obtained by replacing the variables of α by terminal words. Firstly, we address the class of regular patterns (in which no variable occurs twice) and propose efficient algorithms for this problem, as well as matching conditional lower bounds. We show that the problem can still be solved efficiently if we allow repeated variables, but restrict the way the different variables can be interleaved according to a locality parameter. However, as soon as we allow a variable to occur more than once and its occurrences can be interleaved arbitrarily with those of other variables, even if none of them occurs more than once, the problem becomes intractable.

This paper was presented at MFCS 2021, and it is available at this link:
`https://drops.dagstuhl.de/opus/volltexte/2021/14488/`.

Optimal Regular Expressions for Palindromes of Given Length

Hermann Gruber[A] Markus Holzer[B]

[A]Knowledgepark GmbH,
Leonrodstr. 68, 80636 München, Germany
hermann.gruber@kpark.de

[B]Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
holzer@informatik.uni-giessen.de

During the last two decades or so, literally hundreds of research papers have been investigating deterministic and nondeterministic state complexity of regular languages. Here, general purpose lower bound techniques are available, and in many cases, upper and lower bounds can be obtained that match exactly, not only asymptotically. For recent surveys, see, e.g., [5, 9].

The situation is less desirable if we investigate the minimum required size of regular expressions describing a regular language. While several different lower bound techniques are available, often the best known upper and lower bounds match only asymptotically. For illustration, the size blow-up when going from finite automata over a binary alphabet to regular expressions is at least c^n for some $c > 1$ for large enough n, cf. [7]. The current record holder for the upper bound is $O(1.682^n)$, see [3]. This gives a "tight" bound of $2^{\Theta(n)}$, which is on closer inspection a bit loose. To our knowledge, exactly matching upper and lower bounds for the minimum required expression size are known only for very few nontrivial language families: Namely, the Boolean n-bit parity function [4, 8], the less-than relation on an n-set [1], given as $\{\, ij \mid 1 \leq i < j \leq n \,\}$, and the permutations of an n-set [13].

The set of all palindromes over the alphabet $\{a, b\}$ is context-free but not regular; virtually every computer science student in the world will learn this during their curriculum. Not surprisingly, this basic observation is as old as the Chomsky hierarchy itself [2]. Of course, if we consider only palindromes of a given length, the set thus obtained is finite, and therefore regular. We exactly determine the optimum regular expressions for this set, for every given length. Let P_n ($\widetilde{P}_n$, respectively) consists of all words that are palindromes of length $2n$ ($2n-1$, respectively) over a fixed binary alphabet. Then the following holds:

Theorem 1 *There is a regular expression of alphabetic width $4 \cdot 2^n - 4$ ($3 \cdot 2^n - 4$, respectively) that specifies P_n ($\widetilde{P}_n$, respectively), which is minimal w.r.t. alphabetic width among all expressions that describe P_n ($\widetilde{P}_n$, respectively).*

This is an extended abstract of: H. GRUBER, M. HOLZER, Optimal Regular Expressions for Palindromes of Given Length. In: F. BONCHI, S. J. PUGLISI (eds.), *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science*. Number 202 in Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Tallinn, Estonia, 2021, 52:1–52:15.

In the course of the proof, we also determine the optimum regular expressions for the lexicographically first k palindromes of a given length, for every k. The difficulty of course lies in establishing a matching lower bound. To this end, we use and expand a method from [13] to obtain a recurrent lower bound. The recurrence thus obtained involves a "minvolution" in the sense of [6] and the minimum operator of course yields a nonlinear recurrence and reads as follows—let $\ell(n,k)$ be the minimum alphabetic width of a regular expression describing a subset of P_n, where the subset has cardinality at least k: then $\ell(n,k)$, for $n \geq 0$ and $1 \leq k \leq 2^n$, obeys the following recurrence

$$\ell(n,k) \geq \min\{\,\ell(n-1,k)+2,\ \min_{1\leq i<k}\{\ell(n,i)+\ell(n,k-i)\}\,\},\ \text{for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$\ell(n,k) \geq \min_{1\leq i<k}\{\ell(n,i)+\ell(n,k-i)\},\ \text{for } n \geq 1 \text{ and } k > 2^{n-1},$$

and

$$\ell(n,1) = 2n.$$

A long line of research concerns asymptotic and exact solutions of recurrences involving minimum and maximum functions, see, e.g., [10] and references therein. Our recurrence falls into neither of the known categories. Nevertheless, we can show that

$$f(n,k) = 2n + 4(k-1) - 2S_2(k-1),$$

where $S_2(k)$ denotes the "digit sum to base 2" function, is a solution to the above mentioned recurrence. The digit sum to base 2 funtion function is often referred to as the *Hamming weight function* and denotes the number of ones in the binary expansion of the number k. As a side result we show that

$$\max_{0\leq i\leq n}\{S_2(i)+S_2(n-i)\} = \lambda(n+1)+S_2(n+1)-1, \tag{1}$$

for n being a nonnegative integer, which may be of its own interest. Here $\lambda(n)$ is defined as $\lambda(n) = 0$, if $n = 0$, and $\lambda(n) = \lfloor \log_2 n \rfloor$, otherwise. Remarkably, the formula on the right-hand side of the identity in Equation 1 is famously known as the number of multiplications to compute the $(n+1)$th power by the ancient Indian *Chandah-sutra method.* This appears as sequence A014701 in the On-line Encyclopedia of Integer Sequences, and is referred to as the *left-to-right binary method* [1] in [12, Chap. 4.6.3].

With the lower bound in place, it remains to give an optimal regular expression matching the lower bound. The expression $E_{n,k}$ describes the lexicographically first k palindromes of length $2n$, and is defined recursively as follows:

$$E_{n,k} = a \cdot E_{n-1,k} \cdot a,\ \text{for } n \geq 1 \text{ and } 1 \leq k \leq 2^{n-1},$$

$$E_{n,k} = a \cdot E_{n-1,2^{n-1}} \cdot a + b \cdot E_{n-1,k-2^{n-1}} \cdot b,\ \text{for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$E_{0,1} = \epsilon.$$

We can prove the following statement:

[1] We note that the formula given in [12, p. 463] refers to the *right-to-left binary method.* As explained there, the latter takes one more multiplication than the left-to-right binary method.

Theorem 2 *Let n,k be integers with $n \geq 0$ and $1 \leq k \leq 2^n$. Then the regular expression $E_{n,k}$ of alphabetic width $f(n,k)$ describes the lexicographically first k palindromes of length $2n$ over the alphabet $\{a,b\}$.*

When turning to palindromes of odd length, the lower bound recurrence essentially differs only in the terminating cases. Changing the starting conditions of a nonlinear system may, or may not, change everything. Nevertheless we obtain the following result:

Theorem 3 *Let n,k be integers with $n \geq 1$ and $1 \leq k \leq 2^n$. There is a regular expression $\widetilde{E}_{n,k}$ of alphabetic width $f(n,k)-k$, which describes the lexicographically first k palindromes of length $2n-1$ over the alphabet $\{a,b\}$.*

Next, let $\mathsf{awidth}(L)$ denote the minimum alphabetic with among all regular expressions describing L. As an anonymous reviewer pointed out, our proofs imply the following, which is reminiscent of the Kruskal-Katona Theorem:

Theorem 4 *For $n \geq 0$ and $1 \leq k \leq 2^{\lceil n/2 \rceil}$, let Pal_n denote the set of palindromes of length n, and let $\mathrm{Lex}_{n,k}$ denote the set of the lexicographically first k palindromes of length n. Then*

$$\mathrm{Lex}_{n,k} \in \operatorname*{argmin}_{\substack{|L| \geq k \\ L \subseteq \mathrm{Pal}_n}} \mathsf{awidth}(L).$$

The Kruskal-Katona Theorem is an important result in extremal combinatorics, see, e.g., [11]. Among several equivalent formulations of that theorem, one of them deals with minimization of the size of shadows in layers of the Boolean hypercube. The theorem then states that initial segments with respect to a version of the lexicographic ordering form sets with the smallest shadow possible.

With some extra effort, all of our results can be generalized to larger alphabet sizes.

References

[1] D. Chistikov, S. Iván, A. Lubiw, J. Shallit, Fractional Coverings, Greedy Coverings, and Rectifier Networks. In: H. Vollmer, B. Vallée (eds.), *Proceedings of the* 34*th Symposium on Theoretical Aspects of Computer Science*. Leibniz International Proceedings in Informatics (LIPIcs) 66, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 23:1–23:14.

[2] N. Chomsky, Three models for the description of language. *IRE Transactions on Information Theory* **2** (1956) 3, 113–124.

[3] K. Edwards, G. Farr, Improved Upper Bounds for Planarization and Series-Parallelization of Degree-Bounded Graphs. *The Electronic Journal of Combinatorics* **19** (2012) 2, #P25.

[4] K. Ellul, B. Krawetz, J. Shallit, M.-W. Wang, Regular Expressions: New Results and Open Problems. *Journal of Automata, Languages and Combinatorics* **10** (2005) 4, 407–437.

[5] Y. GAO, N. MOREIRA, R. REIS, S. YU, A Survey on Operational State Complexity. *Journal of Automata, Languages and Combinatorics* **21** (2016) 4, 251–310.

[6] D. H. GREENE, D. E. KNUTH, *Mathematics for the Analysis of Algorithms*. 2nd edition, Progress in Computer Science, Birkhäuser, 1982.

[7] H. GRUBER, M. HOLZER, Finite Automata, Digraph Connectivity, and Regular Expression Size. In: L. ACETO, I. DAMGAARD, L. A. GOLDBERG, M. M. HALLDÓRSSON, A. INGÓLFSDÓTTIR, I. WALKUWIEWICZ (eds.), *Proceedings of the 35th International Colloquium on Automata, Languages and Propgramming*. Number 5126 in LNCS, Springer, Reykjavik, Iceland, 2008, 39–50.

[8] H. GRUBER, J. JOHANNSEN, Tight Bounds on the Descriptional Complexity of Regular Expressions. In: R. AMADIO (ed.), *Proceedings of the 11th Conference Foundations of Software Science and Computational Structures*. Number 4962 in LNCS, Springer, Budapest, Hungary, 2008, 273–286.

[9] M. HOLZER, M. KUTRIB, Descriptional and Computational Complexity of Finite Automata—A Survey. *Information and Computation* **209** (2011) 3, 456–470.

[10] H.-K. HWANG, T.-H. TSAI, An asymptotic theory for recurrence relations based on minimization and maximization. *Theoretical Computer Science* **290** (2003) 3, 1475–1501.

[11] S. JUKNA, *Extremal Combinatorics: With Applications in Computer Science*. 2nd edition, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2011.

[12] D. E. KNUTH, *Seminumerical Algorithms*. The Art of Computer Programming 2, 3rd edition, Addison-Wesley, 1998.

[13] A. M. LOVETT, J. O. SHALLIT, Optimal Regular Expressions for Permutations. In: C. BAIER, I. CHATZIGIANNAKIS, P. FLOCCHINI, S. LEONARDI (eds.), *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*. LIPIcs 132, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Patras, Greece, 2019, 121:1–121:12.

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 27–30. UNIVERSITÄT LEIPZIG

State Complexity Investigations on Commutative Regular Languages

Stefan Hoffmann[(A)]

[(A)]Informatikwissenschaften, FB IV, Universität Trier, Germany,
hoffmanns@informatik.uni-trier.de

Abstract

The state complexity of a regularity-preserving operation is the maximal number of states needed in an automaton for the result of this operation dependent on the number of states of automata for the input languages. A commutative language is a language closed under permutation of letters. Here, we investigate the state complexity of the upward and downward closure and interior operations, union and intersection, the projection operation and the shuffle operation on the subclasses of commutative languages accepted by permutation automata, by counter-free automata and those with product-form minimal automata.

1. Preliminaries

The set Σ^* denotes the set of all finite sequences, i.e., of all *words*. The finite sequence of length zero, or the *empty word*, is denoted by ε. For a given word we denote by $|w|$ its *length*, and for $a \in \Sigma$ by $|w|_a$ the *number of occurrences of the symbol a* in w. A *language* is a subset of Σ^*.

The *shuffle operation*, denoted by $\sqcup\!\sqcup$, is defined by

$$u \sqcup\!\sqcup v = \{ w \in \Sigma^* \mid w = x_1 y_1 x_2 y_2 \cdots x_n y_n \textit{ for some words } \\ x_1, \ldots, x_n, y_1, \ldots, y_n \in \Sigma^* \textit{ such that } u = x_1 x_2 \cdots x_n \textit{ and } v = y_1 y_2 \cdots y_n \},$$

for $u, v \in \Sigma^*$ and $L_1 \sqcup\!\sqcup L_2 := \bigcup_{x \in L_1, y \in L_2} (x \sqcup\!\sqcup y)$ for $L_1, L_2 \subseteq \Sigma^*$.

Let $\Gamma \subseteq \Sigma$. The *projection homomorphism* $\pi_\Gamma : \Sigma^* \to \Gamma^*$ is the homomorphism given by $\pi_\Gamma(x) = x$ for $x \in \Gamma$, $\pi_\Gamma(x) = \varepsilon$ otherwise and extended by $\pi_\Gamma(\varepsilon) = \varepsilon$ and $\pi_\Gamma(wx) = \pi_\Gamma(w)\pi_\Gamma(x)$ for $w \in \Sigma^*$ and $x \in \Sigma$. As a shorthand, we set, with respect to a given naming $\Sigma = \{a_1, \ldots, a_k\}$, $\pi_j = \pi_{\{a_j\}}$. Then $\pi_j(w) = a_j^{|w|_{a_j}}$. For $L \subseteq \Sigma^*$, we set $\pi_\Gamma(L) = \{\pi_\Gamma(u) \mid u \in L\}$.

A quintuple $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is a *finite deterministic and complete automaton*, where Σ is the *input alphabet*, Q the *finite set of states*, $q_0 \in Q$ the *start state*, $F \subseteq Q$ the set of *final states* and $\delta : Q \times \Sigma \to Q$ is the *totally defined state transition function*. The transition function $\delta : Q \times \Sigma \to Q$ extends to a transition function on words $\delta^* : Q \times \Sigma^* \to Q$ by setting $\delta^*(q, \varepsilon) := q$ and $\delta^*(q, wa) := \delta(\delta^*(q, w), a)$ for $q \in Q$, $a \in \Sigma$ and $w \in \Sigma^*$. In the remainder, we drop the distinction between both functions and will also denote this extension by δ. Here, we do not consider incomplete automata. The language *recognized* (or *accepted*) by an automaton

$\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. A language $L \subseteq \Sigma^*$ is called *regular* if $L = L(\mathcal{A})$ for some finite automaton $\mathcal{A}$.

A language $L \subseteq \Sigma^*$ is a *group language*, if there exists a *permutation automaton* $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$, i.e., an automaton such that the map $q \mapsto \delta(q, a)$ is a permutation for each $a \in \Sigma$, recognizing L. An automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is a *counter-free automaton*, if for all words $w \in \Sigma^*$, states $q \in Q$ and $n \geq 1$, the condition $\delta(q, w^n) = q$ implies $\delta(q, w) = q$.

A language $L \subseteq \Sigma^*$ is *commutative*, if, for $u, v \in \Sigma^*$ such that $|v|_x = |u|_x$ for every $x \in \Sigma$, we have $u \in L$ if and only if $v \in L$.

In [1] the state complexity of the shuffle operation on general regular languages has been investigated and the following upper bound (with complete automata as a model) was obtained

$$f(n,m) = 2^{nm-1} + 2^{(m-1)(n-1)}(2^{m-1} - 1)(2^{n-1} - 1).$$

Let $u, v \in \Sigma^*$. Then, u is a *subsequence* of v, denoted by $u \preccurlyeq v$, if and only if $v \in u \sqcup\!\sqcup \Sigma^*$. The thereby given order is called the *subsequence order*. Let $L \subseteq \Sigma^*$. Then, we define the *upward closure* $\uparrow L = L \sqcup\!\sqcup \Sigma^* = \{u \in \Sigma^* : \exists v \in L : v \preccurlyeq u\}$ and the *downward closure* $\downarrow L = \{u \in \Sigma^* : u \sqcup\!\sqcup \Sigma^* \cap L \neq \emptyset\} = \{u \in \Sigma^* : \exists v \in L : u \preccurlyeq v\}$.

2. Results on General Commutative and Regular Languages

For general commutative regular languages, state complexity results are summarized in Table 1.

Operation	Upper B.	Lower Bound	Ref.
$\pi_\Gamma(U)$ $\emptyset \neq \Gamma \subseteq \Sigma$	n	n	[3, 7]
$U \sqcup\!\sqcup V$	$\begin{cases} (2nm)^{\lvert\Sigma\rvert} & \lvert\Sigma\rvert > 1 \\ nm & \lvert\Sigma\rvert = 1 \end{cases}$	$\begin{cases} nm & \lvert\Sigma\rvert \leq 2 \\ \max\left\{ \left(\frac{n-2}{\lvert\Sigma\rvert-1} + 1\right)^{\lvert\Sigma\rvert-1} \cdot m, nm \right\} & \lvert\Sigma\rvert > 2 \end{cases}$	[3, 4]
$\uparrow U$	$n^{\lvert\Sigma\rvert}$	$\Omega\left(\left(\frac{n}{\lvert\Sigma\rvert}\right)^{\lvert\Sigma\rvert}\right)$	[3, 6]
$\downarrow U$	$n^{\lvert\Sigma\rvert}$	n	[3, 6]
$U \cap V, U \cap V$	nm	sharp, for each Σ	[3, 4]

Table 1: Bounds for various operations on commutative regular languages $U, V \subseteq \Sigma^*$ accepted by automata with n and m states. For shuffle we actually have the upper bound $\min\{(2nm)^{|\Sigma|}, f(n,m)\}$. Also, the lower bound from Table 2 for shuffle could also be added, but has been left out due to space.

3. Permutation Automata and Counter-Free Automata

The state complexity on general permutation automata was investigated in [8] and the state complexity on counter-free automata was investigated in [2]. Of particular interest are these language classes in case of commutative languages, as every regular commutative language could be written as a finite union of shuffle products of a language accepted by a permutation automaton and a language accepted by a counter-free automaton [4, Theorem 11]. The results from [6] are summarized in Table 2.

	Perm. Aut.		Counter-Free Automata		
Op.	$\leq$	$\geq$	$\leq$	$\geq$	Ref.
$\pi_\Gamma(U)$	n	n	n	n	[6, 7]
$U \sqcup\!\sqcup V$	$(nm)^{\lvert\Sigma\rvert}$	nm	$(n+m-1)^{\lvert\Sigma\rvert}$	$\begin{cases} \Omega\left(\frac{m-n}{2}+\left(\frac{n}{2}\right)^2\right)(m>n) \\ \quad \text{if } \lvert\Sigma\rvert > 1 \\ n+m-1 \\ \quad \text{if } \lvert\Sigma\rvert = 1 \end{cases}$	[2, 6]
$\uparrow U$	$n^{\lvert\Sigma\rvert}$	n	$\min\{n^{\lvert\Sigma\rvert}, 2^{n-2}+1\}$	$\Omega\left(\left(\frac{n}{\lvert\Sigma\rvert}\right)^{\lvert\Sigma\rvert}\right)$	[6, 9]
$\downarrow U$	1	1	$\min\{n^{\lvert\Sigma\rvert}, 2^{n-1}\}$	n	[6, 9]
$U \cap V$ $U \cup V$	nm	nm	$\begin{cases} nm & \lvert\Sigma\rvert \geq 2 \\ \max\{n,m\} & \lvert\Sigma\rvert = 1 \end{cases}$	$\begin{cases} nm & \lvert\Sigma\rvert \geq 2 \\ \max\{n,m\} & \lvert\Sigma\rvert = 1 \end{cases}$	[2, 6]

Table 2: The state complexities of various operations for commutative input languages accepted by automata with n and m states. The upper bound ($\leq$) and the best known lower bound ($\geq$) are indicated.

4. Minimal Automata of Product-Form

A commutative regular language has a *minimal automaton of product-form*, if there exists an automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ with the minimal number of states accepting the language such that

$$\delta(q_0, u) = \delta(q_0, v) \Leftrightarrow \forall a \in \Sigma : \delta(q_0, a^{|u|_a}) = \delta(q_0, a^{|v|_a}). \tag{1}$$

It could be shown that for commutative languages, the implication from right to left is always true. For a commutative regular language with a minimal automaton of product-form , it is in fact true that the minimal automaton is isomorphic to an automaton whose state set is a cartesian product, see [5] for details. See Figure 1 for an example and a non-example.

Operation	Upper Bound	Lower Bound	Reference
$\pi_\Gamma(U), \Gamma \subseteq \Sigma$	n	n	[5]
$U \sqcup\!\sqcup V$	$2nm$	$\Omega(nm)$	[5]
$\uparrow U, \downarrow U$	n	n	[5]
$U \cap V, U \cup V$	nm	tight for each Σ	[5]

Table 3: State Complexity results on the subclass of commutative languages with product-form minimal automaton for input languages accepted by automata with n and m states.

References

[1] J. A. BRZOZOWSKI, G. JIRÁSKOVÁ, B. LIU, A. RAJASEKARAN, M. SZYKULA, On the State Complexity of the Shuffle of Regular Languages. In: C. CÂMPEANU, F. MANEA, J. O. 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, 73–86.

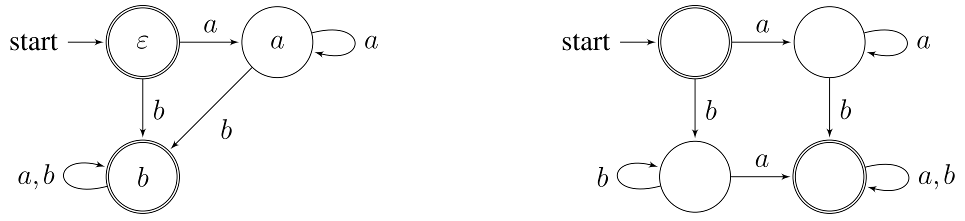

Figure 1: The automaton on the left accepts $\{u \in \{a,b\}^* : |u|_b > 0 \vee |u| = 0\}$ and does not have a minimal automaton of product-form (for the given automaton, set $u = ab$ and $v = bb$ in Equation (1)). The language $\{u \in \{a,b\}^* : |u|_a > 0 \wedge |u|_b > 0 \vee |u| = 0\}$ is accepted by the automaton on the right, which is an automaton with the least number of states fulfilling the defining Equation (1). Hence, this language has a a minimal automaton of product-form.

[2] J. A. BRZOZOWSKI, B. LIU, Quotient Complexity of Star-Free Languages. *Int. J. Found. Comput. Sci.* **23** (2012) 6, 1261–1276.

[3] S. HOFFMANN, Commutative Regular Languages – Properties and State Complexity. *Information and Computation* (to appear).

[4] S. HOFFMANN, Commutative Regular Languages - Properties and State Complexity. In: M. CIRIC, M. DROSTE, J. PIN (eds.), *Algebraic Informatics - 8th International Conference, CAI 2019, Niš, Serbia, June 30 - July 4, 2019, Proceedings*. Lecture Notes in Computer Science 11545, Springer, 2019, 151–163.

[5] S. HOFFMANN, Commutative Regular Languages with Product-Form Minimal Automata. In: G. JIRÁSKOVÁ, G. PIGHIZZINI (eds.), *Descriptional Complexity of Formal Systems - 22nd International Conference, DCFS 2021, Seoul, South Korea, June 21-24, Proceedings*. Lecture Notes in Computer Science, Springer, 2021.

[6] S. HOFFMANN, State Complexity Investigations on Commutative Languages – The Upward and Downward Closure, Commutative Aperiodic and Commutative Group Languages. In: G. JIRÁSKOVÁ, G. PIGHIZZINI (eds.), *Descriptional Complexity of Formal Systems - 22nd International Conference, DCFS 2021, Seoul, South Korea, June 21-24, Proceedings*. Lecture Notes in Computer Science, Springer, 2021.

[7] S. HOFFMANN, State Complexity of Projection on Languages Recognized by Permutation Automata and Commuting Letters. In: N. MOREIRA, R. REIS (eds.), *Developments in Language Theory - 25th International Conference, DLT 2021, Porto, Portugal, August 16-20, 2021, Proceedings*. Lecture Notes in Computer Science 12811, Springer, 2021, 192–203.

[8] M. HOSPODÁR, P. MLYNÁRCIK, Operations on Permutation Automata. In: N. JONOSKA, D. SAVCHUK (eds.), *Developments in Language Theory - 24th International Conference, DLT 2020, Tampa, FL, USA, May 11-15, 2020, Proceedings*. Lecture Notes in Computer Science 12086, Springer, 2020, 122–136.

[9] P. KARANDIKAR, M. NIEWERTH, P. SCHNOEBELEN, On the state complexity of closures and interiors of regular languages with subwords and superwords. *Theoretical Computer Science* **610** (2016), 91–107.

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 31–33. UNIVERSITÄT LEIPZIG

The Range of State Complexities of Languages Resulting from the Cascade Product

Markus Holzer[(A)] Christian Rauch[(A)]

[(A)]Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{holzer,christian.rauch}@informatik.uni-giessen.de

A *deterministic finite automaton* (DFA) is a quintuple $A = (Q,\Sigma,\cdot,q_0,F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and the *transition function* $\cdot$ maps $Q\times\Sigma$ to Q. The *language accepted* by the DFA A is defined as

$$L(A) = \{\, w \in \Sigma^* \mid q_0 \cdot w \in F \,\},$$

where the transition function is recursively extended to a mapping $Q\times\Sigma^* \to Q$ in the usual way. Obviously, every letter $a \in \Sigma$ induces a mapping on the state set Q to Q by $q \mapsto \delta(q,a)$, for every $q \in Q$. A DFA is *unary*, if the input alphabet Σ is a singleton set, that is, $\Sigma = \{a\}$, for some input symbol a. Moreover, a DFA is said to be a *permutation-reset* automaton (PRFA), if every input letter induces either a permutation or a constant mapping on the state set. If every letter of the automaton induces only permutations on the state set, then we simply speak of a *permutation* automaton (PFA). Finally, a DFA is said to be a *reset* automaton (RFA), if every letter induces either the identity or a constant mapping on the state set. The class of reset, permutation, permutation-reset, and deterministic automata in general are referred to as **RFA**, **PFA**, **PRFA**, and **FA**, respectively. It is obvious that inclusions $X\mathbf{FA} \subseteq \mathbf{PRFA} \subseteq \mathbf{FA}$, where $X \in \{\mathbf{P},\mathbf{R}\}$, holds. Moreover, it is not hard to see that the classes **RFA** and **PFA** are incomparable.

The cascade product [2] is originally introduced for semi-automata, which are automata with no initial nor final states. For our needs we enrich the cascade product with initial and final states and follow for the definition of the final states the lines of [1]. The *cascade product* of two DFAs $A = (Q_A,\Sigma,\cdot_A,q_{0,A},F_A)$ and $B = (Q_B,Q_A\times\Sigma,\cdot_B,q_{0,B},F_B)$, denoted by $A\circ B$, is defined as the automaton

$$A\circ B = (Q_A\times Q_B,\Sigma,\cdot,(q_{0,A},q_{0,B}),F_A\times F_B),$$

This is an extended abstract of: M. Holzer, C. Rauch. *The Range of State Complexities of Languages Resulting from the Cascade Product - The Unary Case (Extended Abstract).* In S. Maneth (ed.), *Proceedings of the 25th International Conference on Implementation and Application of Automata.* Number 12803 in LNCS, Springer, Bremen, Germany, 2021, 90–101 and M. Holzer, C. Rauch. *The Range of State Complexities of Languages Resulting from the Cascade Product - The General Case (Extended Abstract).* In N. Moreira and R. Reis (eds.), *Proceedings of the 25th International Conference on Developments in Language Theory.* Number 12811 in LNCS, Springer, Bremen, Germany, 2021, 229–241.

where the transition function is given by

$$(p,q)\cdot a = (p \cdot_A a, q \cdot_B (p,a)),$$

for $p \in Q_A$, $q \in Q_B$, and $a \in \Sigma$. We say that A is the *first automaton* and B the *second automaton* in the cascade product $A \circ B$.

Adapting the notion of [3] on the determinization of nondeterministic finite automata, state numbers that cannot be reached by a binary operation on two finite state devices of a particular size are called “magic.” So we specify which numbers are magic and which are reachable. Clearly we can distinguish here the size of the input alphabet Σ of the left automaton. In the unary case it turns out that a lot of magic numbers exist. The reachable state complexities in case the first automaton is unary are summarized in Table ; in most cases magic numbers exist, as highlighted by the gray shaded results.

Automata		State complexities of minimal DFAs for $L(A \circ B)$
A	B	for *unary* A
RFA	RFA	$[1,3]$
	PFA	$[1, m+1]$
	PRFA	
	DFA	
P[R]FA	RFA	$[1,2n]$
	PFA	$\{1\} \cup \{nx \mid 1 \le x \le m\} \cup \bigcup_{\substack{t \mid n \\ t \in \{2,3,\ldots,n-1\}}} \{tx \mid 1 \le x < m\}$
	PRFA	$[1,2n] \cup \{nx \mid 1 \le x \le m\} \cup \bigcup_{\substack{t \mid n \\ t \in \{2,3,\ldots,n-1\}}} \{tx \mid 1 \le x < m\}$
	DFA	$[1,nm]$
DFA	RFA	$[1,2n]$
	PFA	$\bigcup_{k=1}^{n} (M_k \oplus [0,n-k])$, where $M_k = \{1\} \cup \{kx \mid 1 \le x \le m\} \cup \bigcup_{\substack{t \mid k \\ t \in \{2,3,\ldots,k-1\}}} \{tx \mid 1 \le x < m\}$
	PRFA	
	DFA	$[1,nm]$

Table 1: The range of state complexities for the cascade product of a minimal unary n-state automaton A and a minimal m-state finite state device B of the mentioned types. Moreover, the operation $\oplus$ on sets of numbers S_1 and S_2 is defined as $S_1 \oplus S_2 = \{x + y \mid x \in S_1 \text{ and } y \in S_2\}$. In all cases where magic numbers exist (gray shaded results), except for the cascade product of two RFAs, the number $nm-1$ turns out to be magic.

If we now increase the alphabet size of A to be at least equal to two we obtain that there are nearly no magic numbers anymore.

Theorem 1 *Let $X,Y \in \{RFA, PFA, PRFA, DFA\}$ with $\{X,Y\} \ne \{PFA\}$. Moreover, let $n,m \ge 2$, where n (m, respectively) is restricted to 2 in case $X = RFA$ ($Y = RFA$, respectively). Then for every α with $1 \le \alpha \le nm$, there exists a minimal binary n-state automaton A*

of type X and a minimal m-state automaton B of type Y such that the minimal DFA for the language $L(A \circ B)$ has α states.

For PFAs we see that there are also for arbitrary alphabet sizes a lot of magic numbers.

Theorem 2 *Let $n,m \geq 2$. For every α that is* not *in*

$$\{1\} \cup \{nx \mid 1 \leq x \leq m\} \cup \bigcup_{\substack{t|n \\ t\in\{2,3,\ldots,n-1\}}} \{tx \mid 1 \leq x < m\} \cup \bigcup_{\substack{t|n \\ t\in\{2,3,\ldots,n-1\}}} \{tm\},$$

there exists no *minimal n-state PFA A and a minimal m-state PFA B such that the minimal DFA for the language $L(A \circ B)$ has α states.*

We observe that for the PFA A being unary the numbers in the first three sets are reachable. For some numbers in the fourth set $\bigcup_{\substack{t|n \\ t\in\{2,3,\ldots,n-1\}}} \{tm\}$ we have shown that they are reachable if A is allowed to be at least binary. However it remains to prove the following conjecture.

Conjecture 3 *Let $n,m \geq 2$. For every α in*

$$\bigcup_{\substack{t|n \\ t\in\{2,3,\ldots,n-1\}}} \{tm\},$$

there exists a minimal binary n-state PFA A and a minimal binary m-state PFA B such that the minimal DFA for the language $L(A \circ B)$ has α states.

References

[1] T. AE, Direct or cascade product of pushdown automata. *J. Comput. System Sci.* **14** (1977) 2, 257–263.

[2] M. A. ARBIB, *Algebraic Theory of Machines, Languages, and Semigroups*. Academic Press, 1968.

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

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 35.

Absent Subsequences in Words

Maria Kosche Tore Koß Florin Manea Stefan Siemer

Göttingen University,
Computer Science Department and Campus Institute Data Science, Germany
{maria.kosche,tore.koss,florin.manea,stefan.siemer}
@cs.uni-goettingen.de

Abstract

An absent factor of a string w is a string u which does not occur as a contiguous substring (a.k.a. factor) inside w. We extend this well-studied notion and define absent subsequences: a string u is an absent subsequence of a string w if u does not occur as subsequence (a.k.a. scattered factor) inside w. Of particular interest to us are minimal absent subsequences, i.e., absent subsequences whose every subsequence is not absent, and shortest absent subsequences, i.e., absent subsequences of minimal length. We show a series of combinatorial and algorithmic results regarding these two notions. For instance: we give combinatorial characterisations of the sets of minimal and, respectively, shortest absent subsequences in a word, as well as compact representations of these sets; we show how we can test efficiently if a string is a shortest or minimal absent subsequence in a word, and we give efficient algorithms computing the lexicographically smallest absent subsequence of each kind; also, we show how a data structure for answering shortest absent subsequence-queries for the factors of a given string can be efficiently computed.
This paper was accepted at RP 2021. The full paper can be found on arXiv: https://arxiv.org/abs/2108.13968

THEORIE-TAG 2021 A. Maletti (Herausgeber): Theorietag 2021, Leipzig, 20. – 21.9.2021
Universität Leipzig, Fakultät für Mathematik und Informatik, S. 37–40.

Weighted Hexagonal Picture Automata

Meenakshi Paramasivan[(A)] M. Gayathri Lakshmi[(B)]
D. Gnanaraj Thomas[(C)] S. James Immanuel[(D)]

[(A)]Institut für Informatik, Universität Leipzig, D-04009 Leipzig, Germany

[(B)]Research Scholar, Saveetha School of Engineering [SIMATS], Department of Mathematics, Saveetha Engineering College, Chennai 602105, India

[(C)]Department of Applied Mathematics, Saveetha School of Engineering, SIMATS, Chennai - 602105, India

[(D)]Department of Mathematics, Sri Sairam Institute of Technology, Chennai 600044, India

Abstract

Two-dimensional hexagonal arrays seen on a triangular grid can be treated as two-dimensional representations of three-dimensional rectangular parallelopipeds. We are introducing weighted three directions on-line tessellation automata (W3OTA) and investigate formal power series on hexagonal pictures.

1. Introduction

Siromoney had defined an arrowhead catenation for the two-dimensional hexagonal arrays. These arrays on a triangular grid can be viewed or treated as its two-dimensional representations of three-dimensional rectangular parallelopipeds [5]. Hexagonal cellular automata (HCA) was introduced as a variation of the rectangular two-dimensional cellular automata (RCA). Equivalence of HCA and RCA was shown. Hexagonal arrays, patterns are found in literature on picture processing, scene analysis. Hexagonal Image Processing (HIP) provides an introduction to the processing of hexagonally sampled images. The utility of the HIP framework is demonstrated by implementing several basic image processing techniques. The HIP framework serves as a tool for comparing processing of images defined on a square vs hexagonal grid [4].

In [3] two classes namely (i) local hexagonal picture languages (HLOC) and (ii) reognizable hexagonal picture languages (HREC) were introduced also hexagonal wang systems (HWS) and hexagonal tiling systems (HTS) were used to study the coincidence of these languages. In [6] three directions on-line tessellation automata was introduced to recognize HREC. Jaya Abraham et al [1] studied characterizations of hexagonal recognizable picture series through weighted hexapolic picture automata (WHPA).

Now, in this paper, we introduce weighted three directions on-line tessellation automata (W3OTA) and investigate formal power series on hexagonal pictures. We show that W3OTA recognizable series are WHPA recognizable.

2. Hexagonal Pictures

In this section, we shall briefly recall some of the required standard notations and definitions of two-dimensional hexagonal pictures and languages [3, 6].

2.1. Two-dimensional Hexagonal Pictures and Languages

A *hexagonal picture* p over the finite alphabet Σ is a hexagonal array of symbols from Σ. The set of all non-empty hexagonal pictures over Σ is denoted by Σ^{++H}. Let $p \in \Sigma^{++H}$, we get the bordered version of p denoted by $\hat{p}$ when the special symbol # $\notin \Sigma$ is added as boundary to p.

Definition 2.1 *[3] A* hexagonal tiling system *(HTS) is a* 4*-tuple* $\mathcal{T} = (\Sigma, \Gamma, \pi, \Theta)$*, where* Σ *and* Γ *are two finite alphabets,* $\pi : \Gamma \to \Sigma$ *is a projection and* Θ *is a finite set of hexagonal tiles over the alphabet* $\Gamma \cup \{\#\}$.

Note that HREC [3, 6] is exactly the family of all hexagonal picture languages recognizable by hexagonal tiling systems $\mathcal{L}(HTS)$.

3. Weighted Automata over Hexagonal Pictures

In this section, we shall briefly recall some of the required standard notations and definitions with respect to picture series and hexagonal picture series.

3.1. Series on Hexagonal Pictures

A *hexagonal picture series* [1] is a mapping $S : \Sigma^{++H} \to K$. We let $K\langle\langle\Sigma^{++H}\rangle\rangle$ contains all hexagonal picture series over Σ. We write (S,p) for $S(p)$, then a hexagonal picture series S is written as $S = \Sigma_{p\in\Sigma^{++H}}(S,p)\cdot p$. The set supp$(S) = \{p \in \Sigma^{++H} \mid (S,p) \neq 0\}$ is the *support* of S. For a language $L \subseteq \Sigma^{++H}$, the *characteristic series* $\mathbb{1}_L : \Sigma^{++H} \to K$ is defined by setting $(\mathbb{1}_L,p) = 1$ if $p \in L$, and $(\mathbb{1}_L,p) = 0$ otherwise. For $K = \mathbb{B}$, the mapping $L \mapsto \mathbb{1}_L$ gives a natural bijection between languages over Σ and series in $\mathbb{B}\langle\langle\Sigma^{++H}\rangle\rangle$.

We recall *rational* operations on hexagonal picture series $\oplus$, $\odot$, $\oslash$, $\obslash$ and $\ominus$ referred to as *sum*, *Hadamard product*, *x-directional multiplication*, *y-directional multiplication* and *z-directional multiplication* respectively, and also $\cdot : K \times K\langle\langle\Sigma^{++H}\rangle\rangle \to K\langle\langle\Sigma^{++H}\rangle\rangle$, the *scalar multiplications* with elements of the semiring [1].

In this paper, we introduce W3OTA and in order to prove the equivalence of W3OTA and WHPA, we consider two types of devices for our study on quantitative setting:

1. 3 directions on-line tessellation automata (3OTA) [6].

2. weighted hexapolic picture automata (WHPA) [1]

We now present the definition of a weighted 3 directions on-line tessellation automata. It generalizes in a straightforward way the automata-theoretic definition of recognizability for hexagonal picture languages in terms of 3OTA.

Definition 3.1 *A* weighted 3 directions on-line tessellation automata (W3OTA) *over Σ is a tuple $\mathcal{H} = (Q, E, I, F)$, consisting of a finite set Q of states, a finite set of transitions $E \subseteq Q \times Q \times Q \times \Sigma \times K \times Q$ and sets of initial and final states $I, F \subseteq Q$, respectively.*

For a transition $e = (q_x, q_y, q_z, a, w, q) \in E$, we set $\sigma_x(e) = q_x$, $\sigma_y(e) = q_y$, $\sigma_z(e) = q_z$ and $\sigma(e) = q$. We denote by label(e) *its label a and by* weight(e) *its weight w. We extend these both functions to hexagonal pictures by setting, for $c = (c_{i,j,k}) \in E^{\ell \times m \times n}$:*

$$\text{label}(c)(i,j,k) := \text{label}(c_{i,j,k}), \qquad \text{weight}(c) = \prod_{i,j,k} \text{weight}(c_{i,j,k}).$$

It defines functions label : $E^{++H} \to \Sigma^{++H}$ *and* weight : $E^{++H} \to K$. *We call* label(c) *the* label *and* weight(c) *the* weight *of c. A* run *(or* computation*) in $\mathcal{H}$ is an element in $E^{\ell \times m \times n}$ satisfying natural compatibility properties, more precisely, for $c = (c_{i,j,k}) \in E^{\ell \times m \times n}$ we have* $\forall 1 \leq i \leq \ell, 1 \leq j \leq m, 1 \leq k \leq n$:

$$\sigma_x(c_{i,j,k}) = \sigma(c_{i-1,j,k}), \quad \sigma_y(c_{i,j,k}) = \sigma(c_{i,j-1,k}), \quad \sigma_z(c_{i,j,k}) = \sigma(c_{i,j,k-1}).$$

A run $c \in E^{\ell \times m \times n}$ is successful *if for all $1 \leq i \leq \ell$, $1 \leq j \leq m$ and $1 \leq k \leq n$, we have $\sigma_x(c_{1,j,k}), \sigma_y(c_{i,1,k}), \sigma_z(c_{i,j,1}) \in I$ and $\sigma(c_{\ell,m,n}) \in F$. The set of all successful runs labelled with a hexagonal picture p is denoted by $I \overset{p}{\rightsquigarrow} F$.*

We define a hexagonal picture series $||\mathcal{H}||$ as follows. If $p \in \Sigma^{++H}$ has no successful run in $\mathcal{H}$, $||\mathcal{H}||$ sends p to 0. Otherwise, we define

$$(||\mathcal{H}||, p) = \sum_{c \in I \overset{p}{\rightsquigarrow} F} \text{weight}(c).$$

Similar to common constructions on picture automata and using ideas in [2, 1], we have the following.

Proposition 3.2 *Let K be a commutative semiring. W3OTA-recognizable hexagonal picture series over K are closed under $\odot$, $\oplus$, scalar multiplications with elements of K, projections and inverse projections. For languages, inverse projections of languages that are deterministically 3OTA-recognizable are again recognizable by some deterministic 3OTA. If L is deterministically 3OTA-recognizable then $\mathbb{1}_L$ is W3OTA-recognizable.*

Next we recall weighted hexapolic picture automata. These devices were introduced by A. Jaya et al in [1]. The family of hexagonal picture series computed by WHPA over Σ will be denoted by $K^{\text{rec}}\langle\langle \Sigma^{++H}, \mathit{WHPA} \rangle\rangle$. We call elements of this family *WHPA-recognizable*.

4. W3OTA-Recognizable Series are WHPA-Recognizable

We shall now convert a weighted 3 directions on-line tessellation automaton into a weighted hexapolic picture automaton. This inclusion is by defining some intermediate "hexagonal tiling" device, describing the context of pixels within their computation. Here these hexagonal tiles are encoded into the states of the new automaton.

Let K be a commutative semiring. For the proof of Theorem 4.5 we will first convert a given W3OTA into some “deterministic" device of a certain type via a projection similar to a construction in [1] where a Kleene-Schützenberger Theorem for hexagonal picture series is proved. However, in the present paper we apply this contruction to W3OTA rather than to WHPA. The behaviour of the constructed deterministic automaton will then be proved to be WHPA-recognizable.

Definition 4.1 *A weighted 3 directions on-line tessellation automaton is called* rule deterministic *if for every input label a of the underlying alphabet there is at most one transition with label a.*

Given a rule determinsitic W3OTA with transition set E, for $(q_x, q_y, q_z, a, w, q) \in E$ as a transition with label a we abbreviate (q_x, q_y, q_z, a, w, q) by $r(a)$.

Proposition 4.2 *Let $\mathcal{H}$ be a W3OTA over Σ. There exists a rule deterministic W3OTA $\mathcal{H}'$ over an alphabet Γ and a projection $\pi : \Gamma \to \Sigma$ satisfying $||\mathcal{H}|| = \pi(||\mathcal{H}'||)$.*

Proposition 4.3 *Every hexagonal picture series that is recognizable by a rule deterministic W3OTA is WHPA-recognizable.*

Similar to Proposition 3.2 we can prove that the family of WHPA-recognizable series are closed under projection.

Lemma 4.4 *[1] Let $\pi : \Gamma \to \Sigma$ and $S \in K^{\text{rec}}\langle\langle \Gamma^{++H}, \mathit{WHPA}\rangle\rangle$. Then $\pi(S) \in K^{\text{rec}}\langle\langle \Sigma^{++H}, \mathit{WHPA}\rangle\rangle$*

Theorem 4.5 $K^{\text{rec}}\langle\langle \Sigma^{++H}, W3OTA\rangle\rangle \subseteq K^{\text{rec}}\langle\langle \Sigma^{++H}, WHPA\rangle\rangle$

References

[1] J. Abraham, D. K. S., Characterizations of Hexagonal Recognizable Picture Series. *Journal of Global Research in Mathematical Archives* **5** (2018) 5, 65–71.

[2] S. Bozapalidis, A. Grammatikopoulou, Recognizable Picture Series. In: M. Droste, H. Vogler (eds.), *Special Issue on Weighted Automata, Presented at WATA 2004, Dresden.* 10, Journal of Automata, Languages and Combinatorics, 2005, 159–183.

[3] K. S. Dersanambika, K. Krithivasan, C. Martín-Vide, K. G. Subramanian, LOCAL AND RECOGNIZABLE HEXAGONAL PICTURE LANGUAGES. *IJPRAI* **19** (2005) 7, 853–871.

[4] L. Middleton, J. Sivaswamy, *Hexagonal Image Processing: A Practical Approach.* Advances in Pattern Recognition, Springer, 2005.

[5] G. Siromoney, R. Siromoney, Hexagonal Arrays and Rectangular Blocks. *Computer Graphics and Image Processing* **5** (1976), 353–381.

[6] D. G. Thomas, M. H. Begam, N. G. David, C. de la Higuera, Hexagonal Array Acceptors and Learning. In: M. Mukund, K. Rangarajan, K. G. Subramanian (eds.), *Formal Models, Languages and Applications [this volume commemorates the 75th birthday of Prof. Rani Siromoney].* Series in Machine Perception and Artificial Intelligence 66, World Scientific, 2007, 364–378.