Reset Complexity of Ideal Languages Over a Binary Alphabet

Marina Maslennikova

Reset Complexity of Ideal Languages Over a Binary Alphabet

Marina Maslennikova*

Ural Federal University, Ekaterinburg, Russia
maslennikova.marina@gmail.com

Abstract. We prove **PSPACE**-completeness of checking whether a given ideal language serves as the language of reset words for some automaton with at most four states over a binary alphabet.
Keywords: ideal language, synchronizing automaton, reset word, reset complexity, **PSPACE**-completeness.

Introduction

Regular languages admit compact representations by different tools: deterministic and nondeterministic finite automata, syntactic monoids, regular expressions, etc. Each of these tools gives rise to the corresponding complexity measure of regular languages. Along with these general tools, there are more specific devices for representing regular languages from some special classes. One of such classes is formed by ideal regular languages. A language $I \subseteq \Sigma^*$ is called a *two-sided ideal* (or simply an *ideal*) if I is non-empty and $\Sigma^* I \Sigma^* \subseteq I$. In what follows we consider only languages which are regular, thus we drop the adjective "regular". Thus, the expression "ideal language" (or simply "ideal") always means a regular two-sided ideal language.

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a *deterministic finite automaton* (DFA), where Q is the *state set*, Σ stands for the *input alphabet*, and $\delta\colon Q \times \Sigma \to Q$ is the totally defined *transition function* defining the action of the letters in Σ on Q. The function δ is extended uniquely to a function $Q \times \Sigma^* \to Q$, where Σ^* stands for the free monoid over Σ. The latter function is still denoted by δ. In the theory of formal languages the definition of a DFA usually includes the *initial state* $q_0 \in Q$ and the set $F \subseteq Q$ of *terminal states*. We use these ingredients when dealing with automata as devices for recognizing languages. A language $L \subseteq \Sigma^*$ is *recognized* by an automaton $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ if $L = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. We denote by $L[\mathscr{A}]$ the language recognized by the automaton $\mathscr{A}$.

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action leaves the automaton in one particular state no matter at which state in Q it is applied: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. Any word w with this

* The author acknowledges support by the Russian Foundation for Basic Research, grant no. 16-01-00795, the Ministry of Education and Science of the Russian Federation, project no. 1.3253.2017, and the Competitiveness Enhancement Program of Ural Federal University.

property is said to be *reset* for the DFA $\mathscr{A}$. For the last 50 years synchronizing automata have received a great deal of attention. For a brief introduction to the theory of synchronizing automata we refer the reader to the surveys [8, 10].

In the present paper we focus on some complexity aspects of the theory of synchronizing automata. We denote by $\mathrm{Syn}(\mathscr{A})$ the language of reset words for a given automaton $\mathscr{A}$. It is well known that $\mathrm{Syn}(\mathscr{A})$ is regular [10]. Furthermore, it is an ideal in Σ^*. On the other hand, every regular ideal language I serves as the language of reset words for some automaton. For instance, the minimal automaton recognizing I is synchronized exactly by I [4]. Thus synchronizing automata can be considered as a special representation of ideal languages. Effectiveness of such a representation was addressed in [4]. The *reset complexity* $rc(I)$ of an ideal language I is the minimal possible number of states in a synchronizing automaton $\mathscr{A}$ such that $\mathrm{Syn}(\mathscr{A}) = I$. Every such automaton $\mathscr{A}$ is called a *minimal synchronizing automaton* (for brevity, MSA). Let $sc(I)$ be the number of states in the minimal automaton recognizing I. For every ideal language I, we have $rc(I) \leq sc(I)$ [4]. Moreover, for each $n \geq 3$, there exists a language I_n such that $rc(I_n) = n$ and $sc(I_n) = 2^n - n$ [4]. Thus the representation of an ideal language by means of a synchronizing automaton can be exponentially more succinct than the "traditional" representation via the minimal automaton. This resembles the well-known property of nondeterministic finite automata (for brevity, NFAs): for each $n \geq 3$, there is an n-state NFA $\mathscr{N}$ such that every DFA recognizing the same language as $\mathscr{N}$ has at least $2^n - 1$ states [6, 7].

The following question arises: how hard is it to check that a given synchronizing DFA $\mathscr{B}$ is an MSA for a given ideal I (I is assumed to be given by a synchronizing DFA $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$)? Another question related to the previous one is the following: how hard is it to verify the inequality $rc(I) \leq \ell$ for a given ideal I and a given $\ell \in \mathbb{N}$? The inequality $rc(I) \leq \ell$ means that there exists a synchronizing DFA $\mathscr{B}$ with at most ℓ states such that $\mathrm{Syn}(\mathscr{B}) = I$. The aforementioned questions are trivial for automata over a unary alphabet, thus in what follows we deal with alphabets that have at least two letters. The problem of checking the equality $\mathrm{Syn}(\mathscr{B}) = I$ is equivalent to the problem of checking the equality $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$ for two given synchronizing automata $\mathscr{A}$ and $\mathscr{B}$. The complexity of the latter problem has been studied in [5]. It is well known that the equality of the languages recognized by two given DFAs can be checked in time polynomial of the size of automata. However, the problem of checking the equality of the languages of reset words of two synchronizing DFAs turns out to be **PSPACE**-complete [5]. Recall that the problem of checking the equality of languages recognized by two given NFAs is **PSPACE**-complete as well [9]. In this context we again find that synchronizing automata share some properties of nondeterministic finite automata.

We state formally the SYN-EQUALITY problem:

–*Input:* synchronizing automata $\mathscr{A}$ and $\mathscr{B}$.

–*Question:* is $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$?

In [5] SYN-EQUALITY has been proved to be **PSPACE**-complete. In the present paper we provide a more transparent proof of **PSPACE**-hardness of

this problem. In particular, it allows us to strengthen the result of [5] concerning the problem of evaluating the reset complexity of a given ideal language.

We state formally the RESET-INEQUALITY problem:

–*Input:* synchronizing DFA $\mathscr{A}$ over Σ, $\ell \in \mathbb{N}$.

–*Question:* is $rc(\mathrm{Syn}(\mathscr{A})) \leq \ell$?

In [5] RESET-EQUALITY has been shown to be **PSPACE**-complete for $\ell = 3$ and enough large input alphabet (with at least five letters). In the present paper we significantly strengthen this result and prove that RESET-INEQUALITY, restricted to a binary alphabet, remains **PSPACE**-complete even for $\ell = 4$. Note that RESET-INEQUALITY is trivial for DFAs over a unary alphabet and, furthermore, RESET-INEQUALITY can be solved in polynomial of the size of $\mathscr{A}$ time for $\ell = 1$ and $\ell = 2$ in the general case [5]. Thus the only question that remains open concerns the complexity of RESET-INEQUALITY for $\ell = 3$ in the case of a binary alphabet.

The paper is organized as follows. In Section 1 we introduce some definitions and preliminary results. In Section 2 we provide a modified proof of **PSPACE**-hardness of SYN-EQUALITY. Section 3 contains the main result about **PSPACE**-completeness of the problem RESET-INEQUALITY in the case of a binary alphabet.

1 Preliminaries

A state s of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is said to be a *sink* if $\delta(s, a) = s$ for all $a \in \Sigma$. If the transition function δ is clear from the context, we write $q \,.\, w$ instead of $\delta(q, w)$ for $q \in Q$ and $w \in \Sigma^*$. This notation extends naturally to any subset $H \subseteq Q$ by putting $H \,.\, w = \{\delta(q, w) \mid q \in H\}$.

Recall that a word $u \in \Sigma^*$ is a *prefix* (*suffix* or *factor*, respectively) of a word w if $w = us$ ($w = tu$ or $w = tus$, respectively) for some $t, s \in \Sigma^*$. A reset word for a DFA $\mathscr{A}$ is called *minimal* if none of its proper prefixes nor suffixes is reset.

Due to [5, Corollary 1] we have the following proposition.

Proposition 1. *SYN-EQUALITY is in* **PSPACE**.

To prove that SYN-EQUALITY is a **PSPACE**-complete problem we reduce the following well-known **PSPACE**-complete problem to the complement of SYN-EQUALITY. This problem deals with checking emptiness of the intersection of languages recognized by DFAs from a given collection [2].

FINITE AUTOMATA INTERSECTION

–*Input:* given n DFAs $M_i = \langle Q_i, \Sigma, \delta_i, q_i, F_i \rangle$, for $i = 1, \ldots, n$.

–*Question:* is $\bigcap_i L[M_i] \neq \varnothing$?

Since FINITE AUTOMATA INTERSECTION is known to be **PSPACE**-complete even in the case of a binary alphabet, we may assume that $|\Sigma| = 2$, in particular, let Σ be $\{a, b\}$.

2 PSPACE-hardness of SYN-EQUALITY

The proof of **PSPACE**-hardness of SYN-EQUALITY is based on the same idea as the proof from [5]. However, our modified construction allows us to reduce the number of letters from 5 to 4. We provide a sketch of the proof of **PSPACE**-hardness for the sake of completeness.

Given an instance of FINITE AUTOMATA INTERSECTION, we can assume without loss of generality that each initial state q_i has no incoming edges and $q_i \notin F_i$. Indeed, excluding the case for which the empty word ε is in $L[M_i]$ we can always build a DFA $M_i' = \langle Q_i', \Sigma, \delta_i', q_i', F_i\rangle$, which recognizes the same language as M_i, such that the initial state q_i' has no incoming edges. This can easily be achieved by adding a new initial state q_i' to the state set Q_i and defining the transition function δ_i' by the rule: $\delta_i'(q_i', c) = \delta_i(q_i, c)$ for all $c \in \Sigma$ and $\delta_i'(q, c) = \delta_i(q, c)$ for all $c \in \Sigma,\ q \in Q_i$. Furthermore, we may assume that the sets Q_i, for $i = 1, \ldots, n$, are pairwise disjoint. Also we can suppose that any letter from Σ does not belong to $\bigcap_i L[M_i]$. Otherwise, we add a new initial state q_i'' to each M_i' and put $\delta_i(q_i'', c) = q_i'$ for all $c \in \Sigma$. This assumption will be of use in Section 3.

To build an instance of SYN-EQUALITY from the DFAs M_i, $i = 1, \ldots, n$, we construct a DFA $\mathscr{A} = \langle Q, \Delta, \varphi\rangle$ with $Q = \bigcup_{i=1}^{n} Q_i \cup \{s, h\}$, where s and h are new states not belonging to any Q_i. We add two new letters x and z to the alphabet Σ and let $\Delta = \Sigma \cup \{x, z\}$. The transition function φ of the DFA $\mathscr{A}$ is defined by the following rules:

$$
\begin{array}{ll}
\varphi(q, c) = \delta_i(q, c) & \text{for all } i = 1, \ldots, n,\ q \in Q_i \text{ and } c \in \Sigma;\\
\varphi(q, x) = q_i & \text{for all } i = 1, \ldots, n,\ q \in Q_i;\\
\varphi(q, z) = s & \text{for all } i = 1, \ldots, n,\ q \in F_i;\\
\varphi(q, z) = h & \text{for all } i = 1, \ldots, n,\ q \in Q_i \setminus F_i;\\
\varphi(h, c) = s & \text{for all } c \in \Delta;\\
\varphi(s, c) = s & \text{for all } c \in \Delta.
\end{array}
$$

The resulting automaton $\mathscr{A}$ is shown schematically in Fig. 1. The action of letters from Σ on the states $p \in Q_i$ is not shown. Denote by G_i the set $Q_i \setminus (F_i \cup \{q_i\})$. All the states from the set G_i are shown as the node labeled by G_i. All the states from the set F_i are shown as the node labeled by F_i.

The constructed automaton is synchronizing, for example, by the word zz. It can be easily seen that by the definition of the transition function φ we get $\varphi(Q, w) \cap Q_i \neq \varnothing$ if and only if $w \in (\Sigma \cup \{x\})^*$. Consider the language

$$I = (\Sigma \cup \{x\})^* z \Delta^+.$$

From the observations above and the definition of φ we obtain Lemma 1.

Lemma 1. $\bigcap_{i=1}^{n} L[M_i] = \varnothing$ *if and only if* $\mathrm{Syn}(\mathscr{A}) = I$.

We omit the proof of Lemma 1 because of space constraints.

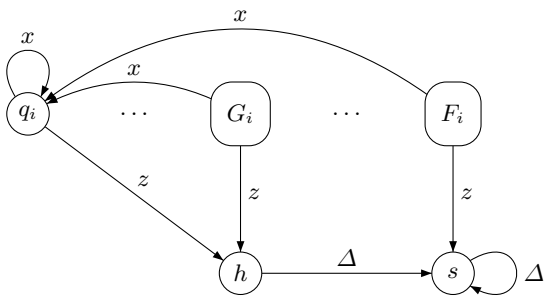

Fig. 1. Automaton $\mathscr{A}$

Now we build a 3-state automaton $\mathscr{B} = \langle P, \Delta, \tau \rangle$ (see Fig. 2). Its state set is $P = \{p_1, p_2, s'\}$, where s' is a unique sink state. It is easy to verify that I serves as the language of reset words for $\mathscr{B}$. Furthermore, I does not serve as the language of reset words for a synchronizing automaton of size at most two over the same alphabet Δ. So $\mathscr{B}$ is an MSA for I and $rc(\mathrm{Syn}(\mathscr{B})) = 3$.

Lemma 2. $\mathrm{Syn}(\mathscr{B}) = I$.

Finally, by Lemmas 1 and 2, we have the following claim.

Lemma 3. $\bigcap_{i=1}^{n} L[M_i] = \varnothing$ *if and only if* $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$.

3 PSPACE-completeness of RESET-INEQUALITY

We have reduced the problem FINITE AUTOMATA INTERSECTION to the complement of SYN-EQUALITY. By construction of DFAs $\mathscr{A}$ and $\mathscr{B}$, we have $\Delta = \{z, a, b, x\}$. Now we are going to study the complexity of checking the inequality $rc(I) \leq \ell$ in the case of a binary alphabet. First we build DFAs $\mathscr{C} = \langle C, \{\mu, \lambda\}, \varphi_2 \rangle$ and $\mathscr{D} = \langle D, \{\mu, \lambda\}, \tau_2 \rangle$ over a binary alphabet $\{\mu, \lambda\}$ with unique sink states ζ_1 and ζ_2 respectively. It turns out that the constructions of $\mathscr{C}$ and $\mathscr{D}$ preserve the equality of reset languages. More precisely, $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$ if and only if $\mathrm{Syn}(\mathscr{C}) = \mathrm{Syn}(\mathscr{D})$. Let $J = \mathrm{Syn}(\mathscr{C})$. We will prove that $rc(J) > 4$ if and only if $\bigcap_{i=1}^{n} L[M_i] \neq \varnothing$. It allows to obtain the desired result about **PSPACE**-completeness of RESET-INEQUALITY for a binary case alphabet.

In order to construct $\mathscr{C} = \langle C, \{\mu, \lambda\}, \varphi_2 \rangle$ and $\mathscr{D} = \langle D, \{\mu, \lambda\}, \tau_2 \rangle$ we apply a recoding technique which has been used in [1,3,5]. Namely, we define morphisms

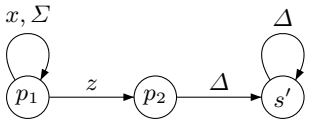

Fig. 2. Automaton $\mathscr{B}$

$h\colon \{\lambda,\mu\}^*\lambda \to \Delta^*$ and $\overline{h}\colon \Delta^* \to \{\lambda,\mu\}^*\lambda$ preserving the property of being a reset word for the corresponding automaton. Since the definitions of morphisms differ from those described in [5], we present them here. Let $d_1 = z$, $d_2 = a$, $d_3 = b$ and $d_4 = x$. We put $h(\mu^k\lambda) = d_{k+1}$ for $k = 0,\ldots,3$ and $h(\mu^k\lambda) = d_4 = x$ for $k \geq 4$. Every word from the set $\{\lambda,\mu\}^*\lambda$ can be uniquely factorized by words $\mu^k\lambda$, $k = 0,1,2,\ldots$, whence the mapping h is totally defined. We also consider the morphism $\overline{h}\colon \Delta^* \to \{\lambda,\mu\}^*\lambda$ defined by the rule $\overline{h}(d_k) = \mu^{k-1}\lambda$.

Now we take the constructed above DFA $\mathscr{B} = \langle P, \Delta, \tau\rangle$ with the state set $P = \{p_1, p_2, s'\}$. We build $\mathscr{D} = \langle D, \{\mu,\lambda\}, \tau_2\rangle$ with a unique sink state ζ_2.

We associate each state p_i of the automaton $\mathscr{B}$ with a 4-element set of states $P_i = \{p_{i,1},\ldots,p_{i,4}\}$ of the automaton $\mathscr{D}$. Namely, the states $p_{i,2}, p_{i,3}, p_{i,4}$ are copies of the state p_i associated with $p_{i,1}$. The action of the letter μ is defined in the following way: $\tau_2(p_{i,k},\mu) = p_{i,k+1}$ for $k \leq 3$, and $\tau_2(p_{i,4},\mu) = p_{i,4}$. We put $D = P_1 \cup P_2 \cup \{\zeta_2\}$, where ζ_2 is a unique sink state. The action of the letter λ is defined by the rules:
– if $\tau(p_i, d_k) = s'$, then $\tau_2(p_{i,k},\lambda) = \zeta_2$;
– if $\tau(p_i, d_k) = p_j$, then $\tau_2(p_{i,k},\lambda) = p_{j,1}$.

The latter rule means that if there is a transition from p_i to p_j labeled by the letter d_k, then there is a transition from $p_{i,1}$ to $p_{j,1}$ labeled by the word $\mu^{k-1}\lambda$.

The DFA $\mathscr{C}$ is constructed in an analogous way. Fig. 3 illustrates the automata $\mathscr{D}$ (left) and $\mathscr{C}$ (right). The actions of μ and λ are shown by solid and dotted arrows, respectively. For compactness, we do not show some transitions labeled by λ in $\mathscr{C}$. Nevertheless, the action of λ is defined on each state in $\mathscr{C}$. The resulting automata $\mathscr{C}$ and $\mathscr{D}$ have $4(|Q|-1)+1$ and 9 states respectively, where $|Q|$ is the cardinality of the state set of $\mathscr{A}$.

Lemma 4. $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$ *if and only if* $\mathrm{Syn}(\mathscr{C}) = \mathrm{Syn}(\mathscr{D})$.

Proof. It is convenient to organize the DFA $\mathscr{D}$ as a table. The set P_i is called the *i-th column* of the set D. For each $k = 1,\ldots,4$, the set $R_k = \{p_{1,k}, p_{2,k}\}$ is called the *k-th row* of the set D. The k-th row contains copies of all states corresponding to the k-th letter from Δ. The i-th column contains the state $p_{i,1}$ corresponding to the state p_i and its copies $p_{i,2}, p_{i,3}, p_{i,4}$. Each state from the i-th column maps under the action of μ to a state from the same column. For $k \leq 3$, the row R_k maps under the action of μ to the next row R_{k+1}. The 4-th row is fixed by μ, that is, $\tau_2(R_4,\mu) = R_4$. The state set D maps under the action of λ to a subset of $R_1 \cup \{\zeta_2\}$. The DFA $\mathscr{C}$ has the same properties.

Assume that $\mathrm{Syn}(\mathscr{A}) \neq \mathrm{Syn}(\mathscr{B})$. From the proof of Lemma 1 it follows that the word $w = xw'z$ with $w' \in \bigcap_i L[M_i]$ is reset for $\mathscr{A}$ and it is not reset for $\mathscr{B}$. Thus $\overline{h}(w) \in \mathrm{Syn}(\mathscr{C})$ and $\overline{h}(w) \notin \mathrm{Syn}(\mathscr{D})$. So $\mathrm{Syn}(\mathscr{C}) \neq \mathrm{Syn}(\mathscr{D})$.

Assume now that $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B})$. We show that every minimal reset word of $\mathscr{C}$ is reset for $\mathscr{D}$ and every minimal reset word of $\mathscr{D}$ is reset for $\mathscr{C}$. Let u be a minimal reset word of $\mathscr{C}$. Any word $u \in \{\mu\}^*$ is not in $\mathrm{Syn}(\mathscr{C})$ since μ brings each column to its subset. Thus u contains some factor from $\{\lambda\}^+$. The automaton $\mathscr{C}$ possesses a unique sink state ζ_1. Hence $\mathscr{C}$ is synchronized to ζ_1. Furthermore, all transitions leading to ζ_1 are labeled by λ, and ζ_1 is fixed by μ

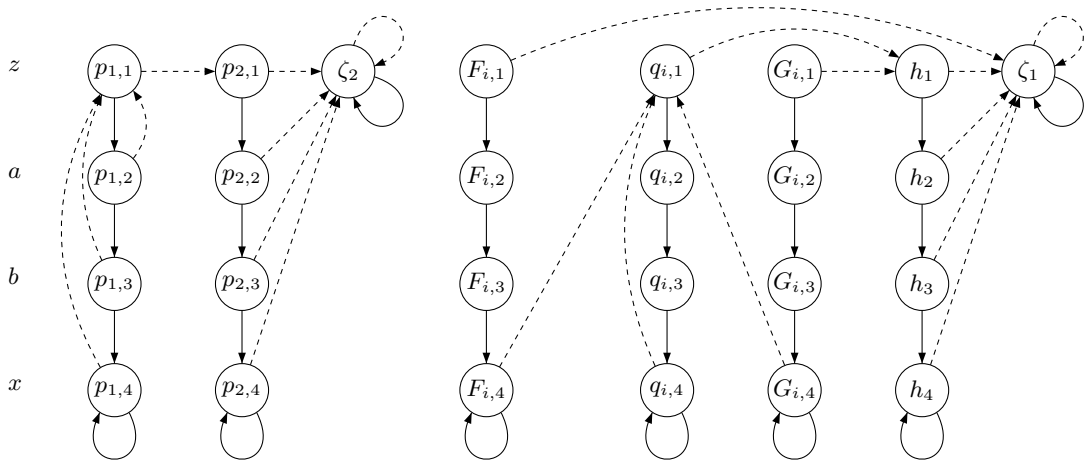

Fig. 3. The automata $\mathscr{D}$ (left) and $\mathscr{C}$ (right)

and λ. Thus if u does not end up with λ, then it is not a minimal reset word. We have $u \in \{\lambda, \mu\}^*\lambda$, i.e., $u = u'\lambda$ for some $u' \in \{\lambda, \mu\}^*$. Let us note that λ^2 appears in u as a factor (otherwise, $u \notin \mathrm{Syn}(\mathscr{C})$). If the last letter of u' is λ and λ^2 is not a factor of u', then $u \notin \mathrm{Syn}(\mathscr{C})$. If λ^2 is a factor of u', then by the definition of the transition functions of $\mathscr{C}$ and $\mathscr{D}$ we have $u \in \mathrm{Syn}(\mathscr{C})$ and $u \in \mathrm{Syn}(\mathscr{D})$. So the inclusion $\mathrm{Syn}(\mathscr{C}) \subseteq \mathrm{Syn}(\mathscr{D})$ takes place. The opposite inclusion $\mathrm{Syn}(\mathscr{D}) \subseteq \mathrm{Syn}(\mathscr{C})$ is verified analogously. □

Lemma 4 implies **PSPACE**-completeness of the problem SYN-EQUALITY restricted to a binary alphabet.

Theorem 1 (Theorem 4, [5]). *SYN-EQUALITY restricted to a binary alphabet case is* **PSPACE***-complete.*

As a corollary, we immediately obtain the following statement.

Proposition 2 ([5]). *Let ℓ be a positive integer number and $\mathscr{A}$ a synchronizing DFA. The problem of checking the inequality $rc(\mathrm{Syn}(\mathscr{A})) \leq \ell$ is in* **PSPACE***.*

So we reduced FINITE AUTOMATA INTERSECTION to the complement of SYN-EQUALITY restricted to a binary case alphabet as follows. For an arbitrary instance of FINITE AUTOMATA INTERSECTION one may build the corresponding automata $\mathscr{C}$ and $\mathscr{D}$ over a binary alphabet $\{\lambda, \mu\}$ such that $\bigcap_{i=1}^{n} L[M_i] \neq \varnothing$ if and only if $\mathrm{Syn}(\mathscr{C}) = \mathrm{Syn}(\mathscr{D})$.

The set of all words synchronizing a fixed subset $H \subseteq D$ of the state set of $\mathscr{D}$ is defined as follows:

$$\mathcal{R}(H) = \{v \in \{\lambda, \mu\}^* \mid \ H \,.\, v = \{\zeta_2\}\}.$$

Since ζ_2 is a unique sink state in $\mathscr{D}$, any reset word for $\mathscr{D}$ maps any subset of D to $\{\zeta_2\}$. Let us note that

$$\mathcal{R}(\{p_{2,1}\}) = \mathcal{R}(\{p_{2,2}\}) = \mathcal{R}(\{p_{2,3}\}) = \mathcal{R}(\{p_{2,4}\});$$
$$\mathcal{R}(\{p_{1,2}\}) = \mathcal{R}(\{p_{1,3}\}) = \mathcal{R}(\{p_{1,4}\}).$$

These equalities imply that the language of reset words of $\mathscr{D}'$ coincides with the language of reset words of $\mathscr{D}$ (see Fig. 4). Indeed, due to the equalities above we can merge states $p_{2,1}$, $p_{2,2}$, $p_{2,3}$, $p_{2,4}$ into a unique state p_2 and merge states $p_{1,2}$, $p_{1,3}$, $p_{1,4}$ into a unique state p_0. Solid arrows still denote the action of μ while dotted arrows stand for λ. In what follows we consider the DFA $\mathscr{D}'$ instead of $\mathscr{D}$.

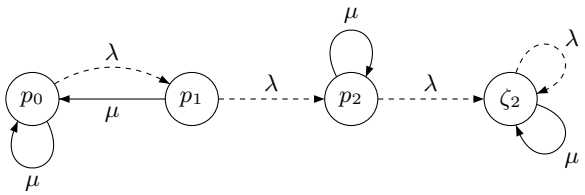

Fig. 4. The automaton $\mathscr{D}'$

A standard tool for finding the language of reset words of a given DFA $\mathscr{K} = \langle Q, \delta, \Sigma\rangle$ is the *power automaton* $\mathcal{P}(\mathscr{K})$. Its state set is the set $\mathcal{Q}$ of all nonempty subsets of Q, and the transition function is defined as a natural extension of δ to the set $\mathcal{Q} \times \Sigma$ (the resulting function is also denoted by δ), namely, $\delta(S, c) = \{\delta(q, c) \mid q \in S\}$ for $S \subseteq Q$ and $c \in \Sigma$. If we take the set Q as the initial state and singletons as final states in $\mathcal{P}(\mathscr{K})$, then we obtain an automaton recognizing $\mathrm{Syn}(\mathscr{K})$. It is easy to see that if all singletons are merged into a unique sink state s, the resulting automaton still recognizes $\mathrm{Syn}(\mathscr{K})$. Throughout the paper the term *power automaton* and the notation $\mathcal{P}(\mathscr{K})$ refer to this modified version.

Lemma 5. *Let $J = \mathrm{Syn}(\mathscr{C})$. The equality $\bigcap_{i=1}^{n} L[M_i] = \varnothing$ takes place if and only if $rc(J) \leq 4$.*

Proof. Let $\mathscr{E} = \langle P, \{\lambda, \mu\}, \gamma\rangle$ be an MSA for J. Assume that $\bigcap_{i=1}^{n} L[M_i] = \varnothing$. By Lemma 1 we get that $\mathrm{Syn}(\mathscr{A}) = \mathrm{Syn}(\mathscr{B}) = I$ with $I = (\Sigma \cup \{x\})^* z \Delta^+$. By Lemma 4 we have $J = \mathrm{Syn}(\mathscr{C}) = \mathrm{Syn}(\mathscr{D}')$. Let us note that $\lambda^3 \in \mathrm{Syn}(\mathscr{D}')$. Furthermore, $\lambda, \lambda^2 \notin \mathrm{Syn}(\mathscr{D}')$. It means that $P \,.\, \lambda^3 \subsetneq P \,.\, \lambda^2 \subsetneq P \,.\, \lambda \subsetneq P$. Hence

$|P| \geq 4$. Therefore, $rc(J) \geq 4$. On the other hand, $\mathrm{Syn}(\mathscr{D}') = J$. Thus $\mathscr{D}'$ is an MSA for $J = \mathrm{Syn}(\mathscr{C})$, i.e., $rc(J) = 4$.

Let us assume now that $\bigcap_{i=1}^{n} L[M_i] \neq \varnothing$. We are going to show that $rc(J) > 4$ in this case. Let u be a minimal reset word for $\mathscr{C}$. By the arguments from the proof of Lemma 4 and by the construction of $\mathscr{C}$, one may note that u can be factorized as $u = u'\lambda$ for some $u' \in \{\mu, \lambda\}^+$. Also λ^2 is necessarily a factor of u. Moreover, by the definition of the transition function of $\mathscr{C}$ we have $\lambda^3 \in J$ while $\lambda, \lambda^2 \notin J$. From the arguments above it follows that every MSA for J has at least 4 states. Arguing by contradiction, let us assume that there exists a 4-state automaton $\mathscr{E} = \langle P, \{\lambda, \mu\}, \gamma\rangle$ with $\mathrm{Syn}(\mathscr{E}) = J$. Without loss of generality suppose that $P = \{0, 1, 2, 3\}$. We define the action of λ on the state set P. Since $\lambda^3 \in \mathrm{Syn}(\mathscr{E})$ and $\lambda, \lambda^2 \notin \mathrm{Syn}(\mathscr{E})$, there is a unique up to isomorphism way to define the action of λ on the states from the state set P (see Fig. 5).

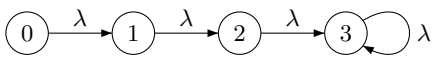

Fig. 5. The action of λ in $\mathscr{E}$

The word $\lambda^2\mu\lambda$ is in $\mathrm{Syn}(\mathscr{C})$, i.e., $\lambda^2\mu\lambda \in J$. By the definition of the action of λ in $\mathscr{E}$ we get that $\gamma(P, \lambda^2) = \{2, 3\}$. On the other hand, $\lambda^2\mu \notin \mathrm{Syn}(\mathscr{C})$, thus $\lambda^2\mu \notin \mathrm{Syn}(\mathscr{E})$. Hence $\{2, 3\}$ under the action of μ maps to a two-element subset which is translated by λ into $\{3\}$. So we need to guarantee the equality $\gamma(\{2, 3\}, \mu) = \{2, 3\}$. The action of μ at states 2 and 3 can be defined in two possible ways such that the last equality is true (see Fig. 6).

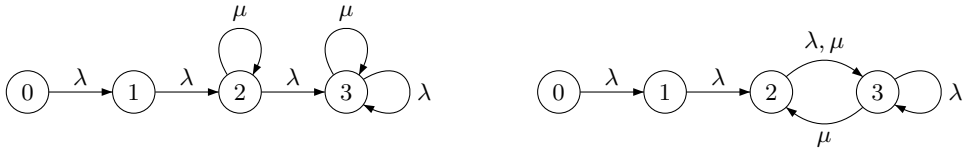

Fig. 6. The action of μ at states 2 and 3 in the DFA $\mathscr{E}$

Note that $\lambda\mu\lambda \notin \mathrm{Syn}(\mathscr{C})$. Therefore, $\lambda\mu\lambda$ is not a reset word for $\mathscr{E}$. Since we have $\gamma(\{1, 2, 3\}, \lambda\mu\lambda) = \{3\}$ for each variant from Fig. 6, the word $\lambda\mu\lambda$ should map the state 0 to a state different from 3. However $\gamma(0, \lambda) = 1$, thus 1 under the action of μ maps to either 0 or 1. All in all, for each variant from Fig. 6, we get two ways to define the image of 1 under the action of μ (see Fig. 7).

It remains to define the image of 0 under the action of μ. Let us note that $\mu^2\lambda^2 \notin \mathrm{Syn}(\mathscr{C})$. Since $\gamma(\{2, 3\}, \mu^2\lambda^2) = \{3\}$, one of the equalities, $\gamma(0, \mu) = 0$ or

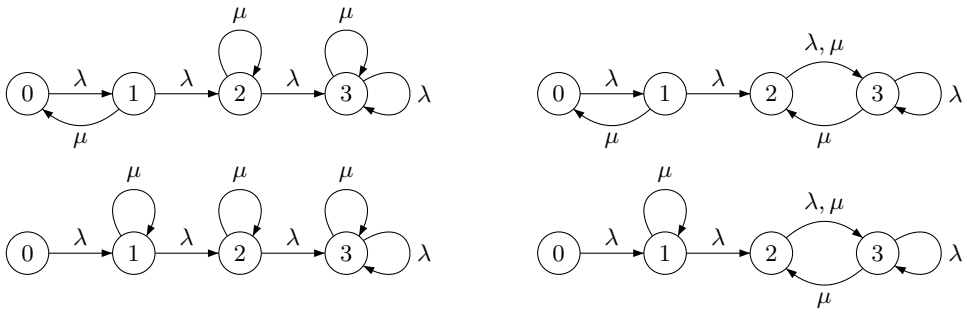

Fig. 7. Possible ways to define the action of μ at 1 in $\mathscr{E}$

$\gamma(0,\mu) = 1$, is required to take place. Furthermore, we have $\gamma(\{1,2,3\},\mu^2\lambda^2) = \{3\}$ in the third and forth variants from Fig. 7. It means that there exists the only possibility to put $\gamma(0,\mu) = 0$ for these variants. So we have six automata over the alphabet $\{\lambda,\mu\}$ shown in Fig. 8. It remains to check whether J could coincide with the set of reset words for one of these DFAs.

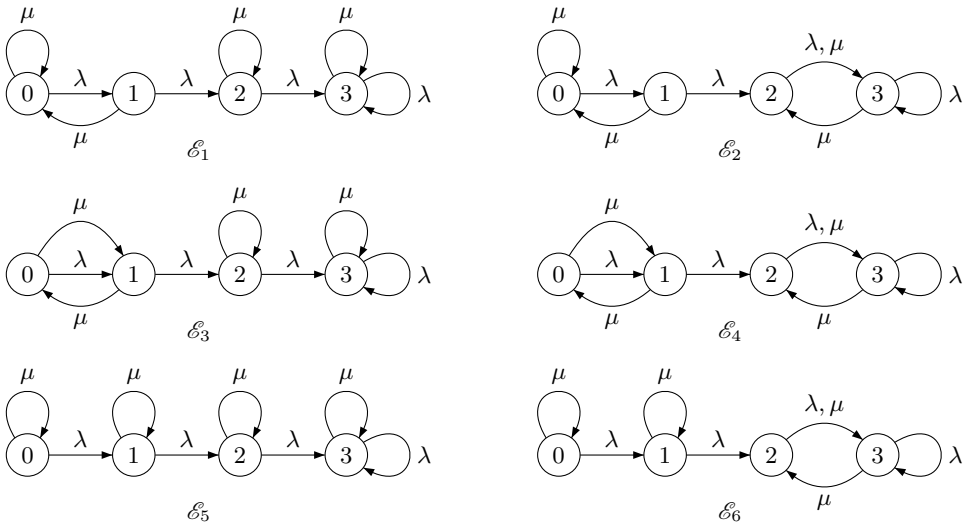

Fig. 8. Possible candidates for $\mathscr{E}$

The DFA $\mathscr{E}_1$ from Fig. 8 is isomorphic to $\mathscr{D}'$ for which we have $\mathrm{Syn}(\mathscr{D})' = \mathrm{Syn}(\mathscr{D})$. So $\mathrm{Syn}(\mathscr{E}_1) = \mathrm{Syn}(\mathscr{D})$. By the assumption $\bigcap_{i=1}^{n} L[M_i] \neq \varnothing$, hence by

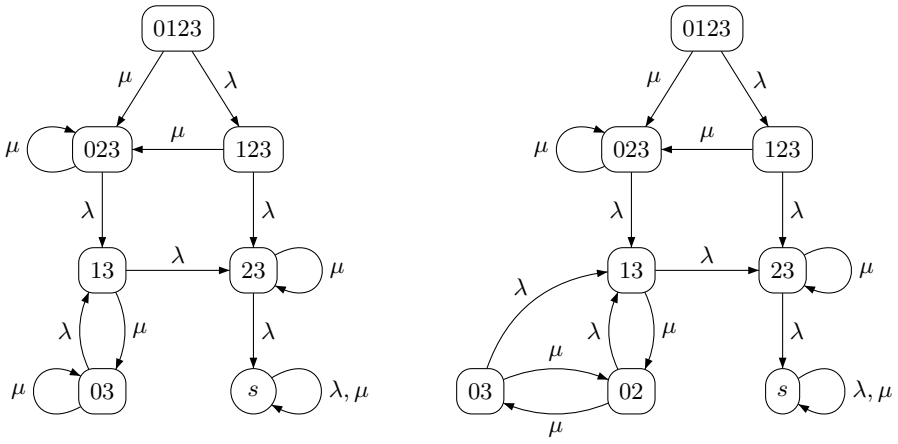

Fig. 9. The power automata $\mathcal{P}(\mathscr{E}_1)$ and $\mathcal{P}(\mathscr{E}_2)$

Lemmas 1 and 4 we have $J = \mathrm{Syn}(\mathscr{C}) \neq \mathrm{Syn}(\mathscr{D})$. Therefore, $\mathscr{E}_1$ is not an MSA for J. For $\mathscr{E}_1$ and $\mathscr{E}_2$ the equality $\mathrm{Syn}(\mathscr{E}_1) = \mathrm{Syn}(\mathscr{E}_2)$ takes place. It can be easily checked by the construction of power automata $\mathcal{P}(\mathscr{E}_1)$ and $\mathcal{P}(\mathscr{E}_2)$ (see Fig. 9). It implies that $\mathscr{E}_2$ cannot be an MSA for J.

By the assumption $c \notin \bigcap_{i=1}^{n} L[M_i]$ for all $c \in \Sigma$. It means that $b \notin L[M_j]$ or, equivalently, $\delta_j(q_j, b) \in Q_j \setminus F_j$ for some index j. Take the word xbz. By the definition of the transition function φ of $\mathscr{A}$ we have $\varphi(Q, xbz) = \{h, s\}$, so $xbz \notin \mathrm{Syn}(\mathscr{A})$. By the definition of the morphism $\overline{h}\colon \Delta^* \to \{\lambda, \mu\}^*\lambda$ we have $\overline{h}(xbz) = \mu^3\lambda\mu^2\lambda\lambda$. By the definition of the transition function φ_2 of $\mathscr{C}$ we get $\varphi_2(q_{j,1}, \mu^3\lambda\mu^2\lambda) = \varphi_2(q_{j,1}, \mu^2\lambda) = p_{t,1}$, where $p_t = \delta_j(q_j, b)$ and $p_t \notin F_j$. Hence $\varphi_2(q_{j,1}, \mu^3\lambda\mu^2\lambda\lambda) = \varphi_2(p_{t,1}, \lambda) = h_1$. So we obtain that $\mu^3\lambda\mu^2\lambda\lambda \notin \mathrm{Syn}(\mathscr{C})$. Therefore, $\mu^3\lambda\mu^2\lambda\lambda \notin J$, but it is easy to see that $\mu^3\lambda\mu^2\lambda\lambda \in \mathrm{Syn}(\mathscr{E}_3)$. Hence $\mathscr{E}_3$ can not be an MSA for J. Analogously, $\mathscr{E}_4$ can not be an MSA for J as well.

Note that $(\mu\lambda)^3 \in \mathrm{Syn}(\mathscr{E}_5)$ and $(\mu\lambda)^3 \in \mathrm{Syn}(\mathscr{E}_6)$. By the definition of the morphism $h\colon \{\lambda, \mu\}^*\lambda \to \Delta^*$, we have $h((\mu\lambda)^3) = (h(\mu\lambda))^3 = a^3$. But the word a^3 is not reset for $\mathscr{A}$ (see Fig. 1). By the definition of the transition function of $\mathscr{C}$ it implies that $\overline{h}(a^3) \notin \mathrm{Syn}(\mathscr{C})$, that is $(\mu\lambda)^3 \notin J$. In this way neither $\mathscr{E}_5$, nor $\mathscr{E}_6$ can be an MSA for J. We have considered all possible candidates for a 4-state MSA for J. Each automaton $\mathscr{E}_i$ cannot be chosen as an MSA for J. Also it is known that $rc(J) \geq 4$. Finally, we have $rc(J) > 4$. □

Now we are in position to state the main result of the paper. Lemma 5 and Proposition 2 imply the following theorem.

Theorem 2. *RESET-INEQUALITY restricted to a binary alphabet is* **PSPACE***-complete for $\ell = 4$.*

References

1. Ananichev, D.S., Gusev, V.V., Volkov, M.V.: Slowly Synchronizing Automata and Digraphs. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 55–65. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15155-2
2. Kozen, D.: Lower bounds for natural proof systems. In: 18th annual symposium on foundations of computer science, pp. 254–266. IEEE, New York (1977). doi:10.1109/SFCS.1977.16
3. Martygin, P.: Computational Complexity of Certain Problems Related to Carefuly Synchronizing Words for Partial Automata and Directing Words for Nondeterministic Automata. In: Ablayev, F. (eds.) Theory Comput. Sci. 2014, vol. 54(2), pp. 293–304. Springer (2014) doi:10.1007/s00224-013-9516-6
4. Maslennikova, M.I.: Reset Complexity of Ideal Languages. In arXiv: 1404.2816. Published in: Bieliková, M. (eds.) Int. Conf. SOFSEM 2012, vol. II, pp. 33–44. Institute of Computer Science Academy of Sciences of the Czech Republic (2012)
5. Maslennikova, M.: Complexity of checking whether two automata are synchronized by the same language. In: Jürgensen, H., Karhumäki, J., Okhotin, Al. (eds.) DCFS 2014, LNCS, vol. 8614, pp. 306–317. Springer, Heidelberg (2014). doi:10.1007/978-3-319-09704-6_27
6. Meyer, A.R, Michael, J.F.: Economy of description by automata, grammars, and formal systems. In: 12th Annual Symposium on Switching and Automata Theory, pp. 188–191. IEEE, New York (1971). doi:10.1109/SWAT.1971.11
7. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. In: IEEE Transactions on Computers, vol. C-20(10), pp. 1211–1214. IEEE, New York (1971).
8. Sandberg, S.: Homing and synchronizing sequences. In: Broy, M. (eds.) Model-Based Testing of Reactive Systems, LNCS, vol. 3472, pp. 5–33. Springer, Heidelberg (2005). doi:10.1007/b137241
9. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: Aho, A.V. (eds.) Proceedings of the 5th Annual ACM Symposium on Theory of Computing STOC '73, pp. 1–9. ACM, New York (1973) doi:10.1145/800125.804029
10. Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008, LNCS, vol. 5196, pp. 11–27. Springer, Heidelberg (2005). doi:10.1007/978-3-540-88282-4_4