Journal of Computer and System Sciences 115 (2021) 214–234

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Preimage problems for deterministic finite automata [☆]

Mikhail V. Berlinkov [a], Robert Ferens [b], Marek Szykuła [b,*]

[a] Institute of Natural Sciences and Mathematics, Ural Federal University, Ekaterinburg, Russia
[b] Institute of Computer Science, University of Wrocław, Wrocław, Poland

ARTICLE INFO

Article history:
Received 10 May 2019
Received in revised form 15 October 2019
Accepted 12 August 2020
Available online 21 August 2020

Keywords:
Avoiding word
Extending word
Extensible subset
Reset word
Synchronizing automaton

ABSTRACT

Given a subset of states S of a deterministic finite automaton and a word w, the preimage is the subset of all states mapped to a state in S by the action of w. We study three natural problems concerning words giving certain preimages. The first problem is whether, for a given subset, there exists a word *extending* the subset (giving a larger preimage). The second problem is whether there exists a *totally extending* word (giving the whole set of states as a preimage)—equivalently, whether there exists an *avoiding* word for the complementary subset. The third problem is whether there exists a *resizing* word. We also consider variants where the length of the word is upper bounded, where the size of the given subset is restricted, and where the automaton is strongly connected, synchronizing, or binary. We conclude with a summary of the complexities in all combinations of the cases.

1. Introduction

A deterministic finite complete (semi)automaton $\mathscr{A}$ is a triple (Q, Σ, δ), where Q is the set of *states*, Σ is the input *alphabet*, and $\delta\colon Q \times \Sigma \to Q$ is the *transition function*. We extend δ to a function $Q \times \Sigma^* \to Q$ in the usual way. Throughout the paper, by n we always denote the number of states $|Q|$.

When the context is clear, given a state $q \in Q$ and a word $w \in \Sigma^*$, we write shortly $q \cdot w$ for $\delta(q, w)$. Given a subset $S \subseteq Q$, the *image* of S under the action of a word $w \in \Sigma^*$ is $S \cdot w = \delta(S, w) = \{q \cdot w \mid q \in S\}$. The *preimage* is $S \cdot w^{-1} = \delta^{-1}(S, w) = \{q \in Q \mid q \cdot w \in S\}$. If $S = \{q\}$, then we usually simply write $q \cdot w^{-1}$.

We say that a word w *compresses* a subset S if $|S \cdot w| < |S|$, *avoids* S if $(Q \cdot w) \cap S = \emptyset$, *extends* S if $|S \cdot w^{-1}| > |S|$, and *totally extends* S if $S \cdot w^{-1} = Q$. A subset S is *compressible*, *avoidable*, *extensible*, and *totally extensible*, if there is a word that, respectively, compresses, avoids, extends and totally extends it.

Remark 1. A word $w \in \Sigma^*$ is avoiding for $S \subseteq Q$ if and only if w is totally extending for $Q \setminus S$.

Fig. 1 shows an example automaton. For $S = \{2, 3\}$, the shortest compressing word is *aab*, and we have $\{2, 3\} \cdot aab = \{1\}$, while the shortest extending word is *ba*, and we have $\{2, 3\} \cdot (ba)^{-1} = \{1, 2\} \cdot b^{-1} = \{1, 2, 4\}$.

Note that the preimage of a subset under the action of a word can be smaller than the subset. In this case, we say that a word *shrinks* the subset (not to be confused with compressing when the image is considered). For example, in Fig. 1, subset $\{3, 4\}$ is shrank by b to subset $\{4\}$.

[☆] A preliminary version of this work without most of the proofs was announced in [1].
[*] Corresponding author.
E-mail addresses: m.berlinkov@gmail.com (M.V. Berlinkov), robert.ferens@cs.uni.wroc.pl (R. Ferens), msz@cs.uni.wroc.pl (M. Szykuła).

https://doi.org/10.1016/j.jcss.2020.08.002
0022-0000/© 2020 Elsevier Inc. All rights reserved.

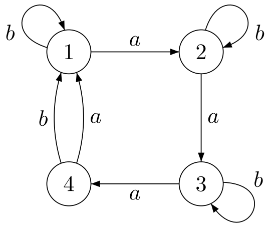

Fig. 1. The Černý automaton with 4 states.

Note that shrinking a subset is equivalent to extending its complement. Similarly, a word totally extending a subset also shrinks its complement to the empty set.

Remark 2. $|S \cdot w^{-1}| > |S|$ if and only if $|(Q \setminus S) \cdot w^{-1}| < |Q \setminus S|$, and $S \cdot w^{-1} = Q$ if and only if $(Q \setminus S) \cdot w^{-1} = \emptyset$.

Therefore, avoiding a subset is equivalent to shrinking it to the empty set.

The *rank* of a word w is the cardinality of the image $Q \cdot w$. A word of rank 1 is called *reset* or *synchronizing*, and an automaton that admits a reset word is called *synchronizing*. Also, for a subset $S \subseteq Q$, we say that a word $w \in \Sigma^*$ such that $|S \cdot w| = 1$ *synchronizes* S.

Synchronizing automata serve as transparent and natural models of various systems in many applications in different fields, e.g., in coding theory [2,3], model testing of reactive systems [4], robotics [5], and biocomputing [6]. They also reveal interesting connections with many parts of mathematics. For example, some of the recent works involve group theory [7], representation theory [8], computational complexity [9], optimization and convex geometry [10], regular languages and universality [11], approximability [12], primitive sets of matrices [13], and graph theory [14]. For a brief introduction to the theory of synchronizing automata we refer the reader to an excellent, though quite outdated, survey [15].

The famous Černý conjecture [16], which was formally stated in 1969 during a conference [15], is one of the most longstanding open problems in automata theory. It states that a synchronizing automaton has a reset word of length at most $(n-1)^2$. The currently best upper bound is cubic and has been improved recently [17] (cf. [18]). Besides the conjecture, algorithmic issues are also important. Unfortunately, the problem of finding a *shortest* reset word is computationally hard [19,9], and also its length approximation remains hard [12]. We also refer to surveys [4,15] dealing with algorithmic issues and the Černý conjecture.

Compressing and extending a subset in general play a crucial role in the synchronization of automata and related areas. In fact, all known algorithms finding a reset word use finding words that either compresses or extends a subset as subprocedures (e.g. [20,21,19,22,23]). Moreover, probably all proofs of upper bounds on the length of the shortest reset words use bounds on the length of words that compress (e.g. [20,24,21,25,19,26,18,27,28]) or extend (e.g. [29,30,21,31–33,18]) some subsets.

In this paper, we study several problems about finding a word yielding a certain preimage. We provide a systematic view of their computational complexity in various combinations of cases.

1.1. Compressing a subset

The complexities of problems related to images of a subset have been well studied. It is known that given an automaton $\mathscr{A}$ and a subset $S \subseteq Q$, determining whether there is a word that synchronizes it is PSPACE-complete [34]. The same holds even for strongly connected binary automata [35].

On the other hand, checking whether the automaton is synchronizing, i.e. whether there is a word that synchronizes Q, can be solved in $\mathcal{O}(|\Sigma|n^2)$ time and space [16,19,15] and in $\mathcal{O}(n)$ average time and space when the automaton is randomly chosen [36]. To this end, we verify whether all pairs of states are compressible. Using the same algorithm, we can determine whether a given subset is compressible.

Deciding whether there exists a synchronizing word of a given length is NP-complete [19] (cf. [9] for the complexity of the corresponding functional problems), even if the given automaton is binary. The NP-completeness holds even when the automaton is Eulerian and binary [37], which immediately implies that for the class of strongly connected automata the complexity is the same.

However, deciding whether there exists a word of a given length that only compresses a subset still can be solved in $\mathcal{O}(|\Sigma|n^2)$ time, as for every pair of states we can compute a shortest word that compresses the pair.

The problems related to images have been also studied in other settings for both complexity and the bounds on the length of the shortest words, for example, in the case of a nondeterministic automaton [34], in the case of a partial deterministic finite automaton [38], in the partial observability setting for various kinds of automata [39], and for the reachability of a given subset in the case of a deterministic finite automaton [40,41].

1.2. Extending a subset and our contributions

In contrast to the problems related to images (compression), the complexity of the problems related to preimages has not been thoroughly studied in the literature. In the paper, we fill this gap and give a comprehensive analysis of all basic cases. We study three families of problems. As noted before, extending is equivalent to shrinking the complementary subset, hence we need to deal only with the extending word problems. Similarly, totally extending words are equivalent to avoiding the complement, thus we do not need to consider avoiding a set of states separately.

Extending words: Our first family of problems is the question whether there exists an extending word (Problems 1, 3, 5, 7, 9, 12 in this paper).

This is motivated by the fact that finding such a word is the basic step of the so-called *extension method* of finding a reset word, which is used in many proofs and also some algorithms. The extension method of finding a reset word is as follows: we start from some singleton $S_0 = \{q\}$ and iteratively find extending words $w_1, \dots, w_k$ such that $|S_0 \cdot w_1^{-1} \cdots w_i^{-1}| > |S_0 \cdot w_1^{-1} \cdots w_{i-1}^{-1}|$ for $1 \le i \le k$, and where $S_0 \cdot w_1^{-1} \cdots w_k^{-1} = Q$. For finding a short reset word one needs to bound the lengths of the extending words. For instance, in the case of synchronizing Eulerian automata, the fact that there always exists an extending word of length at most $n-1$ implies the upper bound $(n-2)(n-1)+1$ on the length of the shortest reset words for this class [32] (the first extending step requires just one letter, as we can choose an arbitrary singleton). In this case, a polynomial algorithm for finding extending words has been proposed [21].

Totally extending words and avoiding: We study the problem whether there exists a totally extending word (Problems 2, 4, 6, 8, 10, 13 in this paper). The question of the existence of a totally extending word is equivalent to the question of the existence of an avoiding word for the complementary subset.

Totally extending words themselves can be viewed as a generalization of reset words: a word totally extending a singleton to the whole set of states Q is a reset word. If we are not interested in bringing the automaton into one particular state but want it to be in any of the states from a specified subset, then it is exactly the question about totally extending word for our subset. In view of applications of synchronization, this can be particularly useful when we deal with non-synchronizing automata, where reset words cannot be applied.

Avoiding word problem is a recent concept that is dual to synchronization: instead of being in some states, we want not to be in them. A quadratic upper bound on the length of the shortest avoiding words of a single state has been established [18], which led to an improvement of the best known upper bound on the length of the shortest reset words (see also [17] for a very recent improvement of that improvement of the upper bound). Furthermore, better upper bounds on the length of the shortest avoiding words would lead to further improvements; in particular, a subquadratic upper bound implies the upper bound on the reset threshold equal to $7n^3/48 + o(n^3)$ [42]. There is a precise conjecture that the shortest avoiding words have length at most $2n-2$ [18, Open Problem 1]. The computational complexity of the problems related to avoiding, both a single state or a subset, has not been established before. We give a special attention to the problem of avoiding one state and a small subset of states (totally extending a large subset), as since they seem to be most important in view of their applications (and as we show, the complexity grows with the size of the subset to avoid).

Resizing: Shrinking a subset is dual to extending, i.e. shrinking a subset means extending its complement. Therefore, the complexity immediately transfers from the previous results. However, in Section 5 we consider the problem of determining whether there is a word whose inverse action results in a subset having a different size, that is, either extends the subset or shrinks it (Problems 15, 16).

Interestingly, in contrast with the computationally difficult problems of finding a word that extends the subset and finding a word that shrinks the subset, for this variant there exists a polynomial algorithm finding a shortest resizing word in all cases.

We can mention that in some cases extending and shrinking words are related, and it may be enough to find either one. For instance, this is used in the so-called *averaging trick*, which appears in several proofs [21,31,32,43].

Summary: For all the problems we consider the subclasses of strongly connected, synchronizing, and binary automata. Also, we consider the problems where an upper bound on the length of the word is additionally given in a binary form in the input. Since, in most cases, the problems are computationally hard, in Section 3 and Section 4, we consider the complexity parameterized by the size of the given subset.

Table 1 and Table 2 summarize our results together with known results about compressing words. For the cases where a polynomial algorithm exists, we put the time complexity of the best one known. All the hardness results hold also in the case of a binary alphabet.

2. Extending a subset in general

2.1. Unbounded word length

In the first studied case, we do not have any restriction on the given subset S neither on the length of the extending word. We deal with the following problems:

Table 1
The computational complexity of decision problems (new results are in bold): given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ with n states and a subset $S \subseteq Q$, is there a word $w \in \Sigma^*$ such that:

Problem	Subclass of automata			
	All automata	Strongly connected	Synchronizing	Str. con. and synch.
$\|S \cdot w\| = 1$ (reset word)	PSPACE-c [34,35]		$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\|S \cdot w\| < \|S\|$ (compressing word)	$\mathcal{O}(\|\Sigma\|n^2)$ [16,15]		$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\|S \cdot w^{-1}\| > \|S\|$ (Problem 1)	**PSPACE-c** (Thm. 3)		**PSPACE-c** (Prop. 5)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$ (Problem 2)	**PSPACE-c** (Thm. 3)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n)}$ (Thm. 6)	$\mathcal{O}(1)$
$\|S \cdot w^{-1}\| > \|S\|$, $\|S\| \le k$ (Problem 5)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k)}$ (Prop. 7)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k)}$ (Prop. 7)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\|S\| \le k$ (Problem 6)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k + n^3)}$ (Prop. 8)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n)}$ (Thm. 6)	$\mathcal{O}(1)$
$\|S \cdot w^{-1}\| > \|S\|$, $\|S\| \ge n - k$ (Problem 9, $k \ge 2$)	**PSPACE-c** (Thm. 10)	Open	**PSPACE-c** (Thm. 10)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\|S\| \ge n - k$ (Problem 10, $k \ge 2$)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k + n^3)}$ (Thm. 12)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n)}$ (Thm. 6)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\|S\| = n - 1$ (Problem 11)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^2)}$ (Thm. 11)		$\mathcal{O}(\|\Sigma\|)$	$\mathcal{O}(1)$
$\|S \cdot w^{-1}\| \ne \|S\|$ (Problem 15)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^3)}$ (Thm. 15)		$\mathcal{O}(1)$	$\mathcal{O}(1)$

Table 2
The computational complexity of decision problems (new results are in bold): given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ with n states, a subset $S \subseteq Q$, and an integer ℓ given in binary form, is there are a word $w \in \Sigma^*$ of length $\le \ell$ such that:

Problem	Subclass of automata			
	All automata	Strongly connected	Synchronizing	Str. con. and synch.
$\|S \cdot w\| = 1$ (reset word)	PSPACE-c [34,35]		NP-c [19]	NP-c [37]
$\|S \cdot w\| < \|S\|$ (compressing word)	$\mathcal{O}(\|\Sigma\|n^2)$ [19]		$\mathcal{O}(\|\Sigma\|n^2)$ [19]	$\mathcal{O}(\|\Sigma\|n^2)$ [19]
$\|S \cdot w^{-1}\| > \|S\|$ (Problem 3)	**PSPACE-c** (Subsec. 2.2)		**PSPACE-c** (Subsec. 2.2)	**NP-c** (Thm. 13)
$S \cdot w^{-1} = Q$ (Problem 4)	**PSPACE-c** (Subsec. 2.2)		**NP-c** (Cor. 14)	**NP-c** (Cor. 14)
$\|S \cdot w^{-1}\| > \|S\|$, $\|S\| \le k$ (Problem 7	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k)}$ (Prop. 7)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k)}$ (Prop. 7)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^k)}$ (Prop. 7)
$S \cdot w^{-1} = Q$, $\|S\| \le k$ (Problem 8)	**NP-c** (Prop. 9)		**NP-c** (Prop. 9)	**NP-c** (Prop. 9)
$\|S \cdot w^{-1}\| > \|S\|$, $\|S\| \ge n - k$ (Problem 12, $k \ge 2$)	**PSPACE-c** (Thm. 10)	Open	**PSPACE-c** (Thm. 10)	**NP-c** (Cor. 14)
$S \cdot w^{-1} = Q$, $\|S\| \ge n - k$ (Problem 13, $k \ge 2$)	**NP-c** (Cor. 14)		**NP-c** (Cor. 14)	**NP-c** (Cor. 14)
$S \cdot w^{-1} = Q$, $\|S\| = n - 1$ (Problem 14)	**NP-c** (Thm. 13)		**NP-c** (Thm. 13)	**NP-c** (Thm. 13)
$\|S \cdot w^{-1}\| \ne \|S\|$ (Problem 16)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^3)}$ (Thm. 15)		$\boldsymbol{\mathcal{O}(\|\Sigma\|n^3)}$ (Thm. 15)	$\boldsymbol{\mathcal{O}(\|\Sigma\|n^3)}$ (Thm. 15)

Problem 1 *(Extensible subset).* Given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is S extensible?

Problem 2 *(Totally extensible subset).* Given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is S totally extensible?

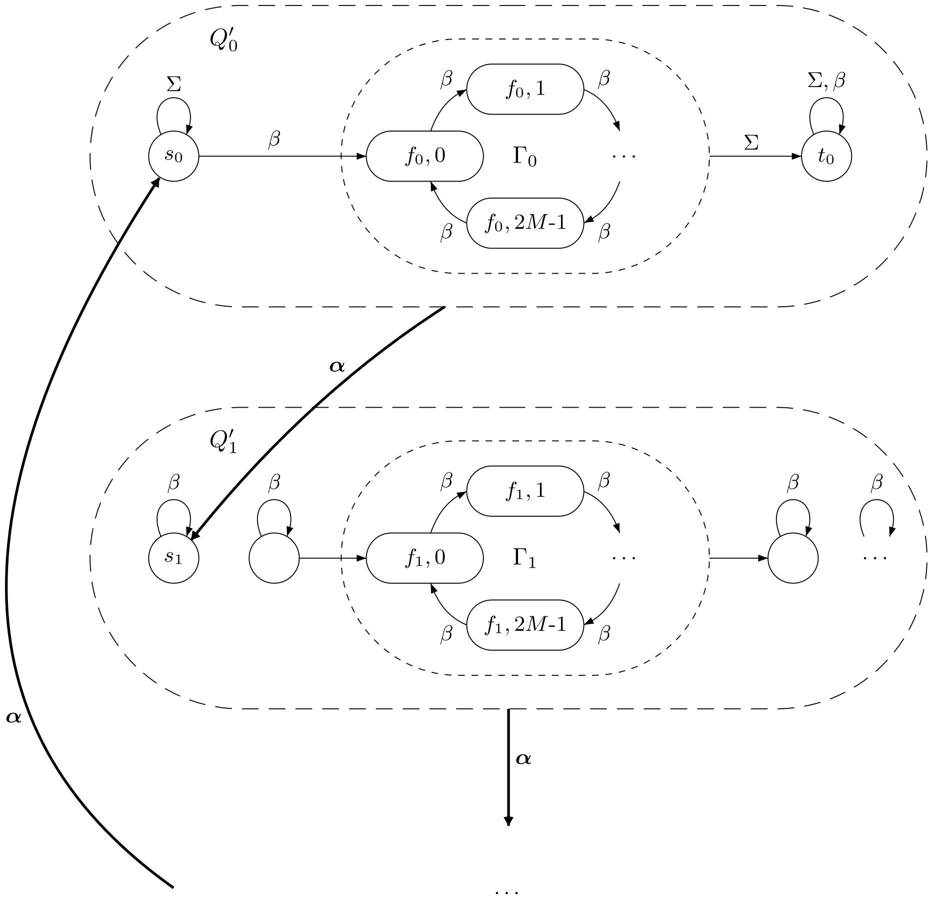

Fig. 2. The automaton $\mathcal{D}'$ from the proof of Theorem 3.

Theorem 3. *Problem 1 and Problem 2 are PSPACE-complete, even if $\mathscr{A}$ is strongly connected.*

Proof. To solve one of the problems in NPSPACE, we guess the length of a word w with the required property, and then guess the letters of w from the end. Of course, we do not store w, which may have exponential length, but just keep the subset $S \cdot u^{-1}$, where u is the current suffix of w. The current subset can be stored in $\mathcal{O}(n)$, and since there are 2^n different subsets, $|w| \leq 2^n$ and the current length also can be stored in $\mathcal{O}(n)$. By Savitch's theorem, the problems are in PSPACE.

For PSPACE-hardness, we construct a reduction from the problem of determining whether an intersection of regular languages given as DFAs is non-empty. We create one instance for both problems that consists of a strongly connected automaton and a subset S extensible if and only if it is also totally extensible, which is simultaneously equivalent to the non-emptiness of the intersection of the given regular languages.

Let $(\mathcal{D}_i)_{i \in \{1,\dots,m\}}$ be the given sequence of DFAs with an i-th automaton $\mathcal{D}_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$ recognizing a language L_i, where Q_i is the set of states, Σ is the common alphabet, δ_i is the transition function, s_i is the initial state, and F_i is the set of final states. The problem whether there exists a word accepted by all $\mathcal{D}_1, \dots, \mathcal{D}_m$ (i.e. the intersection of L_i is non-empty) is a well known PSPACE-complete problem, called Finite Automata Intersection [44]. We can assume that the DFAs are *minimal*; in particular, they do not have unreachable states from the initial state, otherwise, we may easily remove them in polynomial time.

For each $\mathcal{D}_i$ we choose an arbitrary $f_i \in F_i$. Let $M = \sum_{i=1}^{m} |Q_i|$. We construct the (semi)automaton $\mathcal{D}' = (Q', \Sigma', \delta')$ and define $S \subseteq Q'$ as an instance of our both problems. The scheme of the automaton is shown in Fig. 2.

- For $i \in \{0, 1, \dots, m\}$, let $\Gamma_i = \{f_i\} \times \{0, \dots, 2M-1\}$ be fresh states and let $Q'_i = (Q_i \setminus \{f_i\}) \cup \Gamma_i$. Let $Q'_0 = \{s_0, t_0\} \cup \Gamma_0$, where s_0 and t_0 are fresh states. Then $Q' = \bigcup_{i=0}^{m} Q'_i$.
- $\Sigma' = \Sigma \cup \{\alpha, \beta\}$, where α and β are fresh letters.

- δ' is defined by:
  - For $q \in Q_i \setminus \{f_i\}$ and $a \in \Sigma$, we have
$$\delta'(q,a) = \begin{cases} \delta_i(q,a) & \text{if } \delta_i(q,a) \neq f_i, \\ (f_i,0) & \text{otherwise.} \end{cases}$$
  - For $a \in \Sigma$, we have
$$\delta'(t_0,a) = t_0, \quad \delta'(s_0,a) = s_0.$$
  - For $k \in \{0,\dots,2M-1\}$, $i \in \{1,\dots,m\}$, and $a \in \Sigma$, we have
$$\delta'((f_0,k),a) = t_0,$$
$$\delta'((f_i,k),a) = \begin{cases} \delta_i(f_i,a) & \text{if } \delta_i(f_i,a) \neq f_i, \\ (f_i,0) & \text{otherwise.} \end{cases}$$
  - For $q \in Q'_i$, we have
$$\delta'(q,\alpha) = s_{(i+1) \bmod (m+1)}.$$
  - For $i \in \{0,\dots,m\}$ and $k \in \{0,\dots,2M-1\}$, we have
$$\delta'((f_i,k),\beta) = (f_i, k+1 \bmod 2M).$$
  - We have
$$\delta'(s_0,\beta) = (f_0,0).$$
  - For the remaining states $q \in Q' \setminus (\bigcup_{i=0}^{m} \Gamma_i \cup \{s_0\})$, we have
$$\delta'(q,\beta) = q.$$
- The subset $S \subseteq Q'$ is defined as
$$S = \Big(\bigcup_{i=1}^{m} F_i \cap Q'\Big) \cup \bigcup_{i=0}^{m} \Gamma_i \cup \{s_0\}.$$

It is easy to observe that $\mathcal{D}'$ is strongly connected. Take any $i, j \in \{0,\dots,m\}$. We show how to reach any state $q \in Q'_j$ from a state $p \in Q'_i$. First, we can reach s_j by $\alpha^{(m+1+j-i) \bmod (m+1)}$. For $j \geq 1$, each state $q \in Q'_j \setminus \big(\Gamma_j \setminus \{(f_j,0)\}\big)$ is reachable from s_j, since δ' restricted to Σ acts on Q'_j as δ_j on Q_j (with f_j replaced by $(f_j,0)$) and $\mathcal{D}_j$ is minimal. For $j = 0$, states $(f_0,0)$ and t_0 are reachable from s_0 by the transformations of β and βa respectively, for any $a \in \Sigma$. States $q \in \Gamma_j$ can be reached from $(f_j,0)$ using δ_β.

We will show the following statements:

(1) If S is extensible in $\mathcal{D}'$, then the intersection of the languages L_i is non-empty.
(2) If the intersection of the languages L_i is non-empty, then S is extensible to Q' in $\mathcal{D}'$.

This will prove that the intersection of the languages L_i is non-empty if and only if S is extensible, which is also equivalent to that S is extensible to Q'.

(1): Observe that, for each $i \in \{0,\dots,m\}$, if $(S \cdot w^{-1}) \cap \Gamma_i \neq \emptyset$, then $(S \cdot w^{-1}) \cap \Gamma_i = \Gamma_i$. This follows by induction: the empty word possesses this property; the transformation δ_a of $a \in \Sigma \setminus \{\beta\}$ maps every state from Γ_i to the same state, so it preserves the property; δ_β acts cyclically on Γ_i so also preserves the property.

Suppose that S is extensible by a word w. Notice that, M is an upper bound on the number of states in $Q' \setminus \bigcup_{i=0}^{m} \Gamma_i$ (for $m \geq 2$). We also have $|S| \geq 1 + (m+1) \cdot 2M$. We conclude that $\Gamma_i \subseteq S \cdot w^{-1}$ for each $i \in \{0,\dots,m\}$, since
$$|Q' \setminus \Gamma_i| \leq m \cdot 2M + M \leq (m+1) \cdot 2M < |S|,$$
so $(S \cdot w^{-1}) \cap \Gamma_i \neq \emptyset$ and then our previous observation $\Gamma_i \subseteq S \cdot w^{-1}$.

Now, the extending word w must contain the letter α. For a contradiction, if $w \in (\Sigma' \setminus \{\alpha\})^*$, then if it contains a letter $a \in \Sigma$, then $S \cdot w^{-1}$ does not contain any state from $\Gamma_0 \cup \{t_0\}$, as the only outgoing edges from this subset are labeled by α, $t_0 \notin S$, $\Gamma_0 \cdot \beta^{-1} = \Gamma_0$, and $\Gamma_0 \cdot a^{-1} = \emptyset$. This contradicts the previous paragraph. Also, w cannot be of the form β^k, for $k \in \mathbb{N}$, since $S \cdot \beta^k = S$. Hence, $w = w_p \alpha w_s$, where $w_p \in (\Sigma')^*$ and $w_s \in (\Sigma' \setminus \{\alpha\})$.

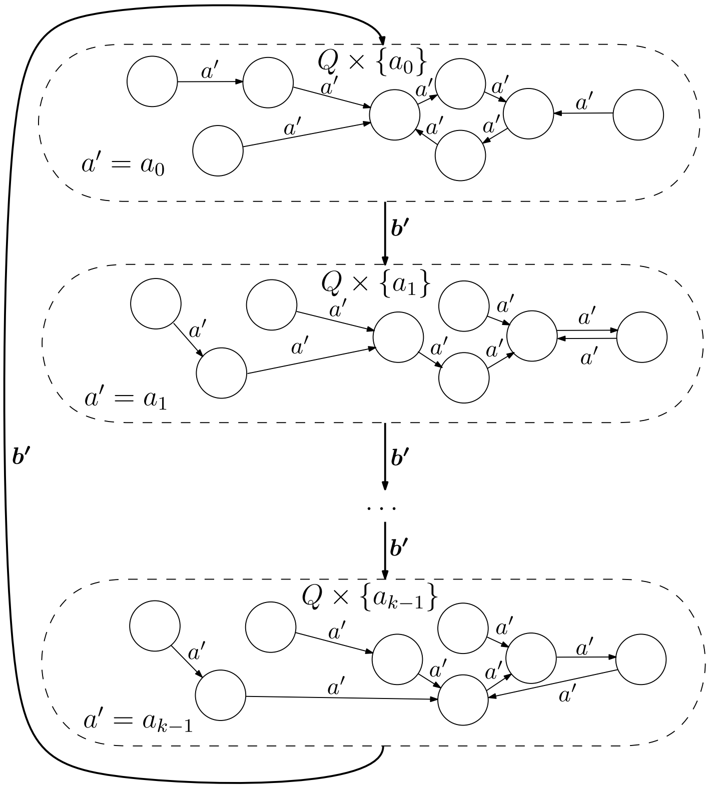

Fig. 3. The binary automaton $\mathcal{A}'$ from the proof of Theorem 4.

Note that if T is a subset of Q' such that $T \cap Q_i' = \emptyset$ for some i, then also $(T \cdot u^{-1}) \cap Q_{i'}' = \emptyset$ for every word u and some i'; because only α maps states Q_i outside Q_i, and it acts cyclically on these sets. Hence, in this case, every preimage of T does not contain some $\Gamma_{i'}$ set. So $\{s_i \mid i \in \{0, \cdots, m\}\} \subseteq S \cdot (w_s)^{-1}$, since in the opposite case $\big(S \cdot (\alpha w_s)^{-1}\big) \cap Q_i' = \emptyset$ for some i.

Let w_s' be the word obtained by removing all β letters from w_s. Note that, for every $i \in \{1, \dots, m\}$ and every suffix u of w_s, we have $(S \cdot u^{-1}) \cap Q_i' = (S \cdot (\beta u)^{-1}) \cap Q_i'$. Hence, $(S \cdot w_s^{-1}) \cap (Q' \setminus Q_0') = S \cdot (w_s')^{-1} \cap (Q' \setminus Q_0')$.

Now, the word w_s' is in Σ^*, and $S \cdot w_s^{-1}$ contains s_i for all $i \in \{1, \dots, m\}$. Hence, the action of w_s' maps s_i to either a state in $F_i \setminus \{f_i\}$ or $(f_i, 0)$, which means that w_s' maps s_i to F_i in $\mathcal{D}_i$. Therefore, w_s' is in the intersection of the languages L_i.

(2): Suppose that the intersection of the languages L_i is non-empty, so there exists a word $w \in \Sigma^*$ such that $s_i \cdot w \in F_i$ for every i. Then we have $S \cdot (\alpha w)^{-1} = Q'$, thus S is extensible to Q'. □

We ensure that both problems remain PSPACE-complete in the case of a binary alphabet, which follows from the following theorem.

Theorem 4. *Given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, we can construct in polynomial time a binary automaton $\mathscr{A}' = (Q', \{a', b'\}, \delta')$ and a subset $S' \subseteq Q'$ such that:*

(1) *$\mathscr{A}$ is strongly connected if and only if $\mathscr{A}'$ is strongly connected;*
(2) *S' is extensible in $\mathscr{A}'$ if and only if S is extensible in $\mathscr{A}$;*
(3) *S' is totally extensible in $\mathscr{A}'$ if and only if S is totally extensible in $\mathscr{A}$.*

Proof. Let $\Sigma = \{a_0, \dots, a_{k-1}\}$. The idea is as follows: We reduce $\mathscr{A}$ to a binary automaton $\mathscr{A}'$ that consists of k copies of $\mathscr{A}$. The first letter a acts in an i-th copy as the letter a_i in $\mathscr{A}$. The second letter b acts cyclically on these copies. Then we define S' to contain states from S in the first copy and all states from the other copies. The construction is shown in Fig. 3.

We construct $\mathscr{A}' = (Q', \{a', b'\}, \delta')$ with $Q' = Q \times \Sigma$ and δ' defined as follows: $\delta'((q, a_i), a') = (\delta(q, a_i), a_i)$, and $\delta'((q, a_i), b') = (q, a_{(i+1) \bmod k})$. Clearly, $\mathscr{A}'$ can be constructed in $\mathcal{O}(nk)$ time, where $k = |\Sigma|$.

(1): Suppose that $\mathscr{A}$ is strongly connected; we will show that $\mathscr{A}'$ is also strongly connected. Let (q_1, a_i) and (q_2, a_j) be any two states of $\mathscr{A}'$. In $\mathscr{A}$, there is a word w such that $q_1 \cdot w = q_2$. Let w' be the word obtained from w by replacing every letter a_h by the word $(b')^h a' (b')^{k-h}$. Note that in $\mathscr{A}'$ we have

$$(p, a_0) \cdot (b')^h a' (b')^{k-h} = (p \cdot a_h, a_0),$$

hence $(q_1, a_0) \cdot w' = (q_1 \cdot w, a_0)$. Then the action of the word $(b')^{k-i} w' (b')^j$ maps (q_1, a_i) to (q_2, a_j).

Conversely, suppose that $\mathscr{A}'$ is strongly connected, so every (q_1, a_i) can be mapped to every (q_2, a_j) by the action of a word w'. Then

$$w' = (b')^{h_1} a' \dots (b')^{h_{m-1}} a' (b')^{h_m},$$

for some $m \geq 1$ and $h_1, \dots, h_m \geq 0$. We construct w of length $m-1$, where the s-th letter is a_r with $r = (i + \Sigma_{j=1}^{s} h_j) \bmod k$. Then w maps q_1 to q_2 in $\mathscr{A}$.

(2) and (3): For $i \in \{0, \dots, k-1\}$ we define $U_i = (Q \times \{\Sigma \setminus \{a_i\}\})$. Observe that for any word $u' \in \{a', b'\}^*$, we have $U_i \cdot (u')^{-1} = U_j$ for some j, which depends on i and the number of letters b' in u'.

We define

$$S' = (S \times \{a_0\}) \cup U_0.$$

Suppose that S is extensible in $\mathscr{A}$ by a word w, and let w' be the word obtained from w as in (1). Then $(w')^{-1}$ maps U_0 to U_0, and $(S \times \{a_0\})$ to $(S \cdot w^{-1}) \times \{a_0\})$. We have:

$$S'(w')^{-1} = ((S \cdot w^{-1}) \times \{a_0\}) \cup U_0,$$

and since $|S \cdot w^{-1}| > |S|$, this means that w' extends S'. By the same argument, if w extends S to Q, then w' extends S' to Q'.

Conversely, suppose that S' is extensible in $\mathscr{A}'$ by a word w', and let w be the word obtained from w' as in (1). Then, for some i, we have

$$S' \cdot (w')^{-1} = ((S \cdot w^{-1}) \times \{a_i\}) \cup U_i,$$

and since $|U_0| = |U_i|$ it must be that $|S \cdot w^{-1}| > |S|$. Also, if $S' \cdot (w')^{-1} = Q'$ then $S \cdot w^{-1} = Q$. □

Now, we consider the subclass of synchronizing automata. We show that synchronizability does not change the complexity of the first problem, whereas the second problem becomes much easier.

Proposition 5. *When the automaton is binary and synchronizing, Problem 1 remains PSPACE-complete.*

Proof. From Theorem 3, Problem 1 is in PSPACE, as the algorithm works the same in the restricted case.

Problem 1 for binary and synchronizing automata is PSPACE-hard, as any general instance with a binary automaton $\mathscr{A} = (Q, \{a, b\}, \delta)$ can be reduced to an equivalent instance with a binary synchronizing automaton $\mathscr{A}'$. For this, we just add a sink state s and a letter which synchronizes Q to s. Additionally, a standard tree-like binarization is suitably used to obtain a binary automaton $\mathscr{A}'$.

Formally, we construct a synchronizing binary automaton $\mathscr{A}'$ from the binary automaton $\mathscr{A}$ as follows. We can assume that $Q = \{q_1, \dots, q_n\}$. Let s be a fresh state. Let $Q' = Q \cup \{q_1^a, \dots, q_n^a\}$. We construct $\mathscr{A}' = (Q' \cup \{s\}, \{a, b\}, \delta')$, where δ' for all i is defined as follows: $\delta'(q_i, a) = q_i^a$, $\delta'(q_i, b) = s$, $\delta'(q_i^a, a) = \delta(q, a)$, and $\delta'(q_i^a, b) = \delta(q, b)$. Then bb is a synchronizing word for $\mathscr{A}'$, and each $S \subseteq Q$ is extensible in $\mathscr{A}'$ if and only if it is extensible in $\mathscr{A}$. □

Theorem 6. *When the automaton is synchronizing, Problem 2 can be solved in $\mathcal{O}(|\Sigma|n)$ time and it is NL-complete.*

Proof. Since $\mathscr{A}$ is synchronizing, Problem 2 reduces to checking whether there is a state $q \in S$ reachable from every state: It is well known that a synchronizing automaton has precisely one strongly connected *sink* component that is reachable from every state. If w is a reset word that synchronizes Q to p, and u is such that $p \cdot u = q$, then wu extends $\{q\}$ to Q. If S does not contain a state from the sink component, then every preimage of S also does not contain these states.

The problem can be solved in $\mathcal{O}(|\Sigma|n)$ time, since the states of the sink component can be determined in linear time by Tarjan's algorithm [45].

It is also easy to see that the problem is in NL: Guess a state $q \in S$ and verify in logarithmic space that it is reachable from every state.

For NL-hardness, we reduce from ST-connectivity: Given a graph $G = (V, E)$ and vertices s, t, check whether there is a path from s to t. We will output a synchronizing automaton $\mathscr{A} = (V, \Sigma, \delta)$ and $S \subseteq Q$ such that S is extensible to Q if and only if there is a path from s to t in G.

First, we compute the maximum out-degree of G, and set $\Sigma = \Sigma' \cup \{\alpha\}$, where $|\Sigma'|$ is equal to the maximum out-degree. We output $\mathscr{A}$ such that for every $q \in V$, every edge $(q, p) \in E$ is colored by a different letter from Σ'. If there is no outgoing edge from q, then we set the transitions of all letters from Σ' to be loops. If the out-degree is smaller than $|\Sigma'|$, then we simply repeat the transition of the last letter. Next, we define $\delta(q, \alpha) = s$ for every $q \in V$. Finally, let $S = \{t\}$. The reduction uses logarithmic space since it requires only counting and enumerating through V and Σ'. The produced automaton $\mathscr{A}$ is synchronizing just by α.

Suppose that there is a path from s to t. Then there is a word w such that $\delta(s, w) = t$, and so $\{t\} \cdot (\alpha w)^{-1} = Q$.

Suppose that $\{t\}$ is extensible to Q by some word w. Let w' be the longest suffix of w that does not contain α. Since α^{-1} results in $\emptyset$ for any subset not containing s, it must be that $s \in \{t\}(w')^{-1}$. Hence $\delta(s, w') = t$, and the path labeled by w' is the path from s to t in G. □

Note that in the case of strongly connected synchronizing automaton, both problems have a trivial solution, since every non-empty proper subset of Q is totally extensible (by a suitable reset word); thus they can be solved in constant time, assuming that we can check the size of the given subset and the number of states in constant time.

2.2. Bounded word length

We turn our attention to the variants in which an upper bound on the length of word w is also given.

Problem 3 *(Extensible subset by short word).* Given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer ℓ given in binary representation, is S extensible by a word of length at most ℓ?

Problem 4 *(Totally extensible subset by short word).* Given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer ℓ given in binary representation, is S totally extensible by a word of length at most ℓ?

Obviously, these problems remain PSPACE-complete (also when the automaton is strongly connected and binary), as we can set $\ell = 2^n$, which bounds the number of different subsets of Q. In this case, both the problems are reduced respectively to Problem 1 and Problem 2.

When the automaton is synchronizing, Problem 4 is NP-complete, which will be shown in Corollary 14. Of course, Problem 3 remains PSPACE-complete for a synchronizing automaton by the same argument as in the general case.

3. Extending small subsets

The complexity of the extending problems is caused by an unbounded size of the given subset. Note that in the proof of PSPACE-hardness in Theorem 3 the used subsets and simultaneously their complements may grow with an instance of the reduced problem, and it is known that the problem of the emptiness of intersection can be solved in polynomial time if the number of given DFAs is fixed. Here, we study the computational complexity of the extending problems when the size of the subset is not larger than a fixed k.

3.1. Unbounded word length

Problem 5 *(Extensible small subset).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|S| \le k$, is S extensible?

Proposition 7. *Problem 5 can be solved in $\mathcal{O}(|\Sigma|n^k)$ time.*

Proof. We build the k-subsets automaton $\mathscr{A}^{\le k} = (Q^{\le k}, \Sigma, \delta^{\le k}, S_0, F)$, where $Q^{\le k} = \{A \subseteq Q : |A| \le k\}$ and $\delta^{\le k}$ is naturally defined by the image of δ on a subset. Let the set of initial states be $I = \{A \in Q^{\le k} : |A \cdot a^{-1}| > |S| \text{ for some } a \in \Sigma\}$, and the set of final states be the set of all subsets of S. A final state can be reached from an initial state if and only if S is extensible in $\mathscr{A}$. We can simply check this condition by a BFS algorithm.

Note that we can compute whether a subset A of size at most k is in I in $\mathcal{O}(|\Sigma|)$, by summing the sizes $|q \cdot a^{-1}|$ for all $q \in A$, where $|q \cdot a^{-1}|$ are computed during a preprocessing, which takes $O(n)$ time for a single $a \in \Sigma$. Also, for a given subset A of size at most k, we can compute $T \cdot a$ in constant time (which depends only k). Hence, the BFS works in linear time in the size of $\mathscr{A}^{\le k}$, so in $O(|\Sigma|n^k)$ time. □

Problem 6 *(Totally extensible small subset).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|S| \le k$, is S totally extensible?

For $k = 1$, Problem 2 is equivalent to checking if the automaton is synchronizing to the given state, thus can be solved in $\mathcal{O}(|\Sigma|n^2)$ time. For larger k we have the following:

Proposition 8. *Problem 6 can be solved in $\mathcal{O}(|\Sigma|n^k + n^3)$ time.*

Proof. Let u be a word of the minimal rank in $\mathscr{A}$. We can find such a word and compute the image $Q \cdot u$ in $\mathcal{O}(n^3 + |\Sigma|n^2)$ time, using the well-known algorithm [19, Algorithm 1] generalized to non-synchronizing automata. The algorithm just stops when there are no more compressible pairs of states contained in the current subset, and since the subset cannot be further compressed, the found word has the minimal rank.

For each $w \in \Sigma^*$ we have $S \cdot w^{-1} = Q$ if and only if $Q \cdot w \subseteq S$. We can meet the required condition for w if and only if $(Q \cdot u) \cdot w \subseteq S$. Surely $|(Q \cdot u) \cdot w| = |Q \cdot u|$. The desired word does not exist if the minimal rank is larger than $|S| = k$. Otherwise, we can build the subset automaton $\mathscr{A}^{\le |Q \cdot u|}$ (similarly as in the proof of Proposition 7). The initial subset is $Q \cdot u$. If some subset of S is reachable by a word w, then the word uw totally extends S in $\mathscr{A}$. Otherwise, S is not totally extensible. The reachability can be checked in at most $\mathcal{O}(|\Sigma|n^k)$ time. However, if the rank r of u is less than k, the algorithm takes only $\mathcal{O}(|\Sigma|n^r)$ time. □

3.2. Bounded word length

We also have the two variants of the above problems when an upper bound on the length of the word is additionally given.

Problem 7 *(Extensible small subset by short word).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|S| \le k$, and an integer ℓ given in binary representation, is S extensible by a word of length at most ℓ?

Problem 7 can be solved by the same algorithm in a Proposition 7, since the procedure can find a shortest extending word.

Problem 8 *(Totally extensible small subset by short word).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|S| \le k$, and an integer ℓ given in binary representation, is S totally extensible by a word of length at most ℓ?

Proposition 9. *For every k, Problem 8 is NP-complete, even if the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. The problem is in NP, as the shortest extending words have length at most $\mathcal{O}(n^3 + n^k)$ (since words of this length can be found by the procedure from Proposition 8).

When we choose S of size 1, the problem is equivalent to finding a reset word that maps every state to the state in S. In [37] it has been shown that for Eulerian automata that are simultaneously strongly connected, synchronizing, and binary, deciding whether there is a reset word of length at most ℓ is NP-complete. Moreover, in this construction, if there exists a reset word of this length, then it maps every state to one particular state s_2 (see [37, Lemma 2.4]). Therefore, we can set $S = \{s_2\}$, and thus Problem 8 is NP-complete. □

4. Extending large subsets

In this section, we consider the case where the subset S contains all except at most a fixed number of states k.

4.1. Unbounded word length

Problem 9 *(Extensible large subset).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, is S extensible?

Problem 10 *(Totally extensible large subset).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, is S totally extensible?

Problem 10 is equivalent to deciding the existence of an avoiding word for a subset S of size $\le k$. Note that Problem 9 and Problem 10 are equivalent for $k = 1$, when they become the problem of avoiding a single given state. Its properties will also turn out to be different than in the case of $k \ge 2$. We give a special attention to this problem, defined as follows, and study it separately.

Problem 11 *(Avoidable state).* Given $\mathscr{A} = (Q, \Sigma, \delta)$ and a state $q \in Q$, is $\{q\}$ avoidable?

The following result may be a bit surprising, in view of that it is the only case where a general problem (i.e., Problems 1 and 2) remains equally hard when the subset size is additionally bounded. We show that Problem 9 is PSPACE-complete for all $k \ge 2$, although the question about its complexity remains open for the class of strongly connected automata.

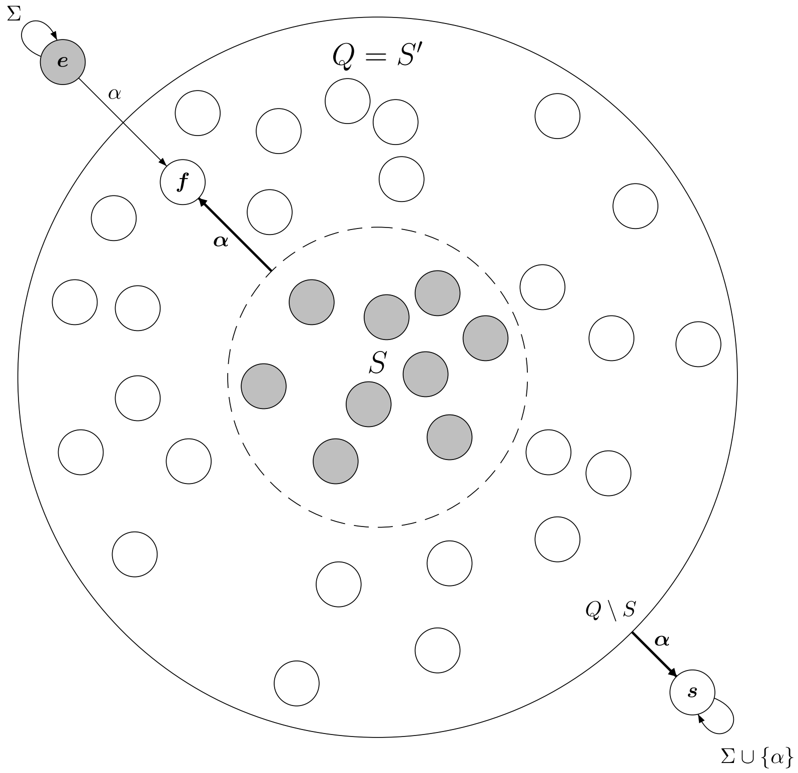

Fig. 4. The constructed automaton $\mathscr{A}'$: States in $Q = S'$ have the transitions on Σ as in $\mathscr{A}$. The preimage of $S' = Q$ by α is marked by gray nodes and reflects the initial situation after applying for any subset containing f and not containing s.

Theorem 10. *Problem 9 is PSPACE-complete for every fixed $k \geq 2$, even if the given automaton is synchronizing and binary.*

Proof. Problem 9 is in PSPACE as a special case of Problem 1, which is PSPACE-complete (Theorem 3).

Now, we show a reduction from Problem 2. The idea is as follows. We construct an automaton $\mathscr{A}'$ from the automaton $\mathscr{A} = (Q, \Sigma, \delta)$ given for Problem 2. We add two new states, e and s, and let the initial set S' contain all the original states of $\mathscr{A}$. State s is a sink state ensuring that the automaton is synchronizing; it cannot be reached from S' by inverse transitions. Hence, to extend S', one needs to get e, which is doable only by a new special letter α. This letter has the transition that shrinks all states Q to the initial subset S for the totally extensible problem. This is done through an arbitrary selected state $f \in Q$. Then we can reach $Q \cup \{e\}$ only by a totally extending word for $\mathscr{A}$. The overall construction is presented in Fig. 4.

Let $\mathscr{A} = (Q, \Sigma, \delta)$ and $S \subseteq Q$ be an instance of Problem 2. We construct an automaton $\mathscr{A}' = (Q' = Q \cup \{e, s\}, \Sigma' = \Sigma \cup \{\alpha\}, \delta')$, where e, s are fresh states and α is a fresh letter. Let f be an arbitrary state from $Q \setminus S$ (if $S = Q$ then the problem is trivial). We define δ' as follows:

1. $\delta'(q, a) = \delta(q, a)$ for $q \in Q$, $a \in \Sigma$;
2. $\delta'(q, a) = q$ for $q \in \{e, s\}$, $a \in \Sigma$;
3. $\delta'(q, \alpha) = f$ for $q \in S \cup \{e\}$;
4. $\delta'(q, \alpha) = s$ for $q \in (Q \cup \{s\}) \setminus S$.

We define $S' = Q$. Note that $|Q' \setminus S'| = 2$, and hence automaton $\mathscr{A}'$ with S' is an instance of Problem 9 for $k = 2$. We will show that S' is extensible in $\mathscr{A}'$ if and only if S is totally extensible in $\mathscr{A}$.

If S is totally extensible in $\mathscr{A}$ by a word $w \in \Sigma^*$, we have $S' \cdot (w\alpha)^{-1} = Q \setminus \{s\}$, which means that S' is extensible in $\mathscr{A}'$.

Conversely, if S' is extensible in $\mathscr{A}'$, then there is some extending word of the form $w\alpha$ for some $w \in \Sigma^*$, because $S' \cdot a^{-1} = S'$ for $a \in \Sigma$, $(Q' \setminus \{s\}) \cdot \alpha^{-1} \subseteq S' \cdot \alpha^{-1}$, and each reachable set (as a preimage) is a subset of $Q' \setminus \{s\}$. We know that $S' \cdot (w\alpha)^{-1} = (S \cup \{e\}) \cdot w^{-1} = (S \cdot w^{-1}) \cup \{e\}$. From the fact that $|S' \cdot (w\alpha)^{-1}| > |S'|$, we conclude that $S \cdot w^{-1} = Q$, so S is totally extensible in $\mathscr{A}$.

Note that $\mathscr{A}'$ is synchronizing, since $Q' \cdot \alpha^2 = \{f, s\} \cdot \alpha = \{s\}$.

Now, we show that we can reduce the alphabet to two letters. Consider the application of the Theorem 4 to Problem 9. Note that the reduction in the proof keeps the size of complement set the same (i.e. $|Q' \setminus S'| = |Q'' \setminus S''|$, where Q'' and S'' are the set and the subset of states in the constructed binary automaton), so we can apply it.

Furthermore, we identify all the states of the form (s, a) for $a \in \Sigma$ in the obtained binary automaton to one sink state s''. In this way, we get a synchronizing binary automaton (since $\mathscr{A}'$ is synchronizing). The extending words remain the same, since the identified state s'' is not reversely reachable from S'', and s'' is not contained in the subset S''.

Finally, we conclude that the proof generalizes to the case of any $k \geq 2$ since we can add an arbitrary number of states with the same transitions as e. □

Now, we focus on totally extending words for large subsets, which we study in terms of avoiding small subsets. First we provide a complete characterization of single states that are avoidable:

Theorem 11. *Let $\mathscr{A} = (Q, \Sigma, \delta)$ be a strongly connected automaton. For every $q \in Q$, state q is avoidable if and only if there exists $p \in Q \setminus \{q\}$ and $w \in \Sigma^*$ such that $q \cdot w = p \cdot w$.*

Proof. First, for a given $q \in Q$, let $p \in Q \setminus \{q\}$ and $w \in \Sigma^*$ be such that $q \cdot w = p \cdot w$. Since the automaton is strongly connected, there is a word w' such that $(p \cdot w) \cdot w' = (q \cdot w) \cdot w' = p$. For each subset $S \subseteq Q$ such that $p \in S$ we have $p \in S \cdot ww'$. Moreover, if $q \in S$ then $|S \cdot ww'| < |S|$, because $\{q, p\} \cdot ww' = \{p\}$. If q is not avoidable, then all subsets $Q \cdot (ww'), Q \cdot (ww')^2, \ldots$ contain q and they form an infinite sequence of subsets of decreasing cardinality, which is a contradiction.

Now, consider the other direction. Suppose for a contradiction that a state $q \in Q$ is avoidable, but there is no state $p \in Q \setminus \{q\}$ such that $\{q, p\}$ can be compressed. Let u be a word of the minimal rank in $\mathscr{A}$, and v be a word that avoids q. Then $w = uv$ has the same rank and also avoids q. Let $\sim$ be the equivalence relation on Q defined with a word w as follows:

$$p_1 \sim p_2 \iff p_1 \cdot w = p_2 \cdot w.$$

The equivalence class $[p]_\sim$ for $p \in Q$ is $(p \cdot w) \cdot w^{-1}$. There are $|Q/{\sim}| = |Q \cdot w|$ equivalence classes and one of them is $\{q\}$, since q does not belong to a compressible pair of states. For every state $p \in Q$, we know that $|(Q \cdot w) \cap [p]_\sim| \leq 1$, because $[p]_\sim$ is compressed by w to a singleton and $Q \cdot w$ cannot be compressed by any word. Note that every state $r \in Q \cdot w$ belongs to some class $[p]_\sim$. From the equality $|Q / \sim| = |Q \cdot w|$ we conclude that for every class $[p]_\sim$ there is a state $r \in (Q \cdot w) \cap [p]_\sim$, thus $|(Q \cdot w) \cap [p]_\sim| = 1$. In particular, $1 = |(Q \cdot w) \cap [q]_\sim| = |(Q \cdot w) \cap \{q\}|$. This contradicts that w avoids q. □

Note that if $\mathscr{A}$ is not strongly connected, then every state from a strongly connected component that is not a sink can be avoided. If a state belongs to a sink component, then we can consider the sub-automaton of this sink component, and by Theorem 11 we know that given $q \in Q$, it is sufficient to check whether q belongs to a compressible pair of states. Hence, Problem 11 can be solved using the well-known algorithm (stage 1 in the proof of [19, Theorem 5]) computing the pair automaton and performing a breadth-first search with inverse edges on the pairs of states. It works in $\mathcal{O}(|\Sigma|n^2)$ time and $\mathcal{O}(n^2 + |\Sigma|n)$ space.

We note that in a synchronizing automaton all states are avoidable except a *sink state*, which is a state q such that $q \cdot a = q$ for all $a \in \Sigma$. We can check this condition and hence verify if a state is avoidable in a synchronizing automaton in $\mathcal{O}(|\Sigma|)$ time.

The above algorithm does not find an avoiding word but checks avoidability indirectly. For larger subsets than singletons, we construct another algorithm finding a word avoiding the subset, which also generalizes the idea from Theorem 11. From the following theorem, we obtain that Problem 10 for a constant $k \geq 2$ can be solved in polynomial time.

Theorem 12. *Let $\mathscr{A} = (Q, \Sigma, \delta)$, let r be the minimum rank in $\mathscr{A}$ over all words, and let $S \subseteq Q$ be a subset of size $\leq k$. We can find a word w such that $(Q \cdot w) \cap S = \emptyset$ or verify that it does not exist in $\mathcal{O}(|\Sigma|(n^{\min(r,k)} + n^2) + n^3)$ time and $\mathcal{O}(n^{\min(r,k)} + n^2 + |\Sigma|n)$ space. Moreover the length of w is bounded by $\mathcal{O}(n^{\min(r,k)} + n^3)$).*

Proof. Similarly to the proof of Theorem 11, let u be a word of the minimal rank r in $\mathscr{A}$ and let $\sim$ be the equivalence relation on Q defined by word u as follows:

$$p_1 \sim p_2 \iff p_1 \cdot u = p_2 \cdot u.$$

The equivalence class $[p]_\sim$ for $p \in Q$ is the set $(p \cdot u) \cdot u^{-1}$. There are $|Q/{\sim}| = |Q \cdot u|$ equivalence classes.

First, we prove a key observation that the image of each word starting with prefix u has exactly one state in each equivalence class of $\sim$ relation. Let $w = uw'$. Then the word w has rank r and its image is not compressible. For every state $p \in Q$, we know that $|(Q \cdot w) \cap [p]_\sim| \leq 1$, because $[p]_\sim$ is compressed by u to a singleton and $Q \cdot w$ cannot be compressed

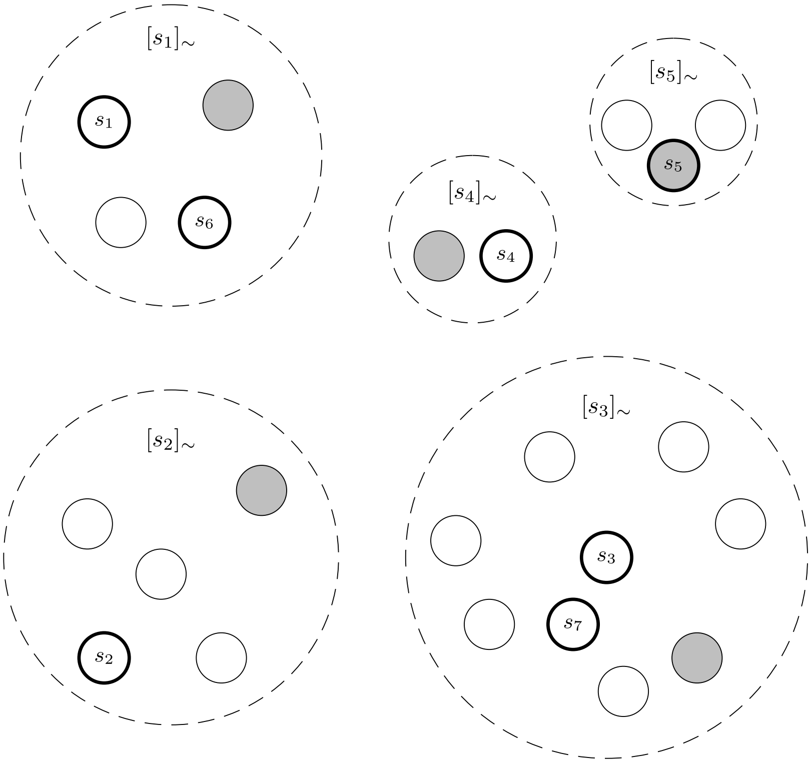

Fig. 5. The states of an automaton divided by $\sim$. The states $s_i \in S$ are marked by bold border and the states $q_{[s_i]_\sim}$ in the image $Q \cdot uw'$ are filled. Every class has exactly one state in the image but can contain more than one state from S. If for each class this state is not in S, then S is avoided. This is not the case in this example, because $s_5 \in Q \cdot uw'$.

by any word. Note that every state $q \in Q \cdot w$ belongs to some class $[p]_\sim$. From the equality $|Q/\sim| = |Q \cdot u| = |Q \cdot w|$ we conclude that for every class $[p]_\sim$ there is an unique state $q_{[p]_\sim} \in (Q \cdot w) \cap [p]_\sim$. This proves the mentioned observation.

Now, we are going to show the following characterization: S is avoidable if and only if there exist a subset $Q' \subseteq Q \cdot u$ of size $|S/\sim|$ and a word w' such that $(Q' \cdot w') \cap ([s]_\sim \setminus S) \neq \emptyset$ for each $s \in S$. The idea of the characterization is illustrated in Fig. 5.

Suppose that S is avoidable, and let w' be an avoiding word for S. Then the word $w = uw'$ also avoids S. Observe that $Q \cdot w$ has an unique state $q_{[p]_\sim} \in (Q \cdot w) \cap [p]_\sim$ for each class $[p]_\sim$. Then for every state $s \in S$, we have $q_{[s]_\sim} \in [s]_\sim \setminus S$, because w avoids S and $q_{[s]_\sim} \in Q \cdot w$. Notice that $[s]_\sim \cap S$ can contain more than one state, so the set $\{q_{[s]_\sim} \mid s \in S\}$ has size $|S/\sim|$, which is not always equal to $|S|$. Therefore, there exists a subset $Q' \subseteq Q \cdot u$ of size $|S/\sim|$ such that $Q' \cdot w' = \{q_{[s]_\sim} \mid s \in S\}$. Now, we know that for every $s \in S$ we have $q_{[s]_\sim} \in Q' \cdot w'$ and $q_{[s]_\sim} \in [s]_\sim \setminus S$. We conclude that, if S is avoidable, then there exist a subset $Q' \subseteq Q \cdot u$ of size $|S/\sim|$ and a word w' such that $(Q' \cdot w') \cap ([s]_\sim \setminus S) \neq \emptyset$ for every $s \in S$.

Conversely, suppose that there is a subset $Q' \subseteq Q \cdot u$ of size $|S/\sim|$ and a word w' such that $(Q' \cdot w') \cap ([s]_\sim \setminus S) \neq \emptyset$ for every $s \in S$. Since in the image $Q \cdot uw'$ there is exactly one state in each equivalence class, we have $((Q \cdot u) \setminus Q') \cdot w' \subseteq Q \setminus \bigcup_{s \in S}([s]_\sim) \subseteq Q \setminus S$, and by the assumption, $(Q' \cdot w') \cap S = \emptyset$. Therefore, we get that uw' is an avoiding word for S.

This characterization gives us Algorithm 1 to find w or verify that S cannot be avoided.

Algorithm 1 Avoiding a subset.

```
Require: Automaton $\mathscr{A}(Q, \Sigma, \delta)$ and a subset $S \subseteq Q$.
Find a word $u$ of the minimal rank.
Compute $|S/\sim|$.
for all $Q' \subseteq Q \cdot u$ of size $|S/\sim|$ do
    if there is a word $w'$ such that $(Q' \cdot w') \cap ([s]_\sim \setminus S) \neq \emptyset$ for each $s \in S$ then
        return $uw'$.
    end if
end for
return "$S$ is unavoidable".
```

Algorithm 1 first finds a word u of the minimal rank. This can be done by in $\mathcal{O}(n^3 + |\Sigma|n^2)$ time and $\mathcal{O}(n^2 + |\Sigma|n)$ space by the well-known algorithm [19, Algorithm 1] generalized to non-synchronizing automata (cf. the proof of Proposition 8. For every subset $Q' \subseteq Q \cdot u$ of size $z = |S/\sim|$ the algorithm checks whether there is a word w' mapping Q' to avoid S, but using its $\sim$-classes. This can be done by constructing the automaton $\mathscr{A}^z(Q^z, \Sigma, \delta^z)$, where δ^z is δ naturally extended to z-tuples of states, and checking whether there is a path from Q' to a subset containing a state from each class $[s]_\sim$ but avoiding the states from S. Note that since Q' cannot be compressed, every reachable subset from Q' has also size $|Q'|$. The number of states in this automaton is $\binom{n}{z} \in \mathcal{O}(n^z)$. Also, note that we have to visit every z-tuple only once during a run of the algorithm, and we can store it in $\mathcal{O}(n^z + |\Sigma|n)$ space. Therefore, the algorithm works in $\mathcal{O}(n^3 + |\Sigma|(n^2 + n^z))$ time and $\mathcal{O}(n^2 + n^z + |\Sigma|n)$ space.

The length of u is bounded by $\mathcal{O}(n^3)$, and the length of w' is at most $\mathcal{O}(n^z)$. Note that $z = |S/{\sim}| \le \min(r, |S|)$, where r is the minimal rank in the automaton. □

4.2. Bounded word length

We now turn our attention to the variants of Problem 9, Problem 10, and Problem 11 where an upper bound on the length of the word is additionally given.

Problem 12 *(Extensible large subset by short word).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, and an integer ℓ given in binary representation, is S extensible by a word of length at most ℓ?

Problem 13 *(Totally extensible large subset by short word).* For a fixed $k \in \mathbb{N} \setminus \{0\}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, and an integer ℓ given in binary representation, is S totally extensible by a word of length at most ℓ?

As before, both problems for $k = 1$ are equivalent to the following:

Problem 14 *(Avoidable state by short word).* Given $\mathscr{A} = (Q, \Sigma, \delta)$, a state $q \in Q$, and an integer ℓ given in binary representation, is $\{q\}$ avoidable by a word of length at most ℓ?

Problem 12 for $k \ge 2$ obviously remains PSPACE-complete. By the following theorem, we show that Problem 14 is NP-complete, which then implies NP-completeness of Problem 13 for every $k \ge 1$ (by Corollary 14).

Theorem 13. *Problem 14 is NP-complete, even if the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. The problem is in NP, because we can non-deterministically guess a word w as a certificate, and verify $q \notin Q \cdot w$ in $\mathcal{O}(|\Sigma|n)$ time. If the state q is avoidable, then the length of the shortest avoiding words is at most $\mathcal{O}(n^2)$ [18]. Then we can guess an avoiding word w of at most quadratic length and compute $Q \cdot w$ in $\mathcal{O}(n^3)$ time.

In order to prove NP-hardness, we present a polynomial-time reduction from the problem of determining the reset threshold in a specific subclass of automata, which is known to be NP-complete [19, Theorem 8]. The reduction has two steps. First, we construct a strongly connected synchronizing ternary automaton $\mathscr{A}'$ for which deciding about the length of an avoiding word is equivalent to determining the existence of a bounded length reset word in the original automaton. Then, based on the ideas from [46], we turn the automaton into a binary automaton $\mathscr{A}$, which still has the desired properties.

Let us have an instance of this problem from the Eppstein's proof of [19, Theorem 8]. Namely, for a given synchronizing automaton $\mathscr{B} = (Q_\mathscr{B}, \{\alpha_0, \alpha_1\}, \delta_\mathscr{B})$ and an integer $m > 0$, we are to decide whether there is a reset word w of length at most m. We do not want to reproduce here the whole construction from the Eppstein proof but we need some ingredients of it. Specifically, $\mathscr{B}$ is an automaton with a sink state $z \in Q_\mathscr{B}$, and there are two subsets $S = \{s_1, \dots, s_d\}$ and $F \subseteq Q_\mathscr{B}$ with the following properties:

1. Each state $q \in Q_\mathscr{B} \setminus S$ is reachable from a state $s \in S$ through a (directed) path in the underlying digraph of $\mathscr{B}$.
2. For each state $s \in S$ and each word w of length m, we have $\delta_\mathscr{B}(s, w) \in F \cup \{z\}$.
3. For each $f \in F$ we have $\delta_\mathscr{B}(f, \alpha_0) = \delta_\mathscr{B}(f, \alpha_1) = z$.
4. For each state $s \in S$ and a non-empty word $w \in \{\alpha_0, \alpha_1\}^{<m}$, we have $\delta_\mathscr{B}(s, w) \notin (F \cup S)$.

In particular, it follows that each word of length $m + 1$ is reset. Deciding whether $\mathscr{B}$ has a reset word of length m is NP-hard.

We transform the automaton $\mathscr{B}$ into $\mathscr{A}'$ as follows. First, we add the subset $R = \{r_0, r_1, \dots, r_m\}$ of states to provide that z is not avoidable by words of length less than $m + 1$. The transitions of both letters are $\delta_{\mathscr{A}'}(r_i, \alpha_0) = \delta_{\mathscr{A}'}(r_i, \alpha_1) = r_{i+1}$ for $i = 0, \dots, m - 1$, and $\delta_{\mathscr{A}'}(r_m, \alpha_0) = \delta_{\mathscr{A}'}(r_m, \alpha_1) = z$.

Secondly, we add a set of states $S' = \{s'_1, \dots, s'_d\}$ of size $d = |S|$ and a letter α_2 to make the automaton strongly connected. Letters α_0 and α_1 map S' to the corresponding states from S, that is, $\delta_{\mathscr{A}'}(s'_i, \alpha_0) = \delta_{\mathscr{A}'}(s'_i, \alpha_1) = s_i \in S$. Letter α_2 connects states $r_0, s'_1, s'_2 \dots, s'_d$ into one cycle, i.e.

$$\delta_{\mathscr{A}'}(r_0, \alpha_2) = s'_1, \quad \delta_{\mathscr{A}'}(s'_1, \alpha_2) = s'_2, \quad \dots, \quad \delta_{\mathscr{A}'}(s'_{d-1}, \alpha_2) = s'_d, \quad \delta_{\mathscr{A}'}(s'_d, \alpha_2) = r_0.$$

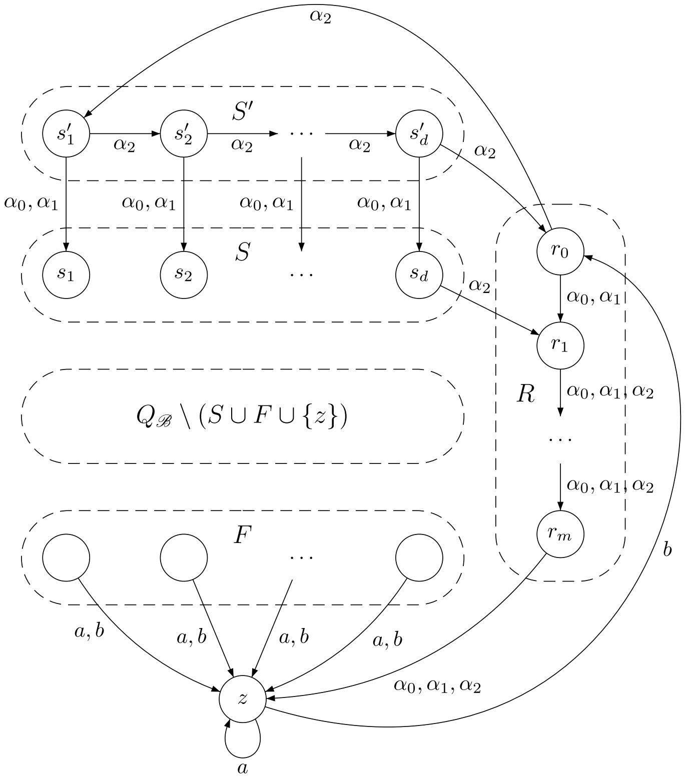

Fig. 6. The automaton $\mathscr{A}$ obtained from $\mathscr{A}'$ in the proof of Theorem 13. Here every state q represents $\phi(q)$, and we have α_0: aa, ab, α_1: ba, and α_2: bb.

We also set $\delta_{\mathscr{A}'}(s_d, \alpha_2) = r_1$, $\delta_{\mathscr{A}'}(z, \alpha_2) = r_0$, and all the other transitions of α_2 we define equal to the transitions of α_0.

Finally, we transform $\mathscr{A}'$ to the final automaton $\mathscr{A} = (Q, \{a, b\}, \delta)$. We encode letters $\alpha_0, \alpha_1, \alpha_2$ by 2-letter words over $\{a, b\}$ alike it was done in [46]. Namely, for each state $q \in Q_{\mathscr{A}'} \setminus (F \cup \{z\})$, we add two new states q^a, q^b and define their transitions as follows:

$$\delta(q, a) = q^a, \quad \delta(q^a, a) = \delta(q^a, b) = \delta_{\mathscr{A}'}(q, \alpha_0),$$
$$\delta(q, b) = q^b, \quad \delta(q^b, a) = \delta_{\mathscr{A}'}(q, \alpha_1), \quad \delta(q^b, b) = \delta_{\mathscr{A}'}(q, \alpha_2).$$

Then, aa, ab correspond to applying letter α_0, ba corresponds to applying letter α_1, and bb corresponds to applying letter α_2. Denote this encoding function by ϕ, i.e. $\phi(\alpha_0) = aa$, $\phi(\alpha_1) = ba$, and $\phi(\alpha_2) = bb$. We also extend ϕ to words over $\{\alpha_0, \alpha_1, \alpha_2\}^*$ as usual. For simplicity, we denote also $\phi(q) = \{q, q^a, q^b\}$, and extend to subsets of $Q_{\mathscr{A}'}$ as usual.

It remains to define the transitions for $F \cup \{z\}$. We set $\delta(z, a) = z$, $\delta(z, b) = r_0$, and $\delta(f, a) = \delta(f, b) = z$ for each $f \in F$. Automaton $\mathscr{A}$ is shown in Fig. 6.

Observe that $\mathscr{A}'$ is strongly connected: z is reachable from each state, from z we can reach r_0 by α_2, from r_0 we can reach every state from S' by applying a power of letter α_2, and we can reach every state of S from the corresponding state from S'. Then every state from $Q_{\mathscr{B}}$ is reachable from a state from S by Property 1. It follows that $\mathscr{A}$ is also strongly connected, since for every $q \in Q_{\mathscr{A}'}$, every state from $\phi(q)$ is reachable from q, and since for $F \cup \{z\}$ the outgoing edges correspond to those in $\mathcal{A}$.

Observe that $\mathscr{A}$ is synchronizing: We claim that a^{4m+6} is a reset word for $\mathscr{A}$. Indeed, aa does not map any state into $\phi(S')$. Every word of length $m + 1$ is reset for $\mathscr{B}$ and synchronizes to z, in particular, α_0^{m+1}. Since $\phi(\alpha_0^{m+1}) = a^{2m+2}$ does not contain bbb, state z cannot go to S' by a factor of this word. Hence, we have

$$\delta(Q, a^{2m+4}) \subseteq \{z\} \cup \phi(R).$$

Then, finally, $a^{2(m+1)}$ compresses $\{z\} \cup \phi(R)$ to z.

Now, we claim that the original problem of checking whether $\mathscr{B}$ has a reset word of length m is equivalent to determining whether z can be avoided in $\mathscr{A}$ by a word of length at most $2m + 3$.

Suppose that $\mathscr{B}$ has a reset word w of length m, and consider $u = \phi(\alpha_0 w)b$. Note that $\phi(\alpha_0) = aa$ does not map any state into $\phi(S')$ nor into $\phi(r_0)$. Hence, we have

$$\delta(Q, \phi(\alpha_0)) \subseteq \phi(Q_{\mathscr{B}}) \cup \phi(R \setminus \{r_0\}).$$

Due to the definition of ϕ, factor bbb cannot appear in the image of words from $\{\alpha_0, \alpha_1\}^*$ by ϕ. Henceforth, z cannot go to S' by a factor of $\phi(w)$. Since $|\phi(w)| = 2m$ and to map z into $\phi(r_m)$ we require a word of length $2m+1$, the factors of $\phi(w)$ do not map z into $\phi(r_m)$. Since also w is a reset word for $\mathscr{B}$ that maps every state from $Q_{\mathscr{B}}$ to z, we have

$$\delta(\phi(Q_{\mathscr{B}}), \phi(w)) \subseteq \{z\} \cup \phi(R \setminus \{r_m\}).$$

By the definition of the transitions on $R \cup \{z\}$ (only $\phi(\alpha_2)$ maps r_0 outside), and since $|\phi(w)| = 2m$, we also have

$$\delta(\phi(R \setminus \{r_0\}), \phi(w)) \subseteq \{z\} \cup \phi(R \setminus \{r_m\}).$$

Finally, we get that $\delta(\{z\} \cup \phi(R \setminus \{r_m\}), b) \subset R$, thus u avoids z.

Now, we prove the opposite direction. Suppose that state z can be avoided by a word u of length at most $2m+3$. Then, by the definition of the transitions on R, $|u| = 2m+3$ because $z \in \delta(R, w)$ for each w of length at most $2(m+1)$. Let $u = u'u''u'''$ with $|u'| = 2$, $|u''| = 2m$, and $|u'''| = 1$.

For words $w \in \{a, b\}^*$ of even length, we denote by $\tilde{\phi}^{-1}(w)$ the inverse image of encoding ϕ with respect to the definition on $\mathscr{A}'$, that is, $\tilde{\phi}^{-1}(aa) = \tilde{\phi}^{-1}(ab) = \alpha_0$, $\tilde{\phi}^{-1}(ba) = \alpha_1$, $\tilde{\phi}^{-1}(bb) = \alpha_2$, which is extended to words of even length by concatenation.

First notice that $\tilde{\phi}^{-1}(u') \neq \alpha_2$. Otherwise $\{z, r_0, r_1, r_2, \dots, r_m\} \subseteq \delta(S' \cup R \cup \{z\}, \tilde{\phi}^{-1}(u'))$ whence by the definition of R the word $u''u'''$ of length $2m+1$ cannot avoid z. Therefore $\tilde{\phi}^{-1}(u') \neq \alpha_2$ and $S \subseteq \delta(S \cup S', u')$.

If α_2 is the second letter of $\tilde{\phi}^{-1}(u)$, then s_d goes to r_1 and we get $\{r_1, r_2, \dots, r_m, z\}$ in the image of the prefix of u of length 4. Then, due to the definition of R, no word of length at most $2m$ can avoid z.

Hence, the first two letters of $\tilde{\phi}^{-1}(u)$ are either α_0 or α_1.

By Property 2 of $\mathscr{B}$, every zero-one word of length m maps $s \in S$ into $\{z\} \cup F$. Since the letter α_2 acts like α_0 on $Q_{\mathscr{B}} \setminus S$ in $\mathscr{A}'$ and $\tilde{\phi}^{-1}(u'')$ starts with α_0 or α_1, u'' maps S into $\{z\} \cup F$. If u'' maps some state to F, then by Property 3 u cannot avoid z. Hence, $\tilde{\phi}^{-1}(u'')$ with all α_2 replaced with α_0 must be a reset word for $\mathscr{B}$. □

By a corollary from Theorem 13 and Theorem 12, we complete our results about extending subsets.

Corollary 14. *Problem 13 is NP-complete, Problem 4 is NP-complete when the automaton is synchronizing, and Problem 12 is NP-complete when the automaton is strongly connected and synchronizing. They remain NP-complete when the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. NP-hardness for all the problems follows from Theorem 13, since we can set $S = Q \setminus \{q\}$.

Problem 13 is solvable in NP as follows. By Theorem 12 if there exists a totally extending word, then there exists such a word of polynomial length. Thus we first run this algorithm, and if there is no totally extending word then we answer negatively. Otherwise, we know that the length of the shortest totally extending words is polynomially bounded, so we can nondeterministically guess such a word of length at most ℓ and verify whether it is totally extending.

Similarly, Problem 4 is solvable in NP for synchronizing automata. For a synchronizing automaton there exists a reset word w of length at most n^3 [15]. Furthermore, if S is totally extensible, then there must exist a reset word w such that $Q \cdot w = \{q\} \subseteq S$, which has length at most n^3+n-1. Therefore, if the given ℓ is larger than this bound, we answer positively. Otherwise, we nondeterministically guess a word of length at most ℓ and verify whether it totally extends S.

By the same argument for Problem 12, if the automaton is strongly connected and synchronizing, then for a non-empty proper subset of Q using a reset word we can always find an extending word of length at most n^3+n-1, thus the problem is solvable in NP. □

5. Resizing a subset

In this section we deal with the following two problems:

Problem 15 *(Resizable subset).* Given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is S resizable?

Problem 16 *(Resizable subset by short word).* Given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer ℓ given in binary representation, is S resizable by a word of length at most ℓ?

In contrast to the cases $|S \cdot w^{-1}| > |S|$ and $|S \cdot w^{-1}| < |S|$, there exists a polynomial-time algorithm for both these problems. Furthermore, we prove that if S is resizable, then the length of the shortest resizing words is at most $n-1$.

To obtain a polynomial-time algorithm, one could reduce Problem 15 to the *multiplicity equivalence of NFAs*, which is the problem whether two given NFAs have the same number of accepting paths for every word. It can be solved in $\mathcal{O}(|\Sigma|n^4)$ time by a Tzeng's algorithm [47], assuming that arithmetic operations on real numbers have a unitary cost; this algorithm relies on linear algebra methods. Alternatively, it can be solved in $\mathcal{O}(|\Sigma|^2n^3)$ time by an algorithm of Archangelsky [48]. It was noted by Diekert that the Tzeng's algorithm could be improved to $\mathcal{O}(|\Sigma|n^3)$ time [48] (unpublished).

However, to obtain the tight upper bound $n-1$ on the length we need to design and analyze a specialized algorithm for our problem. It is also based on the Tzeng's linear algebraic method.

Theorem 15. *Assuming that in our computational model every arithmetic operation has a unitary cost, there is an algorithm with $\mathcal{O}(|\Sigma|n^3)$ time and $\mathcal{O}(|\Sigma|n+n^2)$ space complexity, which, given an n-state automaton $\mathscr{A}=(Q,\Sigma,\delta)$ and a subset $S\subseteq Q$, returns the minimum length ℓ such that $|S\cdot w^{-1}|\neq|S|$ for some word $w\in\Sigma^{\leq\ell}$ if it exists or reports that there is no such a word. Furthermore, we always have $1\leq\ell\leq n-1$.*

Proof. The idea of the algorithm is based on the *ascending chain condition*, often used for automata (e.g. [32,49,18]). We need to introduce a few definitions from linear algebra. We associate a natural linear structure with automaton $\mathscr{A}$. By $\mathbb{R}^n$ we denote the real n-dimensional linear space of row vectors. The value at an i-th entry of a vector $v\in\mathbb{R}^n$ we denote by $v(i)$. Without loss of generality, we assume that $Q=\{1,2,\dots,n\}$ and then assign to each subset $K\subseteq Q$ its *characteristic vector* $[K]\in\mathbb{R}^n$, whose i-th entry $v(i)=1$ if $i\in K$, and $v(i)=0$, otherwise. By $\mathrm{span}(S)$ we denote the linear span of $S\subseteq\mathbb{R}^n$. The *dimension* of a linear subspace L is denoted by $\dim(L)$.

Each word $w\in\Sigma^*$ corresponds to a linear transformation of $\mathbb{R}^n$. By $[w]$ we denote the matrix of this transformation in the standard basis $[1],\dots,[n]$ of $\mathbb{R}^n$. For example, if $\mathscr{A}$ is the automaton from Fig. 1, then

$$[a]=\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{pmatrix},\ [b]=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\1&0&0&0\end{pmatrix},\ [ba]=\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{pmatrix}.$$

Clearly, as the automaton is deterministic, the matrix $[w]$ has exactly one non-zero entry in each row. In particular, $[w]$ is *row stochastic*, which means that the sum of entries in each row is equal to 1. For every words $u,v\in\Sigma^*$, we have $[uv]=[u][v]$. By $[w]^T$ we denote the transpose of the matrix $[w]$. The transpose corresponds to the preimage by the action of a word; one verifies that $[S\cdot w^{-1}]=[S][w]^T$. For two vectors $v_1,v_2\in\mathbb{R}^n$, we denote their usual inner (scalar) product by $v_1\bullet v_2$.

Algorithm description. Now, we design the algorithm, which consists of two parts.

First, consider the auxiliary Filter function shown in Algorithm 2. Its goal is to filter a stream of vectors $g\in\mathbb{R}^n$, keeping only a subset of those vectors that are linearly independent. To perform this subroutine efficiently, we maintain a sequence of vectors G (basis) and a sequence of indices I, which are empty at the beginning. Every time, we use the Gaussian approach to reduce the matrix of vectors from G to a *pseudo-triangular* form. The sequence of (column) indices $I=(i_1,i_2,\dots,i_k)$ and vectors $G=(g_1,\dots,g_k)$ have the property that for each j, $1\leq j\leq k$, there is exactly one vector from $\{g_1,\dots,g_k\}$ with non-zero i_j-th entry, which contains 1.

Algorithm 2 Filter.

```
$G \leftarrow (), I \leftarrow ()$.                                  ▷ Global initialization
function Feed($g \in \mathbb{R}^n$)
    $g' \leftarrow g - \sum_{r=1}^{k} g(i_r) \cdot g_r$
    if $g' = 0$ then
        return False
    else
        $i' \leftarrow \min(i \mid g(i) \neq 0)$
        $g' \leftarrow g'/g'(i')$
        for all $g_r$ from $G$ do
            $g_r \leftarrow g_r - g_r(i') \cdot g'$
        end for
        Append $g'$ to $G$
        Append $i'$ to $I$
        return True
    end if
end function
```

We begin with the first non-zero vector g_1 and put its smallest index i of a non-zero entry to I, and the vector itself is normalized to have 1 in the i-th entry. Now, suppose we are given a vector g and we have already built $G=(g_1,\dots,g_k)$ and $I=(i_1,i_2,\dots,i_k)$ with aforementioned properties. Then, we just compute $g'=g-\sum_{r=1}^{k}g(i_r)\cdot g_r$. Due to the construction, all the entries at the coordinates from I in g' are zero. If there is a non-zero coordinate left in g', then we need to normalize g', and it to G, and update the previous vectors. So we take the smallest coordinate i' whose entry is non-zero in g', normalize g' to have 1 in the i'-th entry, and add g' to G. To update the previous vectors, for each r, $1\leq r\leq k$, we set $g_r\leftarrow g_r-g_r(i')\cdot g'$, which results in that g_r has now zero in the i'-th entry, and finally we add i' to I. In the opposite case, if $g'=0$, then g belongs to $\mathrm{span}(G)$ and thus should not be added.

Note that at any point, the set G is a basis of the linear span of all the processed vectors, which is a straightforward corollary from using the Gaussian approach.

Algorithm 3 Resizing a subset.

```
Require: An automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$
$W_0 \leftarrow \{[Q]\}$
for $i$ from 1 to $n-1$ do
    $D \leftarrow \{g[a] \mid g \in W_{i-1}, a \in \Sigma\}$
    $W_i \leftarrow \{\}$
    for all $z \in D$ do
        if $[S] \cdot z \neq |S|$ then
            return $i$
        else if FEED($z$) then
            Add $z$ to $W_i$
        end if
    end for
    if $W_i = \emptyset$ then
        return None
    end if
end for
return None
```

We now turn to the main procedure of our algorithm, which is shown in Algorithm 3. Our goal is to find the minimum length of a word w such that $|S \cdot w^{-1}| \neq |S|$. This is equivalent to $[S] \cdot [Q][w] \neq |S|$. We do this by using a *wave approach* as in breadth-first search. We start by feeding $[Q]$ to FILTER and let $W_0 = \{[Q]\}$. Then in each iteration $1 \leq i \leq n-1$, we consider the set of vectors $D = \{g[a] \mid g \in W_{i-1}, a \in \Sigma\}$ and build a new subset of independent vectors W_i as follows. For each vector z from D, we first check whether $[S] \cdot z = |S|$. If this is not the case, we claim that i is the length of a shortest word which changes the size of the preimage of S. Otherwise, we feed z to FILTER and add it to (initially empty) W_i if the corresponding basis vector was added to G. Note that the current G after the i-th iteration is equal to $\bigcup_{j=0}^{i} W_i$. We stop if either $W_i = \emptyset$ or the last $(n-1)$-th iteration ends, which means that there is no resizing word.

Correctness. To prove the correctness, note that by the construction all vectors from W_i can be written as $[Q][w]$ for some word w of length i. Thus, if we have found a vector $z \in D$ such that $[S] \cdot z \neq |S|$, this means there is a word w of length i such that

$$[S] \cdot [Q][w] = [S \cdot w^{-1}] \cdot [Q] = |S \cdot w^{-1}| \neq |S|.$$

It remains to show that if we get to an i-th iteration, then there is no word w of length less than i which violates $[S] \cdot [Q][w] = |S|$. For $r \geq 0$, denote $U_r = \bigcup_{i=0}^{r} W_i$. We prove by induction that for each word w of length $r < i$, $[Q][w] \in \operatorname{span}([Q][U_r])$. For $r = 0$ this is trivial. If $r > 0$, then $w = w'a$ for some $a \in \Sigma$ and by induction $[Q][w'] \in \operatorname{span}([Q][U_{r-1}])$, that is,

$$[Q][w'] = \sum_{j=0}^{r-1} \sum_{u \in W_j} \lambda_u [Q][u],$$

for some values $\lambda_u \in \mathbb{R}$. It follows that

$$[Q][w'a] = [Q][w'][a] = \sum_{j=0}^{r-1} \sum_{u \in W_j} \lambda_u [Q][u][a] = g_v + \sum_{u \in W_{r-1}} \lambda_u [Q][u][a],$$

where $g_v \in \operatorname{span}([Q][U_{r-1}])$. By the construction, we feed all vectors of the form $[Q][u][a]$ for $u \in W_{r-1}$ and $a \in \Sigma$ to FILTER function. Since the added vectors to G, and so to W_r, are a linear basis of the linear span of all the processed vectors, every vector $[Q][u][a]$ belongs to $\operatorname{span}([Q][U_r])$, which proves the induction step.

Thus, if we had a word of length w of length less than i with $[S] \cdot [Q][w] \neq |S|$, we would have $[Q][w] = \sum_{u \in U_{i-1}} \lambda_u [Q][u]$ for some $\lambda_u \in \mathbb{R}$. Now, on the one hand we have

$$n = [Q][w] \cdot [Q] = \sum_{u \in U_{i-1}} \lambda_u ([Q][u] \cdot [Q]) = n \sum_{u \in U_{i-1}} \lambda_u, \tag{1}$$

while on the other hand we have

$$|S| \neq [Q][w] \cdot [S] = \sum_{u \in U_{i-1}} \lambda_u [Q][u] \cdot [S] = \sum_{u \in U_{i-1}} \lambda_u |S|$$

contradicting (1).

On the other hand, if W_i is empty for an $i < n$, this means that $\operatorname{span}([Q][\Sigma^{\leq i}]) = \operatorname{span}([Q][\Sigma^{\leq i-1}])$ and by the linear extending argument we know that the same holds for all $j \geq i$, hence there cannot be a word that violates $[S] \cdot [Q][w] = |S|$.

Note that if there is no resizing word, then we always have this case for some $i < n$, because $\dim(\operatorname{span}([Q][w] \mid w \in \Sigma^*)) \le n-1$ and the vectors from all W_j are a basis.

We also conclude that i cannot exceed $n-1$, which proves that the shortest resizing words have length at most $n-1$. Note that the upper bound $n-1$ is the best possible, at least in the cases $|S| \in \{1, n-1\}$, which can be observed in the Černý automata (see Fig. 1 with $S = \{3\}$).

Complexity. Assume that in our computational model every arithmetic operation has a unitary cost. Then clearly a k-th call of Feed can be performed in $\mathcal{O}(kn)$-time. However, note that, if an exact computation is performed using rational numbers, then we may require to handle values of exponential order, and the total complexity would depend on the algorithms used for particular arithmetic operations.

Notice that at an i-th iteration, we call Feed at most $|\Sigma||W_i|$ times, since, by the construction, sets W_i are disjoint because the corresponding vectors are independent. Since the complexity of Feed is in $\mathcal{O}(n^2)$, all calls work in $\mathcal{O}(|\Sigma|n^3)$-time. The other operations took amortized time at most $\mathcal{O}(|\Sigma|n^2)$, which is the cost of computing sets D (at most n vectors in sets W_i; note that one $g[a]$ can be computed in $\mathcal{O}(n)$ time, because the automaton is deterministic). Thus, the whole algorithm works in $\mathcal{O}(|\Sigma|n^3)$ time.

The space complexity is at most $\mathcal{O}(|\Sigma|n + n^2)$, which is caused by storing the automaton and at most $\mathcal{O}(n^2)$ vectors in the sets W_i, G, and I. □

The running time $\mathcal{O}(|\Sigma|n^3)$ of the algorithm is quite large (and may require large arithmetic as discussed in the proof), and it is an interesting open question whether there is a faster algorithm for Problems 15 and 16.

We note that Problem 15 becomes trivial when the automaton is synchronizing: A word resizing the subset exists if and only if $S \neq \emptyset$ and $S \neq Q$, because if w is a reset word and $\{q\} = Q \cdot w$, then $S \cdot w^{-1}$ is either Q when $q \in S$ or $\emptyset$ when $q \notin S$. This implies that there exists a faster algorithm in the sense of expected running time when the automaton over at least a binary alphabet is drawn uniformly at random:

Remark 16. The algorithm from [36] checks in expected $\mathcal{O}(n)$ time (regardless of the alphabet size, which is not fixed) whether a random automaton is synchronizing, and it is synchronizing with probability $1 - \Theta(1/n^{0.5|\Sigma|})$ (for $|\Sigma| \ge 2$). Then only if it is not synchronizing we have to use the algorithm from Theorem 15. Thus, Problem 16 can be solved for a random automaton in the expected time

$$\mathcal{O}(|\Sigma|n^3) \cdot \Theta(1/n^{0.5|\Sigma|}) + \mathcal{O}(n) = \mathcal{O}(|\Sigma|n^{3-0.5|\Sigma|}) \le \mathcal{O}(n^2).$$

Note that the bound is independent on the alphabet size, and this is because a random automaton with a growing alphabet is more likely to be synchronizing, so less likely we need to use Theorem 15.

6. Conclusions

We have established the computational complexity of problems related to extending words. Indirectly, our results about the complexity imply also the bounds on the length of the shortest compressing/extending words, which are of separate interest. In particular, PSPACE-hardness implies that the shortest words can be exponentially long in this case, and polynomial deterministic or nondeterministic algorithms in our proofs imply polynomial upper bounds. For example, the question about the length of the shortest totally extending words (in the equivalent terms of compressing Q to a subset included in S) was recently considered [41], and from our results (PSPACE-completeness) we could infer an answer that the tight upper bound is exponential. The algorithm from Theorem 12 implies also a bound on the length of the shortest avoiding words for a subset. That length is at least cubic, which is useless in the case of synchronizing automata, since reset words can be used as avoiding words and there exists a cubic upper bound on the length of the shortest reset words [17,18].

Some problems are left open. In Tables 1 and 2 there is a gap. The complexity of the existence of an extending word when the subset is large (Problem 9) and the automaton is strongly connected is unknown. The same holds in the case when the length of the extending word is bounded (Problem 12); now, we can only conclude that it is NP-hard, which follows from Corollary 14. The proof of Theorem 10 relies on the automaton being not strongly connected.

Further questions may concern other complexity classes like NL (cf. Theorem 6). Also, one could try improving the complexity of algorithms, in particular, those from Theorems 11 and 12 for avoiding words, and also that from Theorem 15 for resizing words.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the anonymous referee for careful reading and detailed comments. This work was supported by the Competitiveness Enhancement Program of Ural Federal University under grant No. 02.A03.21.006 (Mikhail Berlinkov), and by the National Science Centre, Poland under project number 2014/15/B/ST6/00615 (Robert Ferens) and 2017/25/B/ST6/01920 (Marek Szykuła).

References

[1] M.V. Berlinkov, R. Ferens, M. Szykuła, Complexity of preimage problems for deterministic finite automata, in: Mathematical Foundations of Computer Science, in: LIPIcs, vol. 117, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 32.
[2] J. Berstel, D. Perrin, C. Reutenauer, Codes and Automata, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 2009.
[3] H. Jürgensen, Synchronization, Inf. Comput. 206 (9–10) (2008) 1033–1044.
[4] S. Sandberg, Homing and synchronizing sequences, in: Model-Based Testing of Reactive Systems, in: LNCS, vol. 3472, Springer, 2005, pp. 5–33.
[5] B.K. Natarajan, An algorithmic approach to the automated design of parts orienters, in: Foundations of Computer Science, in: SFCS, IEEE Computer Society, 1986, pp. 132–142.
[6] Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, E. Shapiro, DNA molecule provides a computing machine with both data and fuel, Proc. Natl. Acad. Sci. USA 100 (5) (2003) 2191–2196.
[7] J. Araújo, P.J. Cameron, B. Steinberg, Between primitive and 2-transitive: synchronization and its friends, EMS Surv. Math. Sci. 4 (2017) 101–184.
[8] J. Almeida, S. Margolis, B. Steinberg, M. Volkov, Representation theory of finite semigroups, semigroup radicals and formal language theory, Trans. Am. Math. Soc. 361 (2009) 1429–1461.
[9] J. Olschewski, M. Ummels, The complexity of finding reset words in finite automata, in: Mathematical Foundations of Computer Science, in: LNCS, vol. 6281, Springer, 2010, pp. 568–579.
[10] F. Gonze, R.M. Jungers, On the synchronizing probability function and the triple rendezvous time for synchronizing automata, SIAM J. Discrete Math. 30 (2) (2016) 995–1014.
[11] N. Rampersad, J. Shallit, Z. Xu, The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages, Fundam. Inform. 116 (1–4) (2012) 223–236.
[12] P. Gawrychowski, D. Straszak, Strong inapproximability of the shortest reset word, in: Mathematical Foundations of Computer Science, in: LNCS, vol. 9234, Springer, 2015, pp. 243–255.
[13] V.D. Blondel, R.M. Jungers, A. Olshevsky, On primitivity of sets of matrices, Automatica 61 (2015) 80–88.
[14] M. Grech, A. Kisielewicz, Černý conjecture for edge-colored digraphs with few junctions, Electron. Notes Discrete Math. 54 (2016) 115–120.
[15] M.V. Volkov, Synchronizing automata and the Černý conjecture, in: Language and Automata Theory and Applications, in: LNCS, vol. 5196, Springer, 2008, pp. 11–27.
[16] J. Černý, Poznámka k homogénnym eksperimentom s konečnými automatami, Mat.-Fyz. Čas. Slov. Akad. Vied 14 (3) (1964) 208–216, in Slovak.
[17] Y. Shitov, An improvement to a recent upper bound for synchronizing words of finite automata, J. Autom. Lang. Comb. 24 (2–4) (2019) 367–373.
[18] M. Szykuła, Improving the upper bound on the length of the shortest reset word, in: Symposium on Theoretical Aspects of Computer Science, in: LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 56.
[19] D. Eppstein, Reset sequences for monotonic automata, SIAM J. Comput. 19 (1990) 500–510.
[20] D.S. Ananichev, V.V. Gusev, Approximation of reset thresholds with greedy algorithms, Fundam. Inform. 145 (3) (2016) 221–227.
[21] M. Berlinkov, M. Szykuła, Algebraic synchronization criterion and computing reset words, Inf. Sci. 369 (2016) 718–730.
[22] A. Kisielewicz, J. Kowalski, M. Szykuła, Computing the shortest reset words of synchronizing automata, J. Comb. Optim. 29 (1) (2015) 88–124.
[23] A. Roman, M. Szykuła, Forward and backward synchronizing algorithms, Expert Syst. Appl. 42 (24) (2015) 9512–9527.
[24] D.S. Ananichev, M.V. Volkov, Synchronizing generalized monotonic automata, Theor. Comput. Sci. 330 (1) (2005) 3–13.
[25] M.T. Biskup, W. Plandowski, Shortest synchronizing strings for Huffman codes, Theor. Comput. Sci. 410 (38–40) (2009) 3925–3941.
[26] M. Grech, A. Kisielewicz, The Černý conjecture for automata respecting intervals of a directed graph, Discrete Math. Theor. Comput. Sci. 15 (3) (2013) 61–72.
[27] A.N. Trahtman, The Černý conjecture for aperiodic automata, Discrete Math. Theor. Comput. Sci. 9 (2) (2007) 3–10.
[28] M.V. Volkov, Synchronizing automata preserving a chain of partial orders, Theor. Comput. Sci. 410 (37) (2009) 3513–3519.
[29] M.-P. Béal, M. Berlinkov, D. Perrin, A quadratic upper bound on the size of a synchronizing word in one-cluster automata, Int. J. Found. Comput. Sci. 22 (2) (2011) 277–288.
[30] M. Berlinkov, Synchronizing quasi-Eulerian and quasi-one-cluster automata, Int. J. Found. Comput. Sci. 24 (6) (2013) 729–745.
[31] R.M. Jungers, The synchronizing probability function of an automaton, SIAM J. Discrete Math. 26 (1) (2012) 177–192.
[32] J. Kari, Synchronizing finite automata on Eulerian digraphs, Theor. Comput. Sci. 295 (1–3) (2003) 223–232.
[33] B. Steinberg, The Černý conjecture for one-cluster automata with prime length cycle, Theor. Comput. Sci. 412 (39) (2011) 5487–5491.
[34] I.K. Rystsov, Polynomial complete problems in automata theory, Inf. Process. Lett. 16 (3) (1983) 147–151.
[35] V. Vorel, Subset synchronization of transitive automata, in: Automata and Formal Languages, in: EPTCS, 2014, pp. 370–381.
[36] M. Berlinkov, On the probability of being synchronizable, in: Conference on Algorithms and Discrete Applied Mathematics, in: LNCS, vol. 9602, Springer, 2016, pp. 73–84.
[37] V. Vorel, Complexity of a problem concerning reset words for Eulerian binary automata, Inf. Comput. 253 (Part 3) (2017) 497–509.
[38] P. Martyugin, Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata, Theory Comput. Syst. 54 (2) (2014) 293–304.
[39] K. Guldstrand Larsen, S. Laursen, J. Srba, Synchronizing strategies under partial observability, in: International Conference on Concurrency Theory, in: LNCS, vol. 8704, Springer, 2014, pp. 188–202.
[40] E.A. Bondar, M.V. Volkov, Completely reachable automata, in: Descriptional Complexity of Formal Systems, in: LNCS, Springer, 2016, pp. 1–17.
[41] F. Gonze, R.M. Jungers, On completely reachable automata and subset reachability, in: Developments in Language Theory, in: LNCS, vol. 11088, Springer, 2018, pp. 330–341.
[42] F. Gonze, R.M. Jungers, A.N. Trahtman, A note on a recent attempt to improve the Pin-Frankl bound, Discrete Math. Theor. Comput. Sci. 17 (1) (2015) 307–308.
[43] B. Steinberg, The averaging trick and the Černý conjecture, Int. J. Found. Comput. Sci. 22 (7) (2011) 1697–1706.
[44] D. Kozen, Lower bounds for natural proof systems, in: Foundations of Computer Science, in: SFCS, IEEE Computer Society, 1977, pp. 254–266.
[45] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (2) (1972) 146–160.
[46] M. Berlinkov, On two algorithmic problems about synchronizing automata, in: Developments in Language Theory, in: LNCS, Springer, 2014, pp. 61–67.

[47] W.-G. Tzeng, The equivalence and learning of probabilistic automata, in: Foundations of Computer Science, in: SFCS, IEEE Computer Society, 1989, pp. 268–273.

[48] K. Archangelsky, Efficient algorithm for checking multiplicity equivalence for the finite z - σ^*-automata, in: Developments in Language Theory, Springer, 2003, pp. 283–289.

[49] J.-E. Pin, Utilisation de l'algèbre linéaire en théorie des automates, in: Actes du 1er Colloque AFCET-SMF de Mathématiques Appliquées II, AFCET, 1978, pp. 85–92, in French.