Complexity of Preimage Problems for Deterministic Finite Automata

Mikhail V. Berlinkov[1]
Institute of Natural Sciences and Mathematics, Ural Federal University, Ekaterinburg, Russia
berlm@mail.ru

Robert Ferens[2]
Institute of Computer Science, University of Wrocław, Wrocław, Poland
robert.ferens@interia.pl

Marek Szykuła[3]
Institute of Computer Science, University of Wrocław, Wrocław, Poland
msz@cs.uni.wroc.pl

Abstract

Given a subset of states S of a deterministic finite automaton and a word w, the preimage is the subset of all states that are mapped to a state from S by the action of w. We study the computational complexity of three problems related to the existence of words yielding certain preimages, which are especially motivated by the theory of synchronizing automata. 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 word totally extending the subset (giving the whole set of states) – it is equivalent to the problem whether there exists an avoiding word for the complementary subset. The third problem is whether there exists a word resizing the subset (giving a preimage of a different size). We also consider the variants of the problem where an upper bound on the length of the word is given in the input. Because in most cases our problems are computationally hard, we additionally consider parametrized complexity by the size of the given subset. We focus on the most interesting cases that are the subclasses of strongly connected, synchronizing, and binary automata.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases avoiding word, extending word, extensible subset, reset word, synchronizing automaton

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.32

Related Version `https://arxiv.org/abs/1704.08233`

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 $|Q|$.

[1] Supported by Russian Foundation for Basic Research, grant no. 16-01-00795, and the Competitiveness Enhancement Program of Ural Federal University.
[2] Supported in part by the National Science Centre, Poland under project number 2014/15/B/ST6/00615.
[3] Supported in part by the National Science Centre, Poland under project number 2017/25/B/ST6/01920.

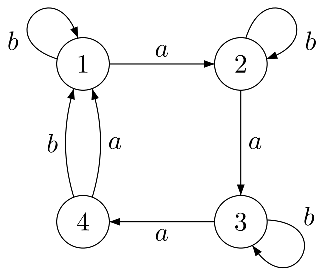

Figure 1 The Černý automaton with 4 states.

When the context automaton 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. 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 $\{2, 3\} \cdot (ba)^{-1} = \{1, 2\} \cdot b^{-1} = \{1, 2, 4\}$.

In fact, 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 $\{1, 4\}$ is shrank by b to subset $\{4\}$.

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. $|S \cdot w^{-1}| > |S|$ 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, such as coding theory, DNA-computing, robotics, testing of reactive systems, and theory of information sources. They also reveal interesting connections with symbolic dynamics, language theory, group theory, and many other parts of mathematics. For a detailed introduction to the theory of synchronizing automata we refer the reader to the survey [31], and for a review of relations with coding theory to [16] and [8].

The famous Černý conjecture [11], which was formally stated in 1969 during a conference ([31]), is one of the most longstanding open problems in automata theory, and is the central problem in the theory of synchronizing automata. It states that a synchronizing automaton has a reset word of length at most $(n-1)^2$. Besides the conjecture, algorithmic issues are also important. Unfortunately, the problem of finding a *shortest* reset word is computationally hard [12, 21], and also its length approximation remains hard [13]. We also refer to surveys [24, 31] about algorithmic issues and the Černý conjecture.

Our general motivation comes from the fact that words compressing and extending subsets play a crucial role in synchronization automata. In fact, all known algorithms finding a reset word as intermediate steps use finding words that either compresses or extends a subset (e.g. [1, 7, 12, 18, 22]). Moreover, probably all proofs of upper bounds on the length of the shortest reset words use bounding the length of words that compress (e.g. [2, 7, 9, 12, 14, 27, 29, 32]) or extend (e.g. [3, 4, 7, 17, 26, 27]) some subsets.

In this paper, we study several natural problems related to preimages. Our goal is to provide a systematic view of their computational complexity and solve several open problems.

1.1 Compressing a Subset

The complexities of problems related to compressing 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 [23]. The same holds even for strongly connected binary automata [33].

On the other hand, checking whether the automaton is synchronizing (whether there is a word that synchronizes Q) can be solved in $\mathcal{O}(|\Sigma|n^2)$ time and space [11, 12, 31] and in $\mathcal{O}(n)$ average time and space for the random binary case [6]. To this end, we just 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 [12] (cf. [21] for the complexity of the corresponding functional problems), even if the given automaton is binary. There exist stronger results, such as NP-completeness of this problem when the automaton is Eulerian and binary [34], which immediately implies that for the class of strongly connected automata the complexity is the same.

However, deciding whether there exists a word 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 have been also studied in other settings than DFAs. We refer to [20, 23] for the cases of NFA and PDFA (*partial deterministic finite automata*), and to [15] for the partial observability setting. Finally, in [10] the problem of reachability of a given subset in a DFA has been studied.

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 well studied. In the paper, we fill this gap. We study three families of problems. As we noted before, extending is equivalent to shrinking the complement, hence we deal only with the extending word problems.

Extending words: Our first family of problems is the question whether there exists an extending word (Problems 1, 7, 9, 13, 16, 22).

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 that is used in many proofs and also some algorithms. The extension method of finding a reset word is to start from some singleton $S_0 = \{q\}$, and iteratively find extending words $w_1, \ldots, w_k$ 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 final $S_0 \cdot w_1^{-1} \cdots w_k^{-1} = Q$. For finding a short reset word one needs to bound the lengths of the extending words. For instance, by showing that in the case of Eulerian automata there are always extending words of length at most n, which implies the upper bound $(n-2)(n-1)+1$ on the length of the shortest reset words for this class [17]. In this case, a polynomial algorithm for finding extending words has been proposed in [7].

Table 1 Computational complexity of decision problems in classes of automata. 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:

Subclass of automata	All automata	Strongly connected	Synchronizing	Str. con. and synch.
$\lvert S \cdot w\rvert = 1$ (reset word)	PSPACE-c [23, 33]		$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\lvert S \cdot w\rvert < \lvert S\rvert$ (compressing word)	$\mathcal{O}(\lvert\Sigma\rvert n^2)$ [11, 31]		$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\lvert S \cdot w^{-1}\rvert > \lvert S\rvert$ (Problem 1)	PSPACE-c (Thm. 3)		PSPACE-c (Prop. 5)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$ (Problem 2)	PSPACE-c (Thm. 3)		$\mathcal{O}(\lvert\Sigma\rvert n)$ (Thm. 6)	$\mathcal{O}(1)$
$\lvert S \cdot w^{-1}\rvert > \lvert S\rvert$, $\lvert S\rvert \le k$ (Problem 9)	$\mathcal{O}(\lvert\Sigma\rvert n^k)$ (Prop. 10)		$\mathcal{O}(\lvert\Sigma\rvert n^k)$ (Prop. 10)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\lvert S\rvert \le k$ (Problem 11)	$\mathcal{O}(\lvert\Sigma\rvert(n^3 + n^k))$ (Prop. 12)		$\mathcal{O}(\lvert\Sigma\rvert n)$ (Thm. 6)	$\mathcal{O}(1)$
$\lvert S \cdot w^{-1}\rvert > \lvert S\rvert$, $\lvert S\rvert \ge n - k$ (Problem 16, $k \ge 2$)	PSPACE-c (Thm. 19)	Open	PSPACE-c (Thm. 19)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\lvert S\rvert \ge n - k$ (Problem 17, $k \ge 2$)	$\mathcal{O}(n^3 + \lvert\Sigma\rvert n^k)$ (Thm. 21)		$\mathcal{O}(\lvert\Sigma\rvert n)$ (Thm. 6)	$\mathcal{O}(1)$
$S \cdot w^{-1} = Q$, $\lvert S\rvert = n - 1$ (Problem 18)	$\mathcal{O}(\lvert\Sigma\rvert n^2)$ (Thm. 20)		$\mathcal{O}(\lvert\Sigma\rvert)$	$\mathcal{O}(1)$
$\lvert S \cdot w^{-1}\rvert \ne \lvert S\rvert$ (Problem 27)	$\mathcal{O}(\lvert\Sigma\rvert n^3)$ (Thm. 29)		$\mathcal{O}(1)$	$\mathcal{O}(1)$

Totally extending words and avoiding: We study the problem whether there exists a totally extending word (Problems 2, 8, 11, 14, 17, 23). The question about the existence of a totally extending word is equivalent to the question about 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 to not be in them. A quadratic upper bound on the length of the shortest avoiding words of a single state have been established in [27], where avoiding words were also used to improve the best known upper bound on the length of the shortest reset words. The computational complexity of the problems related to avoiding, both a single state or a subset, have not been established, which is another motivation to study

Table 2 Computational complexity of decision problems in classes of automata. 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 $\leq \ell$ such that:

Subclass of automata	All automata	Strongly connected	Synchronizing	Str. con. and synch.
$\lvert S \cdot w \rvert = 1$ (reset word)	PSPACE-c [23, 33]		NP-c [12]	NP-c [34]
$\lvert S \cdot w \rvert < \lvert S \rvert$ (compressing word)	$\mathcal{O}(\lvert\Sigma\rvert n^2)$ [12]		$\mathcal{O}(\lvert\Sigma\rvert n^2)$ [12]	$\mathcal{O}(\lvert\Sigma\rvert n^2)$ [12]
$\lvert S \cdot w^{-1} \rvert > \lvert S \rvert$ (Problem 7)	PSPACE-c (Subsec. 2.1)		PSPACE-c (Subsec. 2.1)	NP-c (Thm. 25)
$S \cdot w^{-1} = Q$ (Problem 8)	PSPACE-c (Subsec. 2.1)		NP-c (Cor. 26)	NP-c (Cor. 26)
$\lvert S \cdot w^{-1} \rvert > \lvert S \rvert$, $\lvert S \rvert \leq k$ (Problem 13)	$\mathcal{O}(\lvert\Sigma\rvert n^k)$ (Prop. 10)		$\mathcal{O}(\lvert\Sigma\rvert n^k)$ (Prop. 10)	$\mathcal{O}(\lvert\Sigma\rvert n^k)$ (Prop. 10)
$S \cdot w^{-1} = Q$, $\lvert S \rvert \leq k$ (Problem 14)	NP-c (Prop. 15)		NP-c (Prop. 15)	NP-c (Prop. 15)
$\lvert S \cdot w^{-1} \rvert > \lvert S \rvert$, $\lvert S \rvert \geq n - k$ (Problem 22, $k \geq 2$)	PSPACE-c (Thm. 19)	Open	PSPACE-c (Thm. 19)	NP-c (Cor. 26)
$S \cdot w^{-1} = Q$, $\lvert S \rvert \geq n - k$ (Problem 23, $k \geq 2$)	NP-c (Cor. 26)		NP-c (Cor. 26)	NP-c (Cor. 26)
$S \cdot w^{-1} = Q$, $\lvert S \rvert = n - 1$ (Problem 24)	NP-c (Thm. 25)		NP-c (Thm. 25)	NP-c (Thm. 25)
$\lvert S \cdot w^{-1} \rvert \neq \lvert S \rvert$ (Problem 28)	$\mathcal{O}(\lvert\Sigma\rvert n^3)$ (Thm. 29)		$\mathcal{O}(\lvert\Sigma\rvert n^3)$ (Thm. 29)	$\mathcal{O}(\lvert\Sigma\rvert n^3)$ (Thm. 29)

totally extending words. We give a special attention to the problem of avoiding one state and a small subset of states (totally extending a large subset), 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 27, 28).

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 (e.g. [7, 17, 25]).

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 binary form in the input. Since in most cases, the problems are computationally hard, in Section 3 and Section 4 we consider parameterized complexity 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

We deal with the following problems:

▶ **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 there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$?*

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

Proof idea. To solve both problems in NPSPACE, we just guess the length of a (totally) extending word and then subsequently its letters, storing only the current subset all the time.

For PSPACE-hardness, we perform a reduction from the problem of determining whether an intersection of regular languages given as DFAs is non-empty [19]. 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. ◀

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 $\mathscr{A}'$ 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 idea. Let $\Sigma = \{a_1, \ldots, a_k\}$. 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 the states from S in the first copy and all states from the other copies. ◀

Now we consider the subclass of synchronizing automata.

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

Proof idea. We just add a sink state z and a letter which synchronizes $\mathscr{A} = (Q, \Sigma, \delta)$ to z. Additionally, a standard tree-like binarization is suitably used to obtain a binary automaton that preserves extensibility of the subset. ◀

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

Proof idea. Since $\mathscr{A}$ is synchronizing, the problem 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 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 [28]. ◀

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.

2.1 Bounded Length of the Word

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

▶ **Problem 7** (Extensible subset by short word)**.** *Given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer ℓ given in binary form, is S extensible by a word of length at most ℓ?*

▶ **Problem 8** (Totally extensible subset by short word)**.** *Given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer ℓ given in binary form, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$ 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 8 is NP-complete, which will be shown in Corollary 26. Of course, Problem 7 remains PSPACE-complete for a synchronizing automaton by the same argument as in the general case.

3 Extending Small Subsets

The complexity of extending problems rely on the size of the given subset. Variable subset size is essential for hardness. 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.

▶ **Problem 9** (Extensible small subset)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|S| \le k$, is there a word extending S?*

▶ **Proposition 10.** *Problem 9 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. The size (number of states and edges) of this automaton is bounded by $\mathcal{O}(|\Sigma|n^k)$, so the procedure takes this time. ◀

▶ **Problem 11** (Totally extensible small subset)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|S| \le k$, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$?*

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 12.** *Problem 11 can be solved in $\mathcal{O}(|\Sigma|(n^3 + n^k))$ 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 $\mathcal{O}(|\Sigma|n^3)$ time, using e.g. the algorithm from [12].

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 10). 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. 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. ◀

3.1 Bounded Length of the Word

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

▶ **Problem 13** (Extensible small subset by short word)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|S| \le k$, and an integer ℓ given in binary form, is there a word extending S of length at most ℓ?*

Problem 13 can be solved by the same algorithm in a Proposition 10, since the procedure can find a shortest extending word.

▶ **Problem 14** (Totally extensible small subset by short word)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|S| \le k$, and an integer ℓ given in binary form, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$ of length at most ℓ?*

▶ **Proposition 15.** *For every k, Problem 14 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 12).

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 [34] 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 [34, Lemma 2.4]). Therefore, we can set $S = \{s_2\}$, and thus Problem 14 is NP-complete. ◀

4 Extending Large Subsets

We consider here the case when the subset S contains all except at most a fixed number of states k.

▶ **Problem 16** (Extensible large subset)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, is there a word extending S?*

▶ **Problem 17** (Totally extensible large subset)**.** *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$?*

Problem 17 is equivalent to deciding the existence of an avoiding word for a subset S of size $\le k$. Note that both problems are equivalent for $k = 1$, which is the problem of avoiding a single given state. Their properties will also turn out to be different than in the case of $k \ge 2$. We give a special attention to this problem and study it separately.

▶ **Problem 18** (Avoidable state)**.** *Given $\mathscr{A} = (Q, \Sigma, \delta)$ and a state $q \in Q$, is there a word $w \in \Sigma^*$ such that $q \notin Q \cdot w$?*

The following result may be a bit surprising, in view of that it is the only case where the general problem remains equally hard when the subset size is bounded. We state that the first problem remains PSPACE-complete for all $k \ge 2$, although the problem remains open for strongly connected automata.

▶ **Theorem 19.** *Problem 16 is PSPACE-hard for every $k \ge 2$ and $|\Sigma| \ge 2$ even if the given automaton is synchronizing.*

Proof idea. We show a reduction from PSPACE-complete Problem 2 (Thm. 3). Our obtained automaton is ternary and synchronizing with a sink state e'.

We reduce the alphabet to two letters by an application of the Theorem 4 to Problem 16. The reduction in the proof keeps the size of complement set the same, so we can apply it. Furthermore, we identify all copies of the sink state e', which ensures that it remains synchronizing. ◀

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 20.** *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. Let p and w be the state and the word from the theorem for a given state q. Since the automaton is strongly connected, there is a word w' such that 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 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 defined by

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

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 $|(Q \cdot w) \cap [p]_\sim| \le 1$, 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 20 we know that given $q \in Q$, it is sufficient to check whether q belongs to a compressible pair of states. Hence, Problem 18 can be solved using the well-known algorithm [12] 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 20. From the following theorem, it follows that Problem 17 for $k \geq 2$ can be solved in polynomial time.

▶ **Theorem 21.** *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 $\mathcal{O}(n^3 + |\Sigma|(n^2 + n^{\min(r,k)}))$ time and $\mathcal{O}(n^2 + n^{\min(r,k)} + |\Sigma|n)$ space. Moreover the length of w is bounded by $\mathcal{O}(n^3 + n^{\min(r,k)}))$.*

Proof. Similarly to the proof of Theorem 20, let u be a word of the minimal rank r in $\mathscr{A}$ and let $\sim$ be the equivalence relation on Q defined by

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

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.

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 $s \in S$.

Suppose that S is avoidable, and let w' be an avoiding word for S. Then the word $w = uw'$ also avoids S. Observe that w has rank r as u has. 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 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 a unique state $q_{[p]_\sim} \in (Q \cdot w) \cap [p]_\sim$.

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 Alg. 1 to find w or verify that S cannot be avoided.

Alg. 1 first finds a word u of the minimal rank. This can be done by iterative compressing the subset as long as possible by the algorithm from [12], which works in $\mathcal{O}(n^3 + |\Sigma|n^2)$ time and $\mathcal{O}(n^2 + |\Sigma|n)$ space. For every subset $Q' \subseteq Q \cdot u$ of size $z = |S/{\sim}|$ the algorithm checks

Algorithm 1 Avoiding a subset.

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

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. ◀

4.1 Bounded Length of the Word

We now turn our attention to the variants of the problems where an upper bound on the length of the word is given.

▶ **Problem 22** (Extensible large subset by short word). *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q,\Sigma,\delta)$, a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, and an integer ℓ given in binary form, is there a word extending S of length at most ℓ?*

▶ **Problem 23** (Totally extensible large subset by short word). *For a fixed $k \in \mathbb{N}$, given $\mathscr{A} = (Q,\Sigma,\delta)$, a subset $S \subseteq Q$ with $|Q \setminus S| \le k$, and an integer ℓ given in binary form, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$ of length at most ℓ?*

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

▶ **Problem 24** (Avoidable state by short word). *Given $\mathscr{A} = (Q,\Sigma,\delta)$, a state $q \in Q$, and an integer ℓ given in binary form, is there a word $w \in \Sigma^*$ such that $q \notin Q \cdot w$ of length at most ℓ?*

Problem 22 for $k \ge 2$ obviously remains PSPACE-complete. By the following theorem, we show that Problem 24 is NP-complete, which then implies NP-completeness of Problem 23 for every $k \ge 1$ (by Corollary 26).

▶ **Theorem 25.** *Problem 24 is NP-complete, even if the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof idea. 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 $\mathcal{O}(n^2)$ [27]. Then we can guess an avoiding word w of at most quadratic length and compute $Q \cdot w$ in $\mathcal{O}(n^3)$ time.

To prove NP-hardness, we show a reduction from the problem of determining the reset threshold in the specific subclass of automata constructed in the Eppstein's proof of [12, Theorem 8], which is known to be NP-complete. The reduction has two steps. First, we construct a strongly connected synchronizing ternary automaton for which deciding about the length of an avoiding word is equivalent to determining the existence of a reset word in the original automaton. Then, based on the ideas from [5] we turn the automaton into a binary one that still has the desired properties. ◀

As a corollary from Theorem 25 and Theorem 21, we complete the results.

▶ **Corollary 26.** *Problem 23 is NP-complete, Problem 8 is NP-complete when the automaton is synchronizing, and Problem 22 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.*

5 Resizing a Subset

▶ **Problem 27** (Resizable subset)**.** *Given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$?*

▶ **Problem 28** (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 form, is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$ 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.

▶ **Theorem 29.** *Given an automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, there exists an algorithm working in $\mathcal{O}(|\Sigma|n^3)$ time (assuming constant time arithmetic of integers whose values are bounded by $\mathcal{O}(2^n)$) that computes a shortest word w such that $|S \cdot w^{-1}| \neq |S|$ or verifies that there is no such word. Moreover, the length of the shortest such words is a at most $n - 1$.*

Proof idea. We construct a reduction to the problem of multiplicity equivalence of NFAs and apply the algorithm from [30] with an improvement to achieve the desired complexity[4]. ◀

The running time $\mathcal{O}(|\Sigma|n^3)$ of the algorithm is quite large (and may require large arithmetic), and it is an interesting open question whether there is a faster algorithm for Problems 27 and 28.

We note that Problem 27 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 an at least binary alphabet is drawn uniformly at random:

▶ Remark. The algorithm from [6] 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 $|\Sigma| \geq 2$). Then only if it is not synchronizing we

[4] In a previous version of our proof we presented our own algorithm having $\mathcal{O}(|\Sigma|n^3)$ time complexity under the assumption of performing arithmetic computations in constant time. We thank one of the anonymous reviewers that suggested a shortcut by reducing to multiplicity equivalence of NFAs.

have to use the algorithm from Theorem 29. Thus, Problem 28 can be solved for a random automaton in the expected time

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

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 29.

References

1 D. S. Ananichev and V. V. Gusev. Approximation of Reset Thresholds with Greedy Algorithms. *Fundamenta Informaticae*, 145(3):221–227, 2016.

2 D. S. Ananichev and M. V. Volkov. Synchronizing generalized monotonic automata. *Theoretical Computer Science*, 330(1):3–13, 2005.

3 M.-P. Béal, M. Berlinkov, and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. *International Journal of Foundations of Computer Science*, 22(2):277–288, 2011.

4 M. Berlinkov. Synchronizing Quasi-Eulerian and Quasi-one-cluster Automata. *International Journal of Foundations of Computer Science*, 24(6):729–745, 2013.

5 M. Berlinkov. On Two Algorithmic Problems about Synchronizing Automata. In *Developments in Language Theory*, LNCS, pages 61–67. Springer, 2014.

6 M. Berlinkov. On the probability of being synchronizable. In *CALDAM*, volume 9602 of *LNCS*, pages 73–84. Springer, 2016.

7 M. Berlinkov and M. Szykuła. Algebraic synchronization criterion and computing reset words. *Information Sciences*, 369:718–730, 2016.

8 J. Berstel, D. Perrin, and C. Reutenauer. *Codes and Automata*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2009.

9 M. T. Biskup and W. Plandowski. Shortest synchronizing strings for Huffman codes. *Theoretical Computer Science*, 410(38-40):3925–3941, 2009.

10 E. A. Bondar and M. V. Volkov. Completely reachable automata. In C. Câmpeanu, F. Manea, and J. Shallit, editors, *DCFS*, LNCS, pages 1–17. Springer, 2016.

11 J. Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fyzikálny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. In Slovak.

12 D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19:500–510, 1990.

13 P. Gawrychowski and D. Straszak. Strong inapproximability of the shortest reset word. In *Mathematical Foundations of Computer Science*, volume 9234 of *LNCS*, pages 243–255. Springer, 2015.

14 M. Grech and A. Kisielewicz. The Černý conjecture for automata respecting intervals of a directed graph. *Discrete Mathematics and Theoretical Computer Science*, 15(3):61–72, 2013.

15 K. Guldstrand Larsen, S. Laursen, and J. Srba. Synchronizing Strategies under Partial Observability. In P. Baldan and D. Gorla, editors, *CONCUR 2014*, pages 188–202. Springer, 2014.

16 H. Jürgensen. Synchronization. *Information and Computation*, 206(9-10):1033–1044, 2008.

17 J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295(1-3):223–232, 2003.

18 A. Kisielewicz, J. Kowalski, and M. Szykuła. Computing the shortest reset words of synchronizing automata. *Journal of Combinatorial Optimization*, 29(1):88–124, 2015.

19 D. Kozen. Lower Bounds for Natural Proof Systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, FOCS, pages 254–266, 1977.

20 P. Martyugin. Computational Complexity of Certain Problems Related to Carefully Synchronizing Words for Partial Automata and Directing Words for Nondeterministic Automata. *Theory of Computing Systems*, 54(2):293–304, 2014.

21 J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In *Mathematical Foundations of Computer Science*, volume 6281 of *LNCS*, pages 568–579. Springer, 2010.

22 A. Roman and M. Szykuła. Forward and backward synchronizing algorithms. *Expert Systems with Applications*, 42(24):9512–9527, 2015.

23 I. K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983.

24 S. Sandberg. Homing and synchronizing sequences. In *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 5–33. Springer, 2005.

25 B. Steinberg. The averaging trick and the Černý conjecture. *International Journal of Foundations of Computer Science*, 22(7):1697–1706, 2011.

26 B. Steinberg. The Černý conjecture for one-cluster automata with prime length cycle. *Theoretical Computer Science*, 412(39):5487–5491, 2011.

27 M. Szykuła. Improving the Upper Bound on the Length of the Shortest Reset Word. In *STACS 2018*, LIPIcs, pages 56:1–56:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

28 R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

29 A. N. Trahtman. The Černý conjecture for aperiodic automata. *Discrete Mathematics and Theoretical Computer Science*, 9(2):3–10, 2007.

30 W.-G. Tzeng. The Equivalence and Learning of Probabilistic Automata. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, FOCS, pages 268–273. IEEE Computer Society, 1989.

31 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Language and Automata Theory and Applications*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.

32 M. V. Volkov. Synchronizing automata preserving a chain of partial orders. *Theoretical Computer Science*, 410(37):3513–3519, 2009.

33 V. Vorel. Subset Synchronization of Transitive Automata. In *Proceedings 14th International Conference on Automata and Formal Languages (AFL 2014)*, pages 370–381, 2014.

34 V. Vorel. Complexity of a problem concerning reset words for Eulerian binary automata. *Information and Computation*, 253(Part 3):497–509, 2017.