COMPLEXITY OF PREIMAGE PROBLEMS FOR DETERMINISTIC FINITE AUTOMATA

MIKHAIL V. BERLINKOV

Institute of Natural Sciences and Mathematics, Ural Federal University, Ekaterinburg, Russia

ROBERT FERENS

Institute of Computer Science, University of Wrocław, Wrocław, Poland

MAREK SZYKULA

Institute of Computer Science, University of Wrocław, Wrocław, Poland

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 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 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. Since usually 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.

Keywords: avoiding word, extending word, extensible subset, reset word, synchronizing automaton

1. Introduction

A deterministic finite complete (semi)automaton $A$ is a triple $(Q, \Sigma, \delta)$, where $Q$ is the set of states, $\Sigma$ is the input alphabet, and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function. We extend $\delta$ 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 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 1.** A word $w \in \Sigma^*$ is avoiding for $S \subseteq Q$ if and only if $w$ is totally extending for $Q \setminus S$.

![Figure 1. The Černý automaton with 4 states.](image)

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 shrunk by $b$ to subset $\{4\}$.

We note that the cases in which $S$ is shrank (its size is decreased) or its preimage is empty under the inverse action correspond to extending and totally extending cases.

**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 $(Q \setminus S) \cdot w^{-1} = \emptyset$.

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 [32], and for a review of relations with coding theory to [17] and [9].

The famous Černý conjecture [12], which was formally stated in 1969 during a conference [32], 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 \([22]\), and also its length approximation remains hard \([14]\). We also refer to surveys \([25, 32]\) 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. \([1, 3, 13, 23]\)). 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. \([2, 8, 10, 13, 15, 28, 30, 33]\)) or extend (e.g. \([3, 5, 18, 27, 28]\)) some subsets.

In this paper we study several natural problems related to preimages. Our goal is to provide a systematic view on 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 \(A\) and a subset \(S \subseteq Q\), determining whether there is a word that synchronizes it is PSPACE-complete \([24]\). The same holds even for strongly connected binary automata \([34]\).

On the other hand, checking whether the automaton is synchronizing (whether there is a word that synchronizes \(Q\)) can be solved in \(O(|\Sigma|n^2)\) time and space \([12, 13, 32]\) and in \(O(n)\) average time and space for the random binary case \([7]\). 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 \([13]\) (cf. \([22]\) 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 \([32]\), 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 \(O(|\Sigma|n^2)\) 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 \([21, 24]\) for the cases of NFA and PDFA (partial deterministic finite automata), and to \([16]\) for the partial observability setting. Finally, in \([11]\) 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.

Extend words: Our first family of problems is the question whether there exists an extending word (Problems \([3, 5, 7, 9, 12]\)).

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 find iteratively
extending words $w_1, \ldots, w_k$ such that $|S_0 \cdot (w_1 \ldots w_i)^{-1}| > |S_0 \cdot (w_1 \ldots w_{i-1})^{-1}|$ for $1 \leq i \leq k$, and where final $S_0 \cdot (w_1 \ldots w_k)^{-1} = Q$. 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 [18]. A polynomial algorithm for finding extending words in this case has been proposed in [8].

**Totally extending words and avoiding:** We study the problem whether there exists a totally extending word (Problems 2, 4, 6, 8, 10, 13). The question about existence of a totally extending word is equivalent to the question about 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
Table 2. Computational complexity of decision problems in classes of automata: given an automaton \( \mathcal{A} = (Q, \Sigma, \delta) \) with \( n \) states, a subset \( S \subseteq Q \), and an integer \( \ell \), is there a word \( w \in \Sigma^* \) of length \( \leq \ell \) such that:

| Subclass of automata                                      | All automata | Strongly connected | Synchronizing | Str. con. and synch. |
|----------------------------------------------------------|--------------|--------------------|--------------|-----------------------|
| \(|S \cdot w| = 1\)                                        | PSPACE-c     | NP-c               | NP-c         |                       |
| (reset word)                                             | 24, 34       | 13                 | 35           |                       |
| \(|S \cdot w| < |S|\)                                    | O(|\Sigma|n^2)| O(|\Sigma|n^2)     | O(|\Sigma|n^2) |                       |
| (compressing word)                                       | 13           | 13                 | 13           |                       |
| \(|S \cdot w^{-1}| \geq |S|\)                             | PSPACE-c     | PSPACE-c           | NP-c         |                       |
| (Problem 4)                                              | Subsec. 2.1  | Subsec. 2.1        | Cor. 14      |                       |
| \(|S \cdot w^{-1}| = |Q|\)                                | PSPACE-c     | NP-c               | NP-c         |                       |
| (Problem 5)                                              | Subsec. 2.1  | Cor. 14            | Cor. 14      |                       |
| \(|S \cdot w^{-1}| \geq |S|, |S| \leq k\)                              | O(|\Sigma|n^k)| O(|\Sigma|n^k)     | O(|\Sigma|n^k) |                       |
| (Problem 6)                                              | Prop. 7      | Prop. 7            | Prop. 7      |                       |
| \(|S \cdot w^{-1}| = |Q|, |S| \leq k\)                              | NP-c         | NP-c               | NP-c         |                       |
| (Problem 7)                                              | Prop. 9      | Prop. 9            | Prop. 9      |                       |
| \(|S \cdot w^{-1}| \geq |S|, |S| \geq n - k\)                        | PSPACE-c     | Open               | PSPACE-c     | NP-c                  |
| (Problem 12)                                             | Thm. 10      | Thm. 10            | Thm. 10      | Cor. 14               |
| \(|S \cdot w^{-1}| = |Q|, |S| \geq n - k\)                        | NP-c         | NP-c               | NP-c         |                       |
| (Problem 13)                                             | Cor. 14      | Cor. 14            | Cor. 14      |                       |
| \(|S \cdot w^{-1}| = |Q|, |S| = n - 1\)                               | NP-c         | NP-c               | NP-c         |                       |
| (Problem 14)                                             | Thm. 13      | Thm. 13            | Thm. 13      |                       |
| \(|S \cdot w^{-1}| \neq |S|\)                                    | O(|\Sigma|n^3)| O(|\Sigma|n^3)     | O(|\Sigma|n^3) |                       |
| (Problem 16)                                             | Thm. 15      | Thm. 15            | Thm. 15      |                       |

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 being in them. A quadratic upper bound on the length of the shortest avoiding words of a single state have been established in [28], 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 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).
Problem 2 in several proofs (e.g. [8, 18, 26]).

6 COMPLEXITY OF PREIMAGE PROBLEMS FOR DETERMINISTIC FINITE AUTOMATA

Problem 1 is strongly connected. All the hardness results hold also in the case of a binary alphabet.

words. For the cases where a polynomial algorithm exists, we put the time complexity of the best

in Section 3 and Section 4 we consider parameterized complexity by the size of the given subset.

and binary automata. Also, we consider the problems where an upper bound on the length of the

enough to find either one. For instance, this is used in the so-called averaging trick, which appear

in several proofs (e.g. [8, 18, 26]).

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 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 \( \mathcal{A} = (Q, \Sigma, \delta) \) and a subset \( S \subseteq Q \), is \( S \) extensible?

Problem 2 (Totally extensible subset). Given \( \mathcal{A} = (Q, \Sigma, \delta) \) 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 \( \mathcal{A} \) 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 \( S \cdot w^{-1} \), where \( u \) is the current suffix of \( w \). The

current subset can be stored in \( O(n) \), and since there are \( 2^n \) different subsets, \( |w| \leq 2^n \) and the

current length also can be stored in \( O(n) \). By Savitch’s theorem, the problems are in PSPACE.

To show PSPACE-hardness, we are going to reduce from the problem of determining whether

an intersection of regular languages given as DFAs is non-empty.

Let \( (\mathcal{D}_i)_{i \in \{1, \ldots, m\}} \) 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, \( \Sigma \) is the common alphabet, \( \delta_i \) is the

transition function, \( s_i \) is the initial state, and \( F_i \) is the set of final states. The problem if there

exists a word accepted by all \( \mathcal{D}_1, \ldots, \mathcal{D}_m \) (the intersection of \( L_i \) is non-empty) is a well known

PSPACE-complete problem, called Finite Automata Intersection [20]. We can assume that the

DFAs are minimal; in particular, they do not have unreachable states from the initial state, as

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 semiautoma-
ton \( \mathcal{D}' = (Q', \Sigma', \delta') \) and define \( S \subseteq Q' \) as an instance of our both problems. The scheme of the

automaton is shown in Fig. 2:

- For \( i \in \{0, 1, \ldots, m\} \), let \( \Gamma_i = \{f_i\} \times \{0, \ldots, 2M - 1\} \) 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 \( Q' = \bigcup_{i=0}^m Q'_i \).
\[ Q_0' \]

\[
\begin{array}{c}
\Sigma \\
\alpha \\
\beta \\
\end{array} \quad \begin{array}{c}
s_0 \quad f_0, 0 \\
\beta \quad f_0, 1 \\
\beta \quad f_0, 2M-1 \\
\end{array} \quad \begin{array}{c}
\Gamma_0 \\
\Sigma \\
t_0 \\
\end{array} \quad \begin{array}{c}
\Sigma, \beta \\
\end{array}
\]

\[ Q_1' \]

\[
\begin{array}{c}
\Sigma \\
\alpha \\
\beta \\
\end{array} \quad \begin{array}{c}
s_1 \quad f_1, 0 \\
\beta \quad f_1, 1 \\
\beta \quad f_1, 2M-1 \\
\end{array} \quad \begin{array}{c}
\Gamma_1 \\
\Sigma \\
\end{array} \quad \begin{array}{c}
\Sigma, \beta \\
\end{array}
\]

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

- $\Sigma' = \Sigma \cup \{\alpha, \beta\}$, where $\alpha$ and $\beta$ are fresh letters.
- $\delta'$ 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
    \[
    \delta'(t_0, a) = t_0, \quad \delta'(s_0, a) = s_0.
    \]
Hence, \( w \) contradicts the previous paragraph. Also, only outgoing edges from this subset are labeled by \( \Gamma \) so then if it contains a letter \( \Gamma \) cyclically on \( a \) for \( i \in \{0, \ldots, m\} \) and \( a \in \Sigma \), we have

\[
\delta'(\langle f_0, k \rangle, a) = t_0,
\]

\[
\delta'(\langle f_i, k \rangle, a) = \begin{cases} 
\delta_i(f_i, a) & \text{if } \delta_i(f_i, a) \neq f_i, \\
(f_i, 0) & \text{otherwise}.
\end{cases}
\]

- For \( q \in Q'_i \), we have

\[
\delta'(q, \alpha) = s_{(i+1) \mod (m+1)}.
\]

- For \( i \in \{0, \ldots, m\} \) and \( k \in \{0, \ldots, 2M - 1\} \), we have

\[
\delta'((f_i, k), \alpha) = (f_i, k + 1 \mod 2M).
\]

- We have

\[
\delta'(s_0, \alpha) = (f_0, 0).
\]

- For the remaining states \( q \in Q' \setminus (\bigcup_{i=0}^{m} \Gamma_i \cup \{s_0\}) \), we have

\[
\delta'(q, \beta) = q.
\]

\bullet \text{ The subset } S \subseteq Q' \text{ is } \[ S = \left( \bigcup_{i=1}^{m} F_i \cap Q' \right) \cup \bigcup_{i=0}^{m} \Gamma_i \cup \{s_0\}. \]

It is easy to observe that \( D' \) is strongly connected. Get any \( i, j \in \{0, \ldots, m\} \). We show how to reach any state \( q \in Q'_i \) from a state \( p \in Q'_j \). First, we can reach \( s_j \) by \( a^{(m+1+j-i) \mod (m+1)} \). For \( j \geq 1 \), each state \( q \in Q'_j \setminus (\Gamma_j \setminus \{(f_j, 0)\}) \) is reachable from \( s_j \), since \( \delta' \) restricted to \( \Sigma \) acts on \( Q'_j \) as \( \delta_j \) on \( Q_j \) (with \( f_j \) replaced by \( (f_j, 0) \)) and \( D_j \) is minimal. For \( j = 0 \), states \( (f_0, 0) \) and \( t_0 \) are reachable from \( s_0 \) by the transformations of \( \beta \) and \( \beta a \) respectively, for any \( a \in \Sigma \). States \( q \in \Gamma_j \) can be reached from \( (f_j, 0) \) using \( \delta_\beta \).

We will show the following statements:

1. If \( S \) is extensible in \( D' \), 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 \( D' \).

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 \( i \in \{0, \ldots, m\} \), if \( (S \cdot w^{-1}) \cap \Gamma_i \neq \emptyset \), then \( (S \cdot w^{-1}) \cap \Gamma_i = \Gamma_i \).

This follows by induction since: the empty word possesses this property; the transformation \( \delta_a \) for \( a \in \Sigma \setminus \{\beta\} \) maps every state from \( \Gamma_i \) to the same state, so it preserves the property; \( \delta_\beta \) acts cyclically on \( \Gamma_i \) 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 \( i \in \{0, \ldots, m\} \), 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 \( \alpha \). For a contradiction, if \( w \in (\Sigma' \setminus \{\alpha\})^* \), then if it contains a letter \( \alpha \in \Sigma \), 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 \( \alpha \), \( t_0 \notin S \), \( \Gamma_0 \cdot \beta^{-1} = \Gamma_0 \), and \( \Gamma_0 \cdot a^{-1} = \emptyset \). This contradicts the previous paragraph. Also, \( w \) cannot be of the form \( \beta^k \), for \( k \in \mathbb{N} \), since \( S \cdot \beta^k = S \). Hence, \( w = w_p \omega w_s \), where \( w_p \in (\Sigma')^* \) and \( w_s \in (\Sigma' \setminus \{\alpha\}) \).
Theorem 4. Given an automaton \( \mathcal{A} = (Q, \Sigma, \delta) \) and a subset \( S \subseteq Q \), we can construct in polynomial-time a binary automaton \( \mathcal{A}' = (Q', \{a', b'\}, \delta') \) and a subset \( S' \subseteq Q' \) such that:

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

Proof. We can assume that \( \Sigma = \{a_0, \ldots, a_{k-1}\} \). We construct \( \mathcal{A}' = (Q', \{a', b'\}, \delta') \) with \( Q' = Q \times \Sigma \) and \( \delta' \) defined as follows: \( \delta'((q, a_i), a') = \delta(q, a_i), a_1 \), and \( \delta'((q, a_i), b') = (q, a_{i+1} \mod k) \). Clearly, \( \mathcal{A}' \) can be constructed in \( O(nk) \) time, where \( k = |\Sigma| \).

(1) Suppose that \( \mathcal{A} \) is strongly connected; we will show that \( \mathcal{A}' \) is also strongly connected. Let \( (q_1, a_i) \) and \( (q_2, a_j) \) be any two states of \( \mathcal{A}' \). In \( \mathcal{A} \), 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 \( \mathcal{A}' \) 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 \( \mathcal{A}' \) 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

\[
w' = (b')^{h_1} a' \cdots (b')^{h_m-1} a' (b')^{h_m},
\]

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

(2) and (3): For \( i \in \{0, \ldots, k-1\} \) we define \( U_i = (Q \times \{a_i\}) \). 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 \( \mathcal{A} \) 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 \( \mathcal{A}' \) 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.

**Proposition 5.** When the automaton is binary and synchronizing, Problem \( \mathcal{A} \) remains \( \text{PSPACE-complete} \).

**Proof.** We just add a sink state \( z \) and a letter which synchronizes \( \mathcal{A} = (Q, \Sigma, \delta) \) to \( z \). Additionally, a standard tree-like binarization is suitably used to obtain a binary automaton.

Formally, we construct a binary automaton \( \mathcal{A}' \). We can assume that \( \Sigma = \{a, b\} \), and \( Q = \{q_1, \ldots, q_n\} \). Let \( z \) be a fresh state. Let \( Q' = Q \cup \{q'_1, \ldots, q'_n\} \cup \{q''_1, \ldots, q''_n\} \). We construct \( \mathcal{A}' = (Q' \cup \{z\}, \Sigma \cup \{a_z\}, \delta') \), where \( \delta' \) for all \( i \) is defined as follows: \( \delta'(q_i, a) = q_i', \delta'(q_i, b) = q_i'' \), \( \delta'(q_i', a) = \delta(q_i, a) \), \( \delta'(q_i'', a) = \delta(q_i, b) \), \( \delta'(q_i', b) = z \), and \( \delta'(z, a) = \delta'(z, b) = z \). Then \( S \subseteq Q \) is extensible in \( \mathcal{A}' \) if and only if it is extensible in \( \mathcal{A} \). □

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

**Proof.** Since \( \mathcal{A} \) is synchronizing, Problem \( \mathcal{A} \) 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 [29].

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

For \( \text{NL-hardness} \), 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 \( \mathcal{A}' = (V, \Sigma, \delta) \) 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 output degree of \( G \), and set \( \Sigma = \Sigma' \cup \{\alpha\} \), where \( |\Sigma'| \) is equal to the maximum output degree. We output \( \mathcal{A}' \) such that for every \( q \in V \), every edge \( (q, p) \in E \) is colored by a different letter from \( \Sigma' \). If there is no outgoing edge from \( q \), then we set the transitions of all letters from \( \Sigma' \) to be loops. If the output 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 \( \Sigma' \). The produced automaton \( \mathcal{A}' \) is synchronizing just by \( \alpha \).

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 \( \alpha \). Since \( \alpha^{-1} \) 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.

2.1. Bounded 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 \( \mathcal{A} = (Q, \Sigma, \delta) \), a subset \( S \subseteq Q \), and an integer \( \ell \), is \( S \) extensible by a word of length at most \( \ell \)?

**Problem 4** (Totally extensible subset by short word). Given \( \mathcal{A} = (Q, \Sigma, \delta) \), a subset \( S \subseteq Q \), and an integer \( \ell \), is there a word \( w \in \Sigma^* \) such that \( S \cdot w^{-1} = Q \) of length at most \( \ell \)?

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 way 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 like in the general case.

3. Extending small subsets

The complexity of extending problems rely on the 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 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 5** (Extensible small subset). For a fixed \( k \in \mathbb{N} \), given \( \mathcal{A} = (Q, \Sigma, \delta) \) and a subset \( S \subseteq Q \) with \( |S| \leq k \), is there a word extending \( S \)?

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

**Proof.** We build the \( k \)-subsets automaton \( \mathcal{A}^{\leq k} = (Q^{\leq k}, \Sigma, \delta^{\leq k}, S_0, F) \), where \( Q^{\leq k} = \{ A \subseteq Q : |A| \leq k \} \) and \( \delta^{\leq k} \) is naturally defined by the image of \( \delta \) on a subset. Let the set of initial states be \( I = \{ A \in Q^{\leq k} : |A \cdot a^{-1}| > |S| \text{ for some } a \in \Sigma \} \), 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 \( \mathcal{A} \). We can simply check this condition by a BFS. The size (number of states and edges) of this automaton is bounded by \( O(|\Sigma|n^k) \), so the procedure takes this time. \( \Box \)

**Problem 6** (Totally extensible small subset). For a fixed \( k \in \mathbb{N} \), given \( \mathcal{A} = (Q, \Sigma, \delta) \) and a subset \( S \subseteq Q \) with \( |S| \leq k \), 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 \( O(|\Sigma|n^2) \) time. For larger \( k \) we have the following:

**Proposition 8.** Problem 6 can be solved in \( O(\max(|\Sigma|n^3, |\Sigma|n^k)) \) time.

**Proof.** Let \( u \) be a word of the minimal rank in \( \mathcal{A} \). We can find such a word in \( O(|\Sigma|n^3) \) time, using e.g. the algorithm from [13].

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
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 $A$. Otherwise, $S$ is not totally extensible. Reachability can be checked in at most $O(n^k)$ time. However, if the rank $r$ of $u$ is less than $k$, the algorithm takes only $O(n^r)$ time. □

### 3.1. Bounded 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 $k \in \mathbb{N}$, given $A = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$ with $|S| \leq k$, and an integer $\ell$, is there a word extending $S$ of length at most $\ell$?

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

**Problem 8 (Totally extensible small subset by short word).** For a fixed $k \in \mathbb{N}$, given $A = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|S| \leq k$, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$ of length at most $\ell$?

**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 $O(\min(n^3, n^k))$ (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 [35] 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 $\ell$ 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 [35, Lemma 2.4]). Therefore, we can set $S = \{s_2\}$, and thus Problem 8 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 9 (Extensible large subset).** For a fixed $k \in \mathbb{N}$, given $A = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \leq k$, is there a word extending $S$?

**Problem 10 (Totally extensible large subset).** For a fixed $k \in \mathbb{N}$, given $A = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$ with $|Q \setminus S| \leq k$, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$?

Problem 10 is equivalent to deciding the existence of an avoiding word for a subset $S$ of size $\leq k$. 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 $k \geq 2$. We give a special attention to this problem and study it separately.

**Problem 11 (Avoidable state).** Given $A = (Q, \Sigma, \delta)$ 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 $k \geq 2$, although the problem remains open for strongly connected automata.
**Theorem 10.** Problem \( \varphi \) is PSPACE-hard for every \( k \geq 2 \) and \( |\Sigma| \geq 2 \) even if the given automaton is synchronizing.

**Proof.** We show a reduction from PSPACE-complete Problem \( \varphi \) to Problem \( \varphi' \). Let \( \mathcal{A} = (Q, \Sigma, \delta) \) and \( S \subseteq Q \) be an instance of Problem \( \varphi \). We construct an automaton \( \mathcal{A}' = (Q', \Sigma', \delta') \), where \( e, s, \alpha \) are fresh symbols. Let \( f \) be an arbitrary state from \( Q \). We define \( \delta' \) 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 \( q \in S \cup \{ s \} \);
4. \( \delta'(q,a) = e \) for \( q \in (Q \cup \{ e \}) \setminus S \).

We define \( S' = Q \). Note that \( |Q' \setminus S'| = 2 \), and hence automaton \( \mathcal{A}' \) with \( S' \) is an instance of Problem \( \varphi' \). We will show that \( S' \) is extensible in \( \mathcal{A}' \) if and only if \( S \) is totally extensible in \( \mathcal{A} \).

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

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

Note that \( \delta' \) is synchronizing, if we get strongly connected \( \mathcal{A} \). This case there is some word \( \alpha \), which maps state \( f \) to some state not from \( S \) in \( \mathcal{A} \), hence \( Q' \cdot \alpha w \alpha = \{ f, e \} \cdot w \alpha = \{ e \} \).

Now, we show that we can reduce the alphabet to two letters. Consider the application of the Theorem \( \varphi \) to Problem \( \varphi' \). 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 \( (e,a) \) for \( a \in \Sigma \) in the obtained binary automaton to one sink state \( e'' \). In this way we get a synchronizing binary automaton (since \( \mathcal{A}' \) is synchronizing). The extending words remain the same, since the identified state \( e'' \) is not reversely reachable from \( S'' \), and \( e'' \) is not contained in the subset \( S'' \). \( \square \)

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

**Theorem 11.** Let \( \mathcal{A} = (Q, \Sigma, \delta) \) 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 \( p \in S \cdot w w' \). Moreover, if \( q \in S \) then \( |S \cdot w w'| < |S| \), because \( \{ q, p \} \cdot w w' = \{ p \} \). If \( q \) is not avoidable, then all subsets \( Q \cdot (w w'), Q \cdot (w w')^2, \ldots \) 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 minimal rank in \( \mathcal{A} \), and \( v \) be a word that avoids \( q \). Then \( w = w v \) 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 \(|(Q \cdot w) \cap [p]_\sim| \leq 1\), 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 \(A\) is not strongly connected, then every state from a strongly connected component that is not a sink can be avoided. If a state belong 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 [13] computing the pair automaton and performing a breadth-first search with reversed edges on the pairs of states. It works in \(O(|\Sigma|^2)\) time and \(O(n^2 + |\Sigma|n)\) 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 \(O(|\Sigma|)\) 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, it follows that Problem 10 for \(k \geq 2\) can be solved in polynomial time.

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

**Proof.** Similarly to the proof of Theorem 11, let \(u\) be a word of the minimal rank \(r\) in \(A\) 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 criteria: \(S\) is avoidable if and only if we can find 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 there exists a word \(w'\) avoiding \(S\) as in the theorem. Then the word \(w = uu'\) 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 each state \(s \in S\), we have \(q_{[s]_\sim} \in [s]_\sim \setminus S\), because \(w\) avoids \(S\) and \(q_{[s]_\sim} \in Q \cdot u\). Notice that \([s]_\sim \cap S\) can contain more than one state; however, for every \(x, y \in [s]_\sim \cap S\) we have the same \(q_{[x]_\sim} = q_{[y]_\sim}\). We conclude that there exists a subset \(Q' \subseteq Q \cdot u\) of size \(|S/\sim|\) such that \(Q' \cdot w' = \{q_{[s]_\sim} | s \in S\}\). It is obvious that \(\{q_{[s]_\sim} | s \in S\} \cap S = \emptyset\) and \(|\{q_{[s]_\sim} | s \in S\}| = |S/\sim|\). Therefore, if \(S\) is avoidable, then we can find 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 \(s \in S\).

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 \(Q \cdot uu'\) there is exactly one state
in each equivalence class, we have \(((Q \cdot u) \setminus Q') \cdot w' \subseteq Q \setminus \bigcup_{s \in S}([s]_{\sim}),\) and by the assumption \((Q' \cdot w') \cap S = \emptyset.\) Therefore, we get that \(uw'\) is an avoiding word for \(S.\)

These criteria give us Alg. \([\ref{alg:avoid}],\) to find \(w\) or verify that \(S\) cannot be avoided.

**Algorithm 1** Avoiding a subset.

*Require:* Automaton \(\mathcal{A}(Q, \Sigma, \delta)\) and a subset \(S \subseteq Q.\)

1. Find a word \(u\) of 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
5. end for
8. return “\(S\) is unavoidable”.

Alg. \([\ref{alg:avoid}],\) first finds a word \(u\) of minimal rank. This can be done by iterative compressing the subset as long as possible by the algorithm from \([\ref{thm:compress}],\) which works in \(O(n^3 + |\Sigma|n^2)\) time and \(O(n^2 + |\Sigma|n)\) space. 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 \(\mathcal{A}^z(Q^z, \Sigma, \delta^z),\) where \(\delta^z\) is \(\delta\) 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 \(\binom{|\Sigma|}{z} \in O(n^z).\) Also note that we have to visit every \(z\)-tuple only once during a run of the algorithm, and we can store it in \(O(n^z + |\Sigma|n)\) space. Therefore, the algorithm works in \(O(n^3 + |\Sigma|(n^2 + n^z))\) time and \(O(n^2 + n^z + |\Sigma|n)\) space.

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

\(\square\)

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

**Problem 12** (Extensible large subset by short word). For a fixed \(k \in \mathbb{N},\) given \(\mathcal{A} = (Q, \Sigma, \delta),\) a subset \(S \subseteq Q\) with \(|Q \setminus S| \leq k,\) and an integer \(\ell,\) is there a word extending \(S\) of length at most \(\ell?\)

**Problem 13** (Totally extensible large subset by short word). For a fixed \(k \in \mathbb{N},\) given \(\mathcal{A} = (Q, \Sigma, \delta),\) a subset \(S \subseteq Q\) with \(|Q \setminus S| \leq k,\) and an integer \(\ell,\) is there a word \(w \in \Sigma^*\) such that \(S \cdot w^{-1} = Q\) of length at most \(\ell?\)

As before, both problems for \(k = 1\) are equivalent to the following:

**Problem 14** (Avoidable state by short word). Given \(\mathcal{A} = (Q, \Sigma, \delta),\) a state \(q \in Q,\) and an integer \(\ell,\) is there a word \(w \in \Sigma^*\) such that \(q \not\in Q \cdot w\) of length at most \(\ell?\)

Problem \([\ref{prob:avoid}],\) for \(k \geq 2\) obviously remains \(PSPACE\)-complete. By the following theorem, we show that Problem \([\ref{prob:avoid}],\) is \(NP\)-complete, which then implies \(NP\)-completeness of Problem \([\ref{prob:avoid}],\) for every \(k \geq 1\) (by Corollary \([\ref{cor:avoid}],\)).

**Theorem 13.** Problem \([\ref{prob:avoid}],\) 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 \( O(|\Sigma| n) \) time. If the state \( q \) is avoidable, then the length of the shortest avoiding words is at most \( O(n^2) \) \cite{28}. Hence, the problem is solvable in non-deterministic quadratic time.

In order to prove that the problem is NP-hard, 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.

Let us have an instance of this problem from the Eppstein’s proof of \cite{13} Theorem 8. Namely, for a given synchronizing automaton \( \mathcal{B} = (Q_\mathcal{B}, \{0, 1\}, \delta_\mathcal{B}) \) 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, \( \mathcal{B} \) is an automaton with a sink state \( z \in Q_\mathcal{B} \), and there are two subsets \( S = \{s_1, \ldots, s_d\} \) and \( F \subseteq Q_\mathcal{B} \) with the following properties:

1. Each state \( q \in Q_\mathcal{B} \setminus S \) is reachable from a state \( s \in S \) through a (directed) path in the underlying digraph of \( \mathcal{B} \).
2. For each state \( s \in S \) and each word \( w \) of length \( m \), we have \( \delta_\mathcal{B}(s, w) \in F \cup \{z\} \).
3. For each \( f \in F \) we have \( \delta_\mathcal{B}(f, 0) = \delta_\mathcal{B}(f, 1) = z \).
4. For each state \( s \in S \) and a non-empty word \( w \in \{0, 1\}^< m \), we have \( \delta_\mathcal{B}(s, w) \notin (F \cup S) \).

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

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

Secondly, we add a set of states \( S' = \{s'_1, \ldots, s'_d\} \) of size \( d = |S| \) and a letter 2 to make the automaton strongly connected. Letters 0, 1 map \( S' \) to the corresponding states from \( S \), that is, \( \delta_{\mathcal{A}'}(s'_1, 0) = \delta_{\mathcal{A}'}(s'_1, 1) = s_1 \in S \). Letter 2 connects states \( r_0, s'_1, s'_2, \ldots, s'_d \) into one cycle, i.e.

\[
\delta_{\mathcal{A}'}(r_0, 2) = s'_1, \quad \delta_{\mathcal{A}'}(s'_1, 2) = s'_2, \quad \ldots, \quad \delta_{\mathcal{A}'}(s'_{d-1}, 2) = s'_d, \quad \delta_{\mathcal{A}'}(s'_d, 2) = r_0.
\]

We also set \( \delta_{\mathcal{A}'}(s'_d, 2) = r_1 \) and all the other transitions of 2 we define equal to the transitions of 0.

Finally, we transform \( \mathcal{A}' \) to the final automaton \( \mathcal{A}'(Q, \{a, b\}, \delta) \). We encode letters 0, 1, 2 by 2-letter words over \( \{a, b\} \) alike it was done in \cite{6}. Namely, for each state \( q \in Q_{\mathcal{A}'} \setminus (F \cup \{z\}) \), we add 2 new states \( q^a, q^b \) and define their transitions as follows:

\[
\delta(q, a) = q^a, \quad \delta(q^a, a) = \delta(q^a, b) = \delta_{\mathcal{A}'}(q, 0),
\]

\[
\delta(q, b) = q^b, \quad \delta(q^b, a) = \delta_{\mathcal{A}'}(q, 1), \quad \delta(q^b, b) = \delta_{\mathcal{A}'}(q, 2).
\]

Then, \( aa, ab \) correspond to applying 0, \( ba \) corresponds to applying 1, and \( bb \) corresponds to applying 2. Denote this encoding function by \( \phi \), i.e. \( \phi(0) = aa, \phi(1) = ba, \phi(2) = bb \). We also extend \( \phi \) to words over \( \{0, 1, 2\}^* \) in a similar way.

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 \( \mathcal{A} \) is shown in Fig. \[3\].

Observe that \( \mathcal{A} \) is strongly connected: \( z \) is reachable from each state, from \( z \) we can reach \( r_0 \) by 0, from \( r_0 \) we can reach every state from \( S' \) by applying a power of letter 2, and we can reach every state of \( S \) from the corresponding state from \( S' \). Then every state from \( Q_\mathcal{B} \) is reachable from
a state from $S$ by Property 1. It follows that $\mathcal{A}$ is also strongly connected, since for every $q \in Q_{\mathcal{A}'}$, every state from $\phi(q)$ is reachable from $q$.

Observe that $\mathcal{A}$ is synchronizing: We claim that $a^{4m+6}$ is a reset word for $\mathcal{A'}$. Indeed, $aa$ does not map any state into $\phi(S')$. Every word of length $m+1$ is reset for $\mathcal{B}$ and synchronizes to $z$, in particular, $0^m1$. Since $\phi(0^m1) = a^{2m+2}$ 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 $\mathcal{B}$ has a reset word of length $m$ is equivalent to determining whether $z$ can be avoided in $\mathcal{A}$ by a word of length at most $2m + 3$. 

---

**Figure 3.** The automaton $\mathcal{A}$ obtained from $\mathcal{A'}$ in the proof of Theorem 13. Here every state $q$ represents $\phi(q)$, and we have $0$: $aa, ab$, $1$: $ba$, and $2$: $bb$. 
Suppose that \( \mathcal{B} \) has a reset word \( w \) of length \( m \), and consider \( u = \phi(0w)b \). Note that \( \phi(0) = aa \) does not map any state into \( \phi(S') \) nor into \( \phi(r_0) \). Hence, we have
\[
\delta(Q, \phi(0)) \subseteq \phi(Q_{\mathcal{B}}) \cup \phi(R \setminus \{ r_0 \}).
\]
Due to the definition of \( \phi \), factor \( bbb \) cannot appear in the image of words from \( \{0,1\}^* \) by \( \phi \). 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 \( \mathcal{B} \) that maps every state from \( Q_{\mathcal{B}} \) to \( z \), we have
\[
\delta(\phi(Q_{\mathcal{B}}), \phi(w)) \subseteq \{ z \} \cup \phi(R \setminus \{ r_m \}).
\]
By the definition of the transitions on \( R \cup \{ z \} \) (only \( \phi(2) \) 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 \( \delta(\{ z \} \cup \phi(R \setminus \{ r_m \}), b) \subseteq R \), thus \( u \) avoids \( z \).

Let us 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 \( \phi \) with respect to the definition on \( \mathcal{A}' \), that is, \( \tilde{\phi}^{-1}(aa) = \tilde{\phi}^{-1}(ab) = 0 \), \( \tilde{\phi}^{-1}(ba) = 1 \), \( \tilde{\phi}^{-1}(bb) = 2 \), which is extended to words of even length by concatenation.

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

If the second letter of \( \tilde{\phi}^{-1}(u) = 2 \), then \( s_0 \) goes to \( r_1 \) and we get \( \{ r_1, r_2, \ldots, r_m, z \} \) 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 \( z \). Thus it cannot be the case and first two letters of \( \tilde{\phi}^{-1}(u) \) are either \( 0 \) or \( 1 \).

By Property 2 of \( \mathcal{B} \), every zero-one word of length \( m \) maps \( s \in S \) into \( \{ z \} \cup F \). Since the letter \( 2 \) acts like \( 0 \) on \( Q_{\mathcal{B}} \setminus S \) in \( \mathcal{A}' \) and \( \tilde{\phi}^{-1}(u'') \) starts with \( 0 \) or \( 1 \), \( u'' \) 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 \( 2 \)-s replaced with \( 0 \)-s must be a reset word for \( \mathcal{B} \).

As a corollary from Theorem 13 and Theorem 12 we complete the results.

**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 \( \ell \) 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 \( n^2 \) [32]. 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
Therefore, if the given $\ell$ is larger than this bound, we answer positively. Otherwise, we nondeterministically guess a word of length at most $\ell$ 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 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. $\square$

5. Resizing a Subset

In this section we deal with the following two problems:

**Problem 15** (Resizable subset). Given an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$?

**Problem 16** (Resizable subset by short word). Given an automaton $\mathcal{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and an integer $\ell$, is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$ of length at most $\ell$?

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 15.** Given an automaton $\mathcal{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, there exists an algorithm working in $O(|\Sigma|n^3)$ time (assuming constant time arithmetic of integers of order $O(2^n)$) 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 at most $n - 1$.

**Proof.** We construct a reduction to the problem of multiplicity equivalence of NFAs and apply the algorithm from [31] with an improvement to achieve the desired complexity.

Let $N$ be the NFA obtained by reversing the edges of $\mathcal{A}$. In addition, let the set of initial states be $S$ and the set of accepting states be $Q$. Now we define the second NFA $N'$ with multiple edges. It has only two states $i$ and $f$, which serve as the initial and the accepting state respectively. For each letter $a \in \Sigma$, let $i$ have $|S|$ transitions to $f$ labeled by $a$, and let $f$ have a loop labeled by $a$.

A run of a word $w$ is a path of length $|w|$ starting in an initial state and whose edges are labeled by the letters of $w$ in order. It is clear that each non-empty word is accepted by $N'$ and has exactly $|S|$ accepting runs. On the other hand, every accepting run in $N$ correspond to a path in $\mathcal{A}$ from a state $q \in Q$ to a state $q \cdot w \in S$. Therefore, there are $|S|$ accepting runs in $N$ if and only if in $\mathcal{A}$ we have $|S \cdot w^{-1}| = |S|$.

Tzeng [31] developed an algorithm running in $O(|\Sigma|n^4)$-time for multiplicity equivalence of two NFAs. It uses (not explicitly mentioned anywhere; however) the assumption of performing arithmetic calculations on large integer of size $O(2^n)$ in constant time, since they might appear during exact calculations on rational numbers (which are necessary if we care about exactness of the result). The algorithm iteratively extends an (initially empty) linear subspace by adding independent vectors that are characteristic vectors of subsets of ending points of accepting runs of a word of length $i = 0, 1, \ldots$. If at some iteration, an added vector does not belong to a certain linear subspace, this means that $N$ and $N'$ has a different number of accepting runs of the word corresponding to this vector. Furthermore, such a word must be a shortest one with this property. The algorithm performs at most $n$ iterations starting from the empty subspace, where in an $i$-th iteration we consider a word of length $i - 1$, thus the length of the found word is at most $n - 1$. Therefore, it remains to show how to implement this algorithm in $O(|\Sigma|n^3)$ time.

---

1In a previous version of our proof we presented our own algorithm having $O(|\Sigma|n^3)$ time complexity under the assumption of performing arithmetic computations in constant time. We thank one of anonymous reviewers that suggested a shortcut by reducing to multiplicity equivalence of NFAs.
The complexity of the algorithm comes from the fact that at each \( j \)-th of at most \( n \) iterations, we need to test at most \( |\Sigma| \) vectors on independence with previously added \( j-1 \) vectors. In \cite{volk16} this is done in a straightforward way, that is, by checking whether the matrix comprised of all \( j \) vectors has rank \( j \). Since the complexity of the latter is \( \mathcal{O}(n^3) \) in general, the overall complexity is \( \mathcal{O}(|\Sigma|n^4) \). However, this algorithm can be optimized to have \( \mathcal{O}(|\Sigma|n^3) \)-time complexity as follows. Most likely the same idea was meant by Volker Diekert as noticed in \cite{diekert93}, where an alternative algorithm with \( \mathcal{O}(|\Sigma|^2n^3) \)-time complexity algorithm was designed.

Anyway, we present the improvement here for the sake of completeness. To perform this subroutine efficiently, we maintain a sequence of vectors \( G \) 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 \( I = (i_1, i_2, \ldots, i_k) \) and normalized vectors \( G = \{g'_1, \ldots, g'_k\} \) have the property that for each \( j, 1 \leq j \leq k \), there is exactly one vector from \( \{g'_1, \ldots, g'_j\} \) with non-zero \( j \)-th coordinate, which is equal to 1.

We begin with the first non-zero vector \( g_1 \), which can be normalized (by multiplying by a scalar) to the vector \( g'_1 \) having the coordinate with index \( i_1 \) equal to 1. Now, suppose we are given a vector \( g \) and we have already built \( G \) of size \( k \) and the set of indices \( I = \{i_1, i_2, \ldots, i_k\} \) with aforementioned properties. Then, we just compute \( g' = g - \sum_{r=1}^{k} g(i_r) \cdot g_r \). Since \( g(i_r) \cdot g_r \) has \( g(i_r) \) at the \( i_r \)-th coordinate, all the entries at the coordinates from \( I \) in \( g' \) are zero. If there is a non-zero coordinate left in \( g' \), then take the first such coordinate, normalize \( g' \) to be 1 at this index and add to \( G \). In the opposite case, if \( g' = 0 \), then \( g \) belongs to the subspace spanned by \( G \) and thus should not be added.

Assuming that in our computational model every arithmetic operation has a unitary cost, then clearly this function can be performed in \( \mathcal{O}(kn) \)-time during a \( k \)-th call. 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.

The running time \( \mathcal{O}(|\Sigma|n^3) \) of the algorithm is quite large (and may require large arithmetic as discussed the proof), and it is an interesting open question whether there is a faster algorithm for Problems \ref{preimage:problem15} or \ref{preimage:problem16}.

We note that Problem \ref{preimage:problem15} 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 binary alphabet is drawn uniformly at random.

**Remark 16.** The algorithm from \cite{gusev11} checks in \( \mathcal{O}(n) \) time 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 have to use the algorithm from Theorem \ref{preimage:algorithm15} Thus, the overall expected time is

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

**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] K. Archangelsky. Efficient algorithm for checking multiplicity equivalence for the finite $z \cdot \sigma^*$-automata. In Masami Ito and Masafumi Toyama, editors, DLT, pages 283–289. Springer, 2003.

[4] 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.

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

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

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

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

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

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] K. Gulı̂strand 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.

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

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

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

[20] D. Kozen. Lower Bounds for Natural Proof Systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS’77, pages 254–266, 1977.

[21] P. Martyugin. Computational Complextiy 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.

[22] 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.

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

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

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

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

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

[28] 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.

[29] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
[31] 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.

[32] 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.

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

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

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