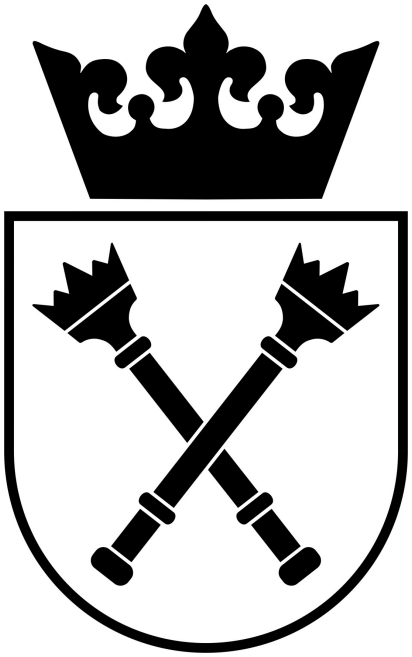
Synchronization of Finite Automata - Problems and Generalizations

Jakub Ruszil



Supervisors: dr hab. Adam Roman prof. UJ, dr Jakub Zygadło

A thesis presented for the degree of

Doctor of Philosophy

Faculty of Mathematics and Computer Science

Jagiellonian University

Cracow, 2024

Podziękowania

Chciałbym podziękować promotorom i dr. Bakalarskiemu. W pierwszej kolejności za obudzenie we mnie zainteresowania szeroko pojętą matematyką dyskretną, a także późniejszą pomoc merytoryczną i organizacyjną podczas pracy nad rozprawą.

Ponadto podziękowania za wieloletnie wsparcie należą się moim rodzicom i braciom, Zuzie, Maćkowi, Patrykowi, Grześkowi, Wojtkowi i Szymonowi.

Nie sposób nie wspomnieć w tym miejscu też o wspaniałej grupie z Mazur. Szkołęście mi dali twardą.

Contents

List of Figures

1 Introduction

1.1 Motivation

Suppose we have n cities. There are two one-way roads (denoted as a and b) leading from each city to another city or back to the city it starts in (forming a loop). Furthermore, let us assume we have a map with cities and roads between them labelled with all necessary symbols (a, b) (that is, a finite automaton) but we do not know the city we are in right now. The question arises whether we can determine our position using only the map and movement between the cities.

It turns out that if the labelled digraph with n vertices that represents a map has the so-called synchronization property, then the answer to the question is positive - there exists a sequence of roads (i.e., a word s over the letters a, b) with a property that, no matter which city we start our trip in, we always finish in the same city while traveling according to the sequence s. What is more, we know which city we are finishing in.

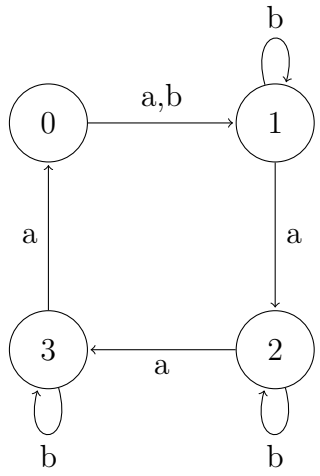


Figure 1: The automaton $\mathcal{C}_4$

Consider an example depicted in Figure 1. It shows four cities, each with two outbound roads labelled a and b. It is easy to see that, regardless of which city we start

from, following the sequence *baaabaaab* will always lead us to city 1, demonstrating that the automaton accepts a synchronizing word *baaabaaab*. Transitions by the word *baaabaaab* from all states are depicted in Fig. 2.

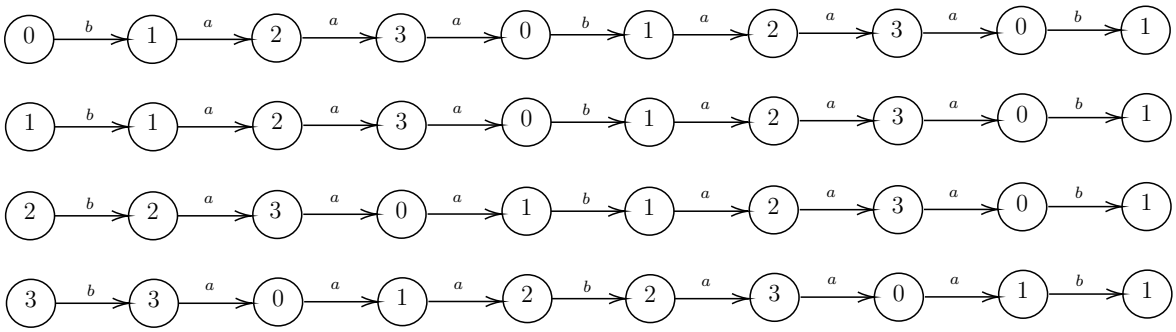


Figure 2: Transitions by the word *baaabaaab* in $\mathcal{C}_4$

The concept of synchronization dates back to at least the 1950s [18, 35] and has garnered considerable attention from theoretical computer scientists due to the famous Černý Conjecture. This conjecture states that if an n-state automaton is synchronizing, then it must have a synchronizing word of length at most $(n-1)^2$. Despite being open for over 60 years, the conjecture was proven for various classes of automata. Synchronization of finite automata also has numerous practical applications, some of which are discussed in Section 2.4. We introduce some necessary preliminary definitions and results in the next subsection.

1.2 Aims of the thesis

The aims of this thesis are as follows:

1. Investigate the impact of the alphabet size on the length of carefully synchronizing words (the analogue of synchronizing words for partial finite automata),

2. Introduce new non-trivial classes of deterministic finite automata (DFAs) that have a quadratic upper bound for the length of the shortest synchronizing word,

3. Extend the notion of one-cluster automata (investigated in [49] and [4]) to partial finite automata, examine the extremal properties of careful synchronizing words for such automata, and address complexity issues related to determining whether such automata are carefully synchronizing,

4. Explore the possibility of designing an asymmetric cryptosystem that utilizes the notion of synchronizability.

Understanding how the size of the alphabet affects the length of carefully synchronizing words is crucial for optimizing the design and efficiency of automata-based systems. It provides insights into the behavior of finite automata under different constraints, which can inform the development of algorithms and applications in various fields. Note that synchronization under constraints was studied earlier for example by Fernau et al. in [15].

Introducing new classes of DFAs with long shortest synchronizing words can contribute to expanding the theoretical understanding of automata theory. For example Pin-Černý Conjecture [37] has been disproved by a family of automata with quadratic length of the shortest synchronizing words in [26]. Moreover such classes may be used to design new test cases for existing algorithms. Such examples are "the hardest" to test amongst all because they concern very long words. Other examples of extremal classes of synchronizing automata can be found in literature, for example [21, 19].

Extending the concept of one-cluster automata to partial finite automata allows for exploring new mathematical structures and properties, which can deepen our understanding of automata theory. Examining the extremal properties of careful synchronizing words in this context sheds light on fundamental properties of automata and their behavior. Addressing complexity issues related to determining whether such automata are carefully synchronizing is important for practical applications, as it informs the feasibility and efficiency of automata-based systems in real-world scenarios.

Exploring the intersection of automata theory and cryptography opens up new

avenues for designing secure and efficient cryptographic systems. Leveraging the notion of synchronizability in cryptographic protocols introduces novel approaches for enhancing security and efficiency in asymmetric encryption schemes. Research in this area has the potential to advance the state-of-the-art in cryptography by providing alternative cryptographic primitives and protocols with strong theoretical foundations.

1.3 Arrangement of the thesis

Section 2 introduces necessary preliminaries used in the latter sections of thesis, discusses the notion of synchronization and its generalizations as well as summarizes the most important applications of it.

We address the problem stated in Point 1 in Section 3. The results from that section are included in [40] and were presented at the CIAA 2022 conferene in Rouen.

Section 4 is devoted to defining the class of deterministic finite automata (DFA) with coinciding cycles and proving the quadratic upper bounds for the length of the shortest synchronizing words for such automata, thus exploring Point 2. These results are published in [41] and were presented at the DLT 2023 conference in Umeå.

In Section 5 we inquire into the problem of careful synchronizability of one-cluster partial finite automata (PFA). We managed to show examples of one-cluster automata having the shortest carefully synchronizing word of exponential length (all other known examples of such extremal families had more than one cluster) and proved that determining if a given one-cluster PFA is carefully synchronizing is NP-hard even for a two-letter alphabet. These results are published in [42], were presented at the DLT 2024 conference in Göttingen, and cover Point 3.

Finally, we investigate the problem of developing a public-private key cryptosystem that grounds its security (even partially) on the problem of careful synchronization in Section 6. We describe the scheme of ciphering and deciphering, attempt to accomplish initial cryptanalysis of the system, as well as discuss the problem of

defining and generating the key for encryption and decryption.

2 Synchronization

2.1 Preliminaries

We define an *alphabet* as a finite non-empty set. Elements of an alphabet are called *letters*. A *word* over the alphabet Σ is a finite sequence of letters from Σ. If a word w equals $(x_1, \ldots, x_n)$ then we omit brackets and commas and write $w = x_1 \ldots x_n$. For a word w we denote $|w|$ as the length of w. If w is the empty word then we denote it as ϵ. A *concatenation* of words $w = x_1 \ldots x_n$ and $v = y_1 \ldots y_m$, written as wv, is the word $z = x_1 \ldots x_n y_1 \ldots y_m$. In case of a k-folded concatenation of a word w with itself we write w^k. We say that a word v is a *prefix* of the word w if there exist a word z such that $w = vz$. For an alphabet Σ and $k \in \mathbb{N} \cup \{0\}$ we define $\Sigma^k = \{x_1 \ldots x_k : x_i \in \Sigma\}$ (the set of words of lengths exactly k over Σ). Finally we define $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$.

A *Deterministic finite automaton* (DFA) is an ordered tuple $\mathcal{A} = (Q, \Sigma, \delta)$ where Σ is an alphabet, Q is a finite set of states, and $\delta : Q \times \Sigma \to Q$ is a transition function. For $w \in \Sigma^*$ and $q \in Q$, we define $\delta(q, w)$ inductively: $\delta(q, \epsilon) = q$ and $\delta(q, aw) = \delta(\delta(q, a), w)$ for $a \in \Sigma$.

A word $w \in \Sigma^*$ is called *synchronizing* (or *reset*) for $\mathcal{A} = (Q, \Sigma, \delta)$ if there exists $\overline{q} \in Q$ such that for every $q \in Q$, $\delta(q, w) = \overline{q}$. A DFA is called *synchronizing* (or *reset*) if it admits any synchronizing word. Note that not every automaton is synchronizing. An example of a non-synchronizing DFA is depicted in Fig. 3.

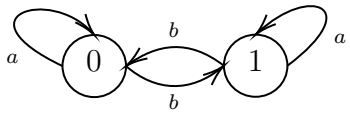


Figure 3: A non-synchronizing DFA

We also define $\mathrm{rt}(\mathcal{A})$ as the length of the shortest word that synchronizes $\mathcal{A}$, if such word exists, and refer to that value as the *reset threshold* of $\mathcal{A}$. The reset threshold

for the automaton $\mathcal{C}_4$ from Fig. 1 is 9 since, it is easy to check that *baaabaaab* is the shortest synchronizing word for that automaton.

Consider a DFA $\mathcal{A} = (Q, \Sigma, \delta)$ and $S \subseteq Q$. We say that $\mathcal{A}$ is *synchronizable* to S (or S is *reachable* in $\mathcal{A}$) if there exists a word $w \in \Sigma^*$ such that $\delta(Q, w) = S$ ($\delta(Q, w)$ stands here as the image of the set of states by the action of δ under the word w). We can also say that *w synchronizes automaton $\mathcal{A}$ to a subset S*. The automaton $\mathcal{C}_4$ from Fig. 1 is synchronizable to the set $\{0, 2, 3\}$ by the word ba, which can be easily verified using Fig. 2.

A *Partial finite automaton* (PFA) is an ordered tuple $\mathcal{A} = (\Sigma, Q, \delta)$ where Σ is a finite set of letters, Q - a finite set of states, and $\delta : Q \times \Sigma \to Q$ is a partial transition function (i.e., it may not be defined everywhere). For $w \in \Sigma^*$ and $q \in Q$, we define $\delta(q, w)$ inductively: $\delta(q, \epsilon) = q$ and $\delta(q, aw) = \delta(\delta(q, a), w)$ for $a \in \Sigma$, where ϵ is the empty word, and $\delta(q, a)$ is defined. A word $w \in \Sigma^*$ is called *carefully synchronizing* if there exists $\overline{q} \in Q$ such that for every $q \in Q$ holds $\delta(q, w) = \overline{q}$ and all transitions are defined. A PFA is called *carefully synchronizing* if it admits any carefully synchronizing word. Note that PFAs are very practical model since in real world scenarios it is rarely true that all possible interactions are defined for all states of the system. An example of a carefully synchronizing automaton, $\mathcal{A}_{car}$, is depicted in Fig. 4. Its shortest carefully synchronizing word, w_{car}, is $abc(ab)^2c^2a$, which can be easily checked via the power automaton construction.

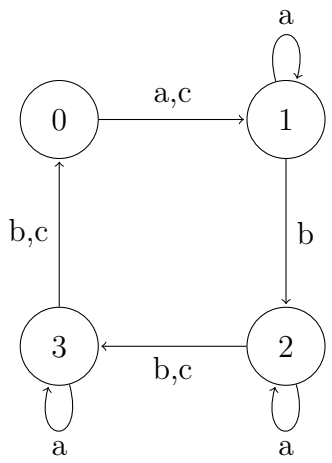


Figure 4: A carefully synchronizing $\mathcal{A}_{car}$

For a given DFA (or PFA) $\mathcal{A} = (Q, \Sigma, \delta)$ we define the *power automaton* $\mathcal{P}(\mathcal{A}) = (2^Q, \Sigma, \tau)$, where 2^Q stands for the set of all subsets of Q, and the alphabet Σ is defined as in $\mathcal{A}$. The transition function $\tau : 2^Q \times \Sigma \to 2^Q$ is defined as follows: let $Q' \subseteq Q$; for every $a \in \Sigma$ we define $\tau(Q', a) = \bigcup_{q \in Q'} \delta(q, a)$ if $\delta(q, a)$ is defined for all states $q \in Q$, otherwise $\tau(Q', a)$ is not defined.

The power automaton of the automaton $\mathcal{C}_4$ (Fig. 1) is depicted in Fig. 5. Bold arrows correspond to the shortest path from the state $Q = \{0, 1, 2, 3\}$ to a singleton and determine the shortest synchronizing word for $\mathcal{C}_4$, which is *baaabaaab*.

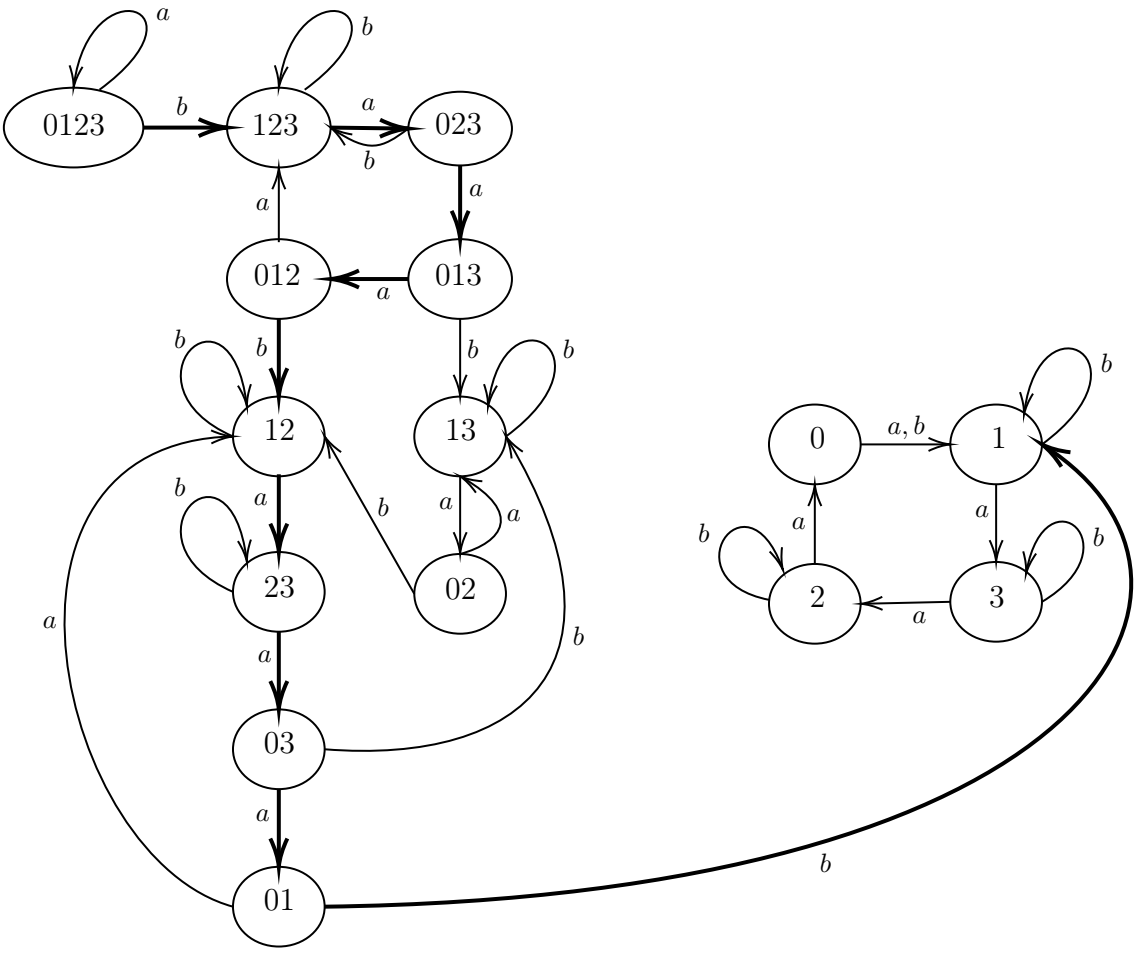


Figure 5: The power automaton $\mathcal{P}(\mathcal{C}_4)$

For the sake of simplicity, we make use of the notation $q.w$ instead of $\delta(q, w)$ and $S.w$ instead of $\delta(S, w)$ for $S \subset Q$ of a DFA (or PFA) with the set of states Q wherever it does not cause ambiguity. We also define $q.w^{-1} = \{p \in Q : p.w = q\}$ and $S.w^{-1} = \{q.w^{-1} : q \in S\}$. In other words, $S.w^{-1}$ is the *preimage* of the subset of states S under the action of the word w. We can now state an obvious fact, useful for deciding whether a given PFA is carefully synchronizing, whether a given DFA is synchronizing, or whether a given DFA is synchronizable to a given subset.

Fact 1. *Let $\mathcal{A}$ be a PFA and let $\mathcal{P}(\mathcal{A})$ be its power automaton. Then $\mathcal{A}$ is synchro-*

nizing (resp. synchronizable to $S \subseteq Q$) if and only if for some state $q \in Q$ (resp. for S) there exists a path in $\mathcal{P}(\mathcal{A})$ from Q to q (resp. to S). The shortest synchronizing word (resp. word synchronizing Q to S) for $\mathcal{A}$ corresponds to the shortest labelled path in $\mathcal{P}(\mathcal{A})$ as above.

We state the characterization of synchronizing DFAs obtained by Černý in [56] (Theorem 1).

Theorem 1. *A DFA $\mathcal{A}$ is synchronizing if and only if for any $p, q \in Q$, there exists a word $w \in \Sigma^*$, such that $p.w = q.w$.*

Theorem 1 can be used to design a simple, polynomial time algorithm that decides if a given DFA is synchronizing and returns a synchronizing word if so. For a given $\mathcal{A} = (Q, \Sigma, \delta)$ define a *pair automaton* $\mathcal{A}^2 = (Q', \Sigma, \delta')$ as follows:

- $Q' = \{\{p, q\} : p, q \in Q\} \cup Q$
- for $S \in Q'$ and $a \in \Sigma$, $\delta'(S, a) = \delta(S, a)$

The pair automaton $\mathcal{C}_4^2$ is shown in Fig. 6.

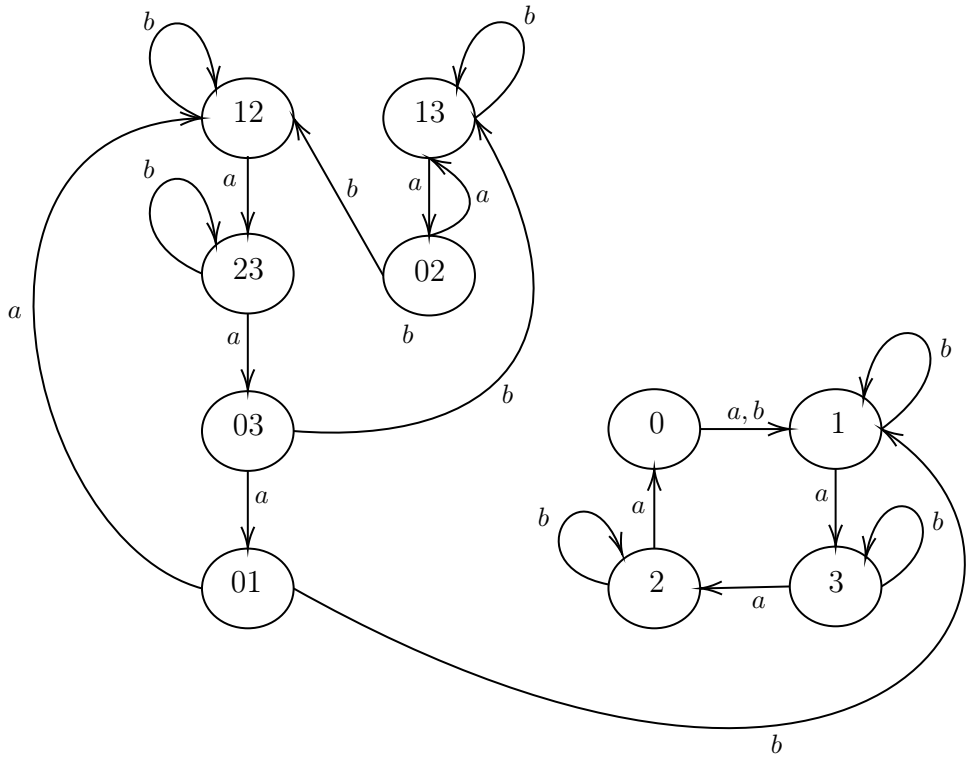


Figure 6: Pair automaton $\mathcal{C}_4^2$

The algorithm, based on Theorem 1, for deciding whether a given DFA is synchronizing is shown as Algorithm 1. Its correctness is straightforwardly implied by Theorem 1. Note that computing the pair automaton $\mathcal{A}^2$ is polynomial in the size of $\mathcal{A}$. Hence, the entire algorithm operates in polynomial time with respect to the size of $\mathcal{A}$. It is worth to mention that there is no similar characterization (as in Theorem 1) for the notion of careful synchronization or synchronizability to a subset.

Algorithm 1 An algorithm for deciding synchronizability of a DFA

Require: $\mathcal{A} = (Q, \Sigma, \delta)$

Ensure: $w \in \Sigma$ synchronizing word for $\mathcal{A}$, or ϵ if such w does not exist

1: $w \leftarrow \epsilon$
2: $P \leftarrow Q$
3: **while** $|P| > 1$ **do**
4: choose $p, q \in P$
5: using $\mathcal{A}^2$ find $v \in \Sigma$ such that $p.v = q.v$
6: **if** v does not exist **then**
7: **return** ϵ
8: **end if**
9: $w \leftarrow wv$
10: $P \leftarrow P.v$
11: **end while**
12: **return** w

The word v from Line 5 of the algorithm can be found using, for example, the DFS algorithm on the automaton $\mathcal{A}^2$.

Let $\mathcal{L}_n = \{\mathcal{A} = (\Sigma, Q, \delta) : \mathcal{A}$ is carefully synchronizing and $|Q| = n\}$. Notice that $\mathcal{L}_n$ does not depend on the alphabet size. We define $d(\mathcal{A}) = \min\{|w| : w$ is a carefully synchronizing word for $\mathcal{A}\}$ and $d(n) = \max\{d(\mathcal{A}) : \mathcal{A} \in \mathcal{L}_n\}$.

Before moving further, we provide a couple of graph theoretic definitions used mainly in Sections 4 and 6. Let V be a finite set of elements (vertices), E be a subset of the set of all pairs of the vertices from V (set of edges). Then we call $G = (V, E)$ an *undirected graph* (or simply a graph). If $E \subseteq V \times V$ then we call $G = (V, E)$ a *directed graph*, or simply a digraph. A *directed path* in a digraph G is a sequence of distinct vertices $(v_1, \ldots, v_k)$ such that $(v_i, v_{i+1}) \in E$ for each $i \in \{1, \ldots k-1\}$ and, for every $i \neq j$, holds $v_i \neq v_j$. For the sake of simplicity, we consider an isolated vertex as a directed path. A *cycle* in a digraph G is a sequence $(v_1, \ldots, v_k, v_1)$ such

that $(v_1, \ldots, v_k)$ is a directed path and $(v_k, v_1) \in E$. We can define a path and a cycle for an undirected graph in a similar way. We say that a digraph is *strongly connected* if for any $(v_1, v_2) \in V \times V$ there exist a path from v_1 to v_2. In case of an undirected graph we say about *connectivity* rather than strong connectivity. We say that a digraph is *weakly connected* if its undirected version (we change each edge to an unordered pair) is connected.

Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a DFA (or PFA), $a \in \Sigma$, $S \subset Q$, and $G = (V, E)$ be a digraph. We say that S *induces* G *on the letter* a *in* $\mathcal{A}$ if a digraph with vertex set S and edge set created from corresponding edges labelled with a is isomorphic to G. For example, the set $\{0, 3, 2\}$ induces a directed path $2 \to 3 \to 0$ in the automaton $\mathcal{A}_{car}$.

Let $\Pi \subset \Sigma$. We define an automaton $\mathcal{A}$ *restricted to* Π as a DFA (or PFA) $\mathcal{B} = (Q, \Pi, \gamma)$ where γ is defined as δ but only on letters from Π. For a letter $a \in \Sigma$ defined for all states, let $\mathcal{G}_a = (Q, E)$ be a directed graph such that $(p, q) \in E$ if and only if $p.a = q$. Put differently, $\mathcal{G}_a$ is a digraph made of only the edges labelled by a in $\mathcal{A}$ or an automaton $\mathcal{A}$ restricted to a.

The graph $\mathcal{G}_a$ is a disjoint union of weakly connected components called *a-clusters*. Since each state has only one outgoing edge in $\mathcal{G}_a$, each a-cluster contains a unique cycle, called *a-cycle*, with some trees attached to a cycle at their roots. For each $p \in Q$, we define a *level* of p as the length of the shortest path between p and any of the vertices on the cycle of the a-cycle. The level of a vertex on the cycle equals 0. The *level* of $\mathcal{G}_a$ is the maximal level of its vertices. An example of the a-cluster is shown in Fig. 7. The level of the vertex v_1 is 1, the level of the vertex v_2 is 2, the level of the vertex v_3 is 3 and the level of the vertex v_4 is 0. The level of the cluster is 3.

One can easily deduce from Fact 1, that at least one letter $a \in \Sigma$ must be defined for all the states of the PFA for it to be carefully synchronizing. We define a *one-cluster* PFA with respect to the letter a as a PFA that has only one a-cluster. We

also refer to a *one-cluster* PFA as an automaton that is one-cluster with respect to any of its letters. Whenever not stated differently in the thesis, when we refer to a one-cluster automaton, we assume that it is one-cluster with respect to the letter a.

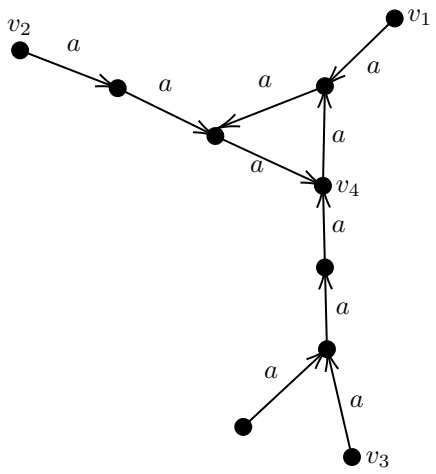


Figure 7: An example of the a-cluster

We say that two DFAs (or PFAs) $\mathcal{A} = (Q, \Sigma, \delta)$ and $\mathcal{A}' = (Q, \Sigma, \delta)$ are *isomorphic* if there exists a bijection $f : Q \to Q'$ such that, for all $p, q \in Q, a \in \Sigma$ we have $\delta(p, a) = q$ if, and only if $\delta'(f(p), a) = f(q)$.

Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a DFA or a PFA and let $S \subseteq Q$. We define an automaton *induced by* S as $\mathcal{A}_S = (S, \Sigma, \delta_S)$, where $\delta_S(p, a) = \delta(p, a)$ if $\delta(p, a) \in S$, otherwise $\delta_S(p, a)$ is undefined.

In the latter sections, when we refer to an automaton or automata, we mean either a PFA or a DFA, wherever not stated differently. We can define strong connectivity for finite automata (either DFAs of PFAs) in an analogous way as for directed graphs. Namely a DFA (or PFA) is *strongly connected* if for any $p, q \in Q$ there exist a word $w \in \Sigma^*$ such that $p.w = q$. We say that an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ is *weakly connected* if its underlying (undirected) graph is connected. We say that $S \subseteq Q$ *induces a strongly connected component* on $\mathcal{A}$ if for all $p, q \in S$ there exists a word $w \in \Sigma^*$ such that $\delta(p, w) = q$.

The following fact gives us a simple structural necessary condition for an automaton to be carefully synchronizing:

Fact 2. *If an automaton $\mathcal{A}$ is carefully synchronizing, then there exists a letter $a \in \Sigma$, such that $\delta(q, a)$ is defined for all $q \in Q$. Additionally, the letter $a \in \Sigma$ partitions the set of states into a-clusters.*

2.2 Classic synchronization

We begin this section with the formulation of the Černý Conjecture (sometimes referred to as the Černý-Starke Conjecture or the Černý-Liu Conjecture):

Conjecture 1 ([56])**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a synchronizing DFA with $|Q| = n$. There exists a word $w \in \Sigma^*$ that synchronizes $\mathcal{A}$ and $|w| \leq (n-1)^2$.*

Researchers have established a significant number of results concerning the synchronization of DFAs over the years, including those related to the length of the shortest synchronizing words (and proofs of the Černý Conjecture for various classes of automata), complexity issues, and language-theoretical properties of synchronization. This section will briefly summarize the current state of knowledge in this area. For an in-depth survey, the reader can refer to Volkov's recent paper [57].

As discussed previously, the Černý Conjecture has remained open for over 60 years. This fact has motivated, and continues to motivate, many scientists to study upper and lower bounds of synchronizing words for DFAs. Before proceeding with the upper bounds, we (following Černý in [56]) construct an infinite family of automata that realize the lower bound from the Černý Conjecture for the length of the shortest synchronizing words (i.e., have the shortest synchronizing words of length $(n-1)^2$), thus proving the following:

Theorem 2. *For every $n \in \mathbb{N}$, there exists an automaton $\mathcal{C}_n$ that has the shortest synchronizing word of length $(n-1)^2$.*

Let $n \in \mathbb{N}$, and define $\mathcal{C}_n = (\{0, \ldots, n-1\}, \{a, b\}, \delta_n)$ where δ_n acts as follows:

$$\delta_n(q, x) = \begin{cases} q + 1 \bmod n & \text{if } x = a, \\ q & \text{if } x = b \wedge q \neq 0, \\ 1 & \text{if } x = b \wedge q = 0. \end{cases}$$

It is proven in [56] that the shortest synchronizing word for the automaton $\mathcal{C}_n$ is $(ba^{n-1})^{n-2}b$. The automaton $\mathcal{C}_4$ can be found in Fig. 1.

To this day $\{\mathcal{C}_n : n \in \mathbb{N}\}$ is the only known infinite family of automata that realizes the bound of $(n-1)^2$ for the length of reset words. Apart from this family, only isolated examples of such *extremal automata* (i.e., realizing Černý's bound) are known; some can be found in [26, 39, 54]. Additionally, there are a few examples of families of automata achieving quadratic lower bounds for the shortest synchronizing word for DFAs, including those with a sink state (see for example [44], Theorem 6.1), automata with co-prime cycle lengths ([21]), and automata with full transition monoid ([19]). The most recent studies focus on the state complexity issues of synchronization, for example [23, 22]

The best known upper bound for the length of the shortest synchronizing word is $\alpha \cdot n^3 + O(n^2)$ (with $\alpha \leqslant 0.1654...$) in the general case, due to Y. Shitov, who improved M. Szykuła's upper bound by a constant factor (established in [51] by utilizing so-called avoiding words and linear algebra arguments) in [48], using a slightly different counting argument than in [51]. Furthermore, there are proofs of the Černý Conjecture for quite a few classes of automata, including for example, monotonic automata [14], aperiodic automata [55], automata on Eulerian digraphs [27], and circular automata [12].

There are also numerous theorems concerning algorithmic and complexity issues related to synchronizing DFAs. The first natural problem to consider is whether a given DFA is synchronizing. Stated formally:

DFA-SYNC
Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$
Question: Is $\mathcal{A}$ synchronizing?

One can easily develop an algorithm that answers this question in polynomial time, utilizing Theorem 1. It is also known that DFA-SYNC is NL-complete. According to [57], the theorem was proven by the Indian computer scientist Sreejith but has not been published by him. The proof can be found in [57], Proposition 2.2.

While determining if a given DFA is synchronizing is easy, checking if it has a short synchronizing word is believed to be a hard problem. Let us state it formally:

SHORT-DFA-SYNC
Input: A synchronizing DFA $\mathcal{A} = (Q, \Sigma, \delta)$ and $k \in \mathbb{N}$
Question: Does $\mathcal{A}$ admit a synchronizing word of length $\leq k$?

The NP-completeness of SHORT-DFA-SYNC was first proven by Rystsov in [43] and then rediscovered several times (for example, in [14] or [46]).

Since obtaining the exact lengths of the shortest synchronizing words is computationally challenging, it is natural to inquire about the existence of polynomial-time algorithms that approximate it. Let P be any algorithm that approximates, for a given DFA $\mathcal{A}$, the value $\mathrm{rt}(\mathcal{A})$ and operates in a polynomial number of steps. We say that P *approximates the reset threshold to within an accuracy* of $f(n)$ if the following inequality holds for any n and any n-state synchronizing $\mathcal{A}$:

$$\frac{P(\mathcal{A})}{\mathrm{rt}(\mathcal{A})} \leq f(n).$$

In [16], Gawrychowski and Straszak have proven the following theorem.

Theorem 3. *Let $\epsilon > 0$. If $P \neq NP$, then no polynomial-time algorithm approximates the reset threshold to within an accuracy of $n^{1-\epsilon}$.*

This theorem implies that (assuming $P \neq NP$), there exist no algorithm that approximates the length of the shortest synchronizing words within a linear factor in the general case.

In the past 20 years, the development of powerful computational devices has led to a number of experimental results concerning the Černý Conjecture. Results from [2, 11, 8, 29, 30, 54] have confirmed the conjecture for any DFA with at most 7 states and an input alphabet of any size, for any 8-state DFA with three input letters, and for any DFA with two input letters and at most 12 states. However, attempts to disprove the conjecture by randomly searching for counterexamples might be futile (at least for small alphabet sizes), as shown in [7], where the authors have demonstrated that the shortest synchronizing word for a randomly generated DFA is $O(\sqrt{n} \cdot \log n)$ with high probability.

2.3 Generalizations

Many practical problems demand the notion of synchronization for models other than DFAs, or require a slightly different approach to the synchronization itself. This has led to a generalization of the definition to accommodate more universal frameworks. Most of those generalizations consider nondeterminism. Two such concepts, namely careful synchronization and synchronizability to a subset, were introduced in Section 2. We will delve into further discussions regarding these concepts, along with presenting related results, in this section.

Define a *nondeterministic finite automaton* (NFA) $\mathcal{A} = (Q, \Sigma, \delta)$, where Σ is a non-empty finite set of letters, Q is a non-empty finite set of states, and $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation. We can extend the relation δ for Σ^* in a similar way as in the definition of DFA and PFA. We say (following [24]) that $\mathcal{A}$ is *i-directable* (for $i \in 1, 2, 3$) if there exists a word $u \in \Sigma^*$ such that:

- $(i = 1)$ $\forall_{p,q \in Q}$ $p.u = q.u$ and $p.u$ is a singleton set.

- $(i = 2)\ \forall_{p,q \in Q}\ p.u = q.u.$
- $(i = 3)\ \bigcap_{q \in Q} q.u \neq \varnothing.$

It has been shown in [25] (Lemma 3) that while dealing with 3-directability we can assume that δ is a partial function instead of a transition relation in the definition of an NFA, which gives us equivalence with the notion of careful synchronizability. Upper and lower bounds for the lengths of the shortest i-directing words are presented in Table 1. All of those bounds were obtained without the restriction on the alphabet size.

Table 1: The directing words bounds

	$i = 1$	$i = 2$	$i = 3$
Upper bound	$O(2^n)$ [17]	$O(2^n)$ [17]	$O(n^2 \cdot \sqrt[3]{4^n})$ [17]
Lower bound	$\Omega(2^n)$ [25]	$\Omega(2^n)$ [24]	$\Omega(\sqrt[3]{3^n})$ [32]

The problem of determining whether a given NFA is 1, 2, or 3-directable seems to be harder than deciding whether a given DFA is synchronizing. This belief stems from the fact that the shortest 1, 2, or 3-directable words can be of exponential length. Indeed, there is a proof in [34] that checking if a given NFA is $1, 2, 3$-directable (as well as checking if the given PFA is carefully synchronizing) is PSPACE-complete, even for a binary alphabet. It can be easily verified that the Černý Conjecture is not true for careful synchronization of PFAs, since $|w_{car}| = 10 > (4-1)^2 = 9$ for the 4-state automaton in Fig. 4.

In [5], the authors provide a definition of synchronizability for a strongly connected PFA different than careful synchronization. The difference lies in how undefined transitions are treated. While in careful synchronization, for any prefix w' of w, all transitions $Q.w'$ must be defined, in [5], the authors accept the possibility of traversing undefined transitions. These considerations led to establishing a corre-

spondence between that definition and the classical definition of synchronization for DFAs.

Another notion we want to describe here is the *subset synchronization* of a DFA. Even if a DFA $\mathcal{A} = (Q, \Sigma, \delta)$ is not synchronizing, there could be various subsets $S \subset Q$ such that $|S.w| = 1$ for $w \in \Sigma^*$. We then say that a subset S *is synchronizable* in $\mathcal{A}$. It has been shown in [47] that the problem of deciding if a given subset is synchronizable in $\mathcal{A}$ is PSPACE-complete.

2.4 Applications

Longstanding open problems such as the Černý Conjecture always draw the attention of mathematicians and computer scientists all over the world. Besides that fact, synchronization of finite automata also has numerous practical aspects, which should not be surprising, since finite automata often model real-world systems. The notion of synchronization is a quite natural concept itself. The following section condenses some of the applications of that theory.

Coding theory makes use of synchronization in a rather elegant way. Decoding prefix codes can be easily implemented using finite automata. Assuming transmission failures between the sender and receiver of the message, a resetting sequence can ensure that after such failure the next bits of the message can be decoded correctly. This branch of coding theory is still under active investigation (see for example [6, 45]).

Synchronization problems naturally emerge in **industrial mechanics** during the development of handling devices for various component manipulations, such as feeding, sorting, and packaging. Within this domain, the concept of a synchronizing automaton was independently rediscovered by Natarajan [36] in the mid-1980s. Natarajan demonstrated how these automata can be utilized in designing orienters for planar rigid polygonal objects. We evoke here the example of Volkov and Ananichev from [1].

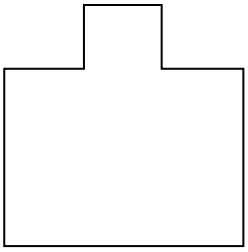


Figure 8: A part

Suppose that a part of a certain device has the polygonal shape depicted in Fig. 8. Such parts arrive to a manufacture in four possible orientations shown in Fig. 9 and need to be sorted and oriented before assembly. Suppose also that before the assembly the part should be oriented to the left (fourth orientation in Fig. 9). We should design an orienter that puts part in the prescribed position independently of the initial orientation.

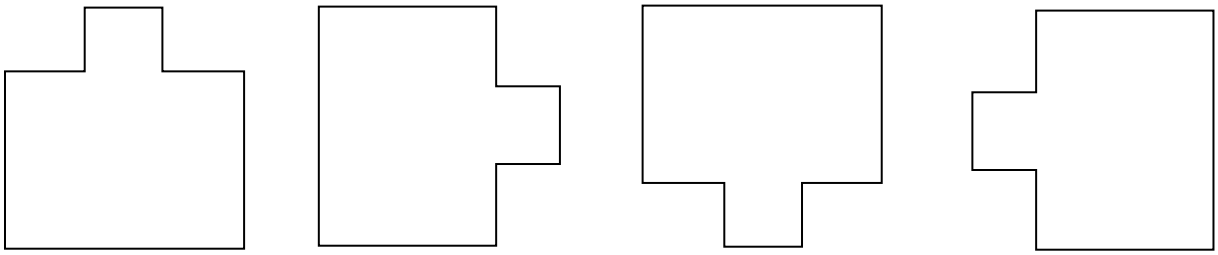


Figure 9: Possible orientations

To proceed, we utilize a conveyor belt system that transports parts to the assembly point. As the parts travel along the belt, they encounter a series of passive obstacles strategically positioned along its path. These obstacles come in two types: high and low. A high obstacle is positioned to intercept any part at its rightmost low angle, ensuring that the part is compelled to rotate clockwise as it continues along the belt (assuming the belt moves from left to right). Conversely, a low obstacle only affects parts in the "bump-down" orientation (as depicted in Fig. 9), causing them to rotate, while parts in other orientations pass by unaffected.

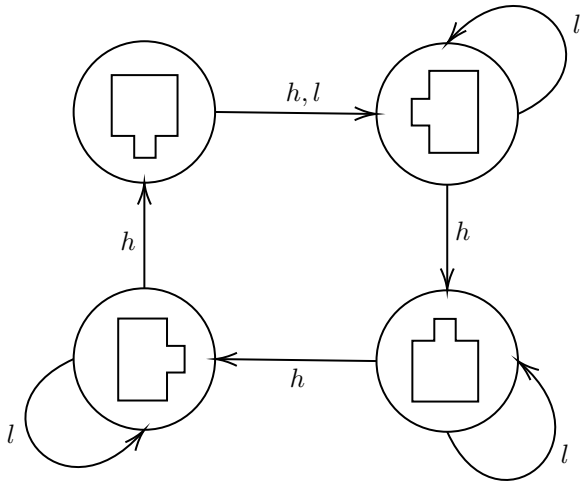


Figure 10: The actions of the obstacles

The diagram in Fig. 10 illustrates the impact of the aforementioned obstacles on the orientation of the part. By recalling the synchronizing automaton depicted in Fig. 1, with its shortest reset word being ba^3ba^3b, we deduce that arranging the obstacles in the sequence $lhhhlhhhl$ achieves the intended outcome of a sensor-free orienter.

Model-based testing is another crucial field that leverages synchronizing words. Consider a system that can be represented by a finite automaton, where the current state of the system is unknown. Synchronizing words facilitate rapidly transitioning the system to the desired state. For further insights into state identification problems, Sandberg's survey provides extensive information [47].

Reset words serve a similar function in motion planning problems in **robotics**, specifically in guiding a robot that has deviated from its intended path back to a predetermined location (see for example [50]).

3 Careful synchronization and subset synchronization

3.1 Reducing the number of letters

Allowing no outgoing transitions from some states for certain letters helps us to model a system for which certain actions cannot be accomplished while being in a specified state. This leads to the problem of finding a synchronizing word for a finite automaton, where transition function is not defined for all states. Notice that this is the most frequent case, if we use automata to model real-world systems. In practice, it rarely happens that a real system can be modeled with a DFA where a transition function is total. The transition function is usually a partial one. This fact motivated many researchers to investigate the properties of partial finite automata relevant to practical problems of synchronization.

In this section we provide a construction of series of automata with reduced number of letters, comparing to Martyugin construction (in [33]), but with the same asymptotic length of the shortest carefully synchronizing words. In particular we construct an infinite family of automata with number of states equal to n, and number of letters equal to $\lfloor \frac{2}{9}n \rfloor + 2$ with shortest carefully synchronizing words of length $\Theta(3^{\frac{n}{3}})$.

This is an improvement in a number of letters, since the best known construction (Martyugin) uses $\frac{1}{3}n + 1$ letters to achieve shortest words of the same length. It is worth mentioning that family of automata in [33] has shortest carefully synchronizing words of length $3^{\frac{n}{3}}$, where n is the number of states, which is nearly exactly the same result as presented in this paper. In [9] the authors obtained an infinite family of PFAs over a binary alphabet having the shortest carefully synchronizing words of length $\Omega(\frac{2^{n/3}}{n^{3/2}})$ and over a ternary alphabet of length $\Omega(\frac{2^{2n/5}}{n})$, which is asymptotically smaller than our result.

We start this section with reminding the Martyugin's construction from [33]. Let

$m = 3n, n \in \mathbb{N}$, $P = \bigcup_{i=0}^{m-1} P_i$ where $P_i = \{p_0^i, p_1^i, p_2^i\}$ and $\mathrm{T} = \{x_0, \ldots, x_{m-1}, y\}$. Define the PFA $\mathcal{B}_m = (P, \mathrm{T}, \gamma)$, where $\gamma : P \times \mathrm{T} \to P$, as follows:

- $\gamma(p_j^i, x_m) = p_j^i$ if $m < i$
- $\gamma(p_j^i, x_m) = p_{j-1}^i$ if $m = i$ and $j \neq 0$
- $\gamma(p_0^i, x_m) = p_0^i$ if $m = i$
- $\gamma(p_0^i, x_m) = p_2^i$ if $m > i$
- $\gamma(p_0^i, y) = p_0^1$

All other transitions are not defined. The automaton $\mathcal{B}_3$ is presented in Fig. 11. Its shortest carefully synchronizing word is:

$$v = (x_0)^2 x_1 (x_0)^2 x_1 (x_0)^2 x_2 (x_0)^2 x_1 (x_0)^2 x_1 (x_0)^2 x_2 (x_0)^2 x_1 (x_0)^2 x_1 (x_0)^2 y$$

and $|v| = 27$.

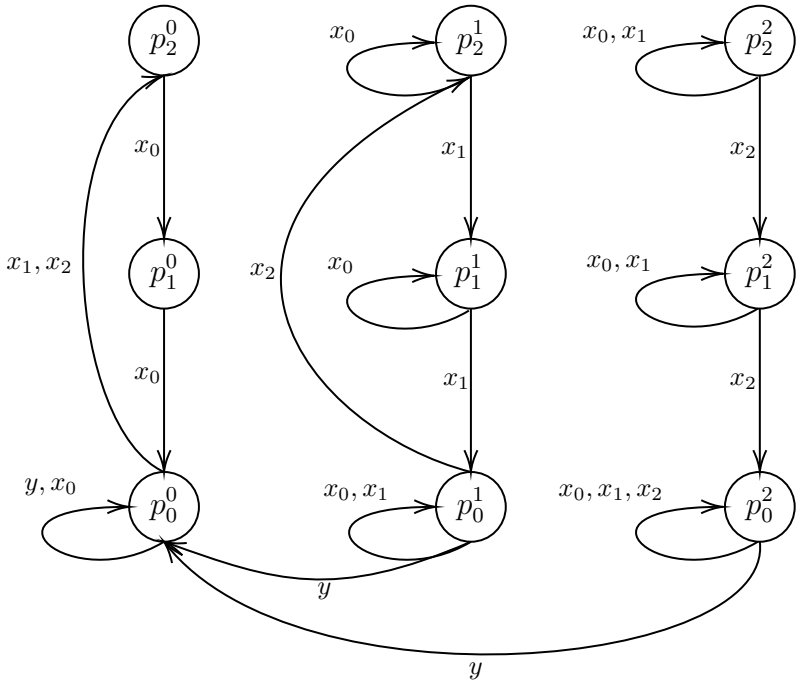


Figure 11: The automaton $\mathcal{B}_3$

The idea of our construction is similar to [33] (implementing the "counter" on a power automaton), but by choosing a different method of "dividing" the set of states we obtain significantly (linear factor) lower alphabet size. In order to make proofs easier for the reader we assume that $n = 9k$ for $k \in \mathbb{N}$, but it is easy to modify the construction to work in the general case. Consider the automaton $\mathcal{A}_3$ depicted in Fig. 12.

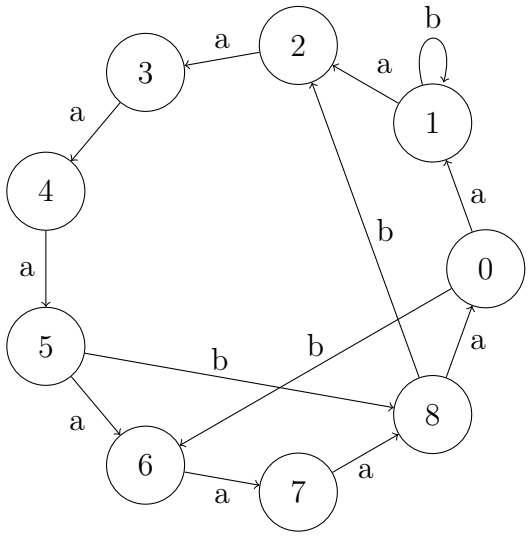


Figure 12: The automaton $\mathcal{A}_3$

Let $\mathcal{A}_3^n = (Q, \Sigma, \delta)$ such that $Q = Q_1 \cup ... \cup Q_k$ where $Q_i = \{q_0^i, .., q_8^i\}$ and define $\Sigma = \{c, a_1, b_1, a_2, b_2, .., a_k, b_k, d\}$. The transition function is defined as follows:

1. $\delta(q_{3m+i}^j, c) = q_i^j$ for $m \in \{0, 1, 2\}$ and $i \in \{0, 1, 2\}$
2. δ on $\{a_i, b_i\}$ imitates $\mathcal{A}_3$ on the subset of states Q_i
3. $\delta(q_0^i, a_j) = q_0^i$, $\delta(q_5^i, a_j) = q_1^i$, $\delta(q_7^i, a_j) = q_2^i$ for $i < j$
4. $\delta(q_0^i, b_j) = q_0^i$, $\delta(q_5^i, b_j) = q_1^i$, $\delta(q_7^i, b_j) = q_2^i$ for $i < j$
5. $\delta(q_l^i, a_j) = \delta(q_l^i, b_j) = q_l^i$, for $i > j$ and for all $l \in \{0, .., 8\}$
6. $\delta(q_0^i, d) = \delta(q_5^i, d) = \delta(q_7^i, d) = q_0^1$

Denote any $\{q_f^i, q_g^i, q_h^i\} = Q_s^i$, where $\{f, g, h\}$ is a state in $\mathcal{P}(\mathcal{A}_3)$, and s is the length of a path from $\{0, 1, 2\}$ to $\{f, g, h\}$. For example $\{q_3^i, q_4^i, q_5^i\} = Q_3^i$, as can be verified with Fig. 13. Also denote $S_b = \bigcup_{i=1}^{k} Q_0^i$ and $S_f = \bigcup_{i=1}^{k} Q_{26}^i$. The intuition behind the construction is that after applying the letter c to the state Q in the power automaton of $\mathcal{A}_3^n$, we obtain the state S_b. Then, we treat $Q_i' \subset Q_i$, such that $|Q_i'| = 3$ and $Q_i' \subset S_b$, as the i-th position of 0 in a k digit number of base 27. Any consecutive letter of the shortest carefully synchronizing word of that automaton acts like incrementing the former number by one. Namely, we implement the base 27 counter with the power automaton of $\mathcal{A}_3^n$.

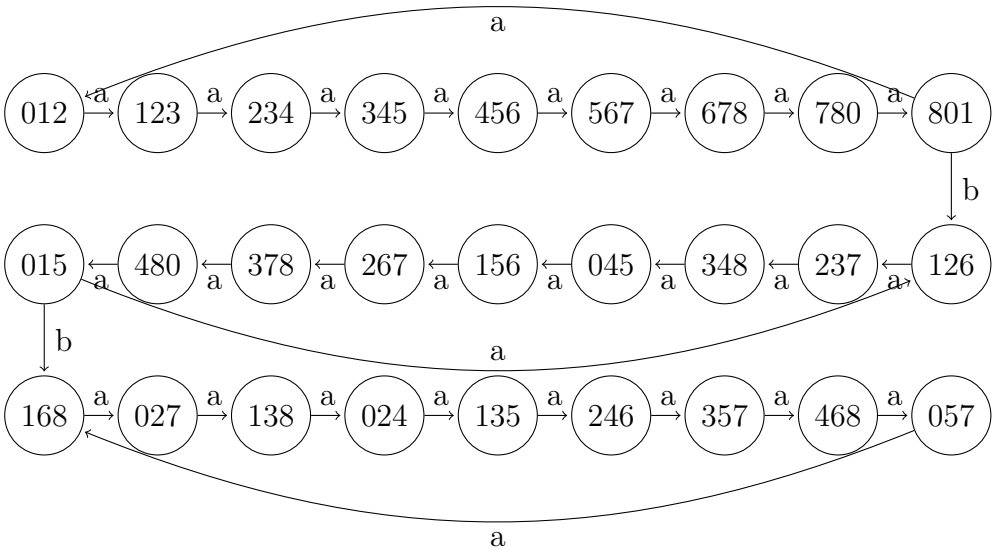


Figure 13: The path from 012 to 057 in $\mathcal{P}(\mathcal{A}_3)$

Lemma 1. *The shortest word $w \in \Sigma^*$ such that $\tau(S_b, w) = S_f$ is of length $(3^3)^k - 1$.*

Proof. The result follows by induction on k.

Let $k = 1$. Then the result is easily verified with Fig. 13.

Now, assume that the statement holds for $k - 1$. It means that there exists w' such that $\tau(\bigcup_{i=1}^{k-1} Q_0^i, w') = \bigcup_{i=1}^{k-1} Q_{26}^i$. We can easily verify from the definition of δ (point 5), that also $\tau((\bigcup_{i=1}^{k-1} Q_0^i) \cup Q_i^k, w') = (\bigcup_{i=1}^{k-1} Q_{26}^i) \cup Q_i^k$ for $0 \leq i < 27$. From

the definition of δ (point 3) we can deduce that $\tau(S_b, w'a_k) = (\bigcup_{i=1}^{k-1} Q_0^i) \cup Q_1^k$. We can repeat this reasoning to obtain $\tau(S_b, (w'a_k)^8) = (\bigcup_{i=1}^{k-1} Q_0^i) \cup Q_8^k$. Notice that $\tau(S_b, (w'a_k)^9) = S_b$. However, $\tau(S_b, (w'a_k)^8 w'b_k) = (\bigcup_{i=1}^{k-1} Q_0^i) \cup Q_9^k$. Acting like that, we obtain $\tau(S_b, w) = S_f$, where $w = (w'a_k)^8 w'b_k (w'a_k)^8 w'b_k (w'a_k)^8 w'$. To prove the minimality of w, it suffices to show that, for each prefix v of w and for each $e \in \Sigma$ such that ve is not a prefix of w, either there exists a prefix u of v such that $\tau(S_b, ve) = \tau(S_b, u)$ or $\tau(S_b, ve)$ is not defined. Straight from the definition of δ we obtain that $\tau(S_b, vc) = S_b$ and $\tau(S_b, vd)$ is not defined. Consider the state $\tau(S_b, v) = (\bigcup_{i=1}^{l-1} Q_{p_i}^i) \cup (\bigcup_{j=l}^{k-1} Q_{r_j}^j)$, where $l \in \{1, ..., k\}$, $p_i \in \{8, 17, 26\}$, $r_j \in \{0, ..., 26\}$ and $r_l \notin \{8, 17, 26\}$. Notice that if $m > l$, transitions $\tau((\bigcup_{i=1}^{l-1} Q_{p_i}^i) \cup (\bigcup_{j=l}^{k-1} Q_{r_j}^j), a_m)$ and $\tau((\bigcup_{i=1}^{l-1} Q_{p_i}^i) \cup (\bigcup_{j=l}^{k-1} Q_{r_j}^j), b_m)$ are undefined. Otherwise, if $m < l$ then $\tau((\bigcup_{i=1}^{l-1} Q_{26}^i) \cup (\bigcup_{j=l}^{k-1} Q_{r_j}^j), a_m)$ and $\tau((\bigcup_{i=1}^{l-1} Q_{26}^i) \cup (\bigcup_{j=l}^{k-1} Q_{r_j}^j), b_m)$ results in states visited before, which can be verified from Fig. 13 and the definition of δ. We conclude that the lemma holds. □

Now we can state and prove the following theorem.

Theorem 4. *For each $n > 0$ the automaton $\mathcal{A}_3^n$ has shortest carefully synchronizing word of length $3^{\frac{n}{3}} + 1$ and uses $\lfloor \frac{2}{9} n \rfloor + 2$ letters.*

Proof. It is easy to observe from the definition of δ, that the letter c is the only one defined for all states, so all synchronizing words must start with this letter. Also, we have $\tau(Q_i, c) = Q_0^i$ for all $i \in \{1, .., k\}$, so $\tau(Q, c) = S_b$. From Lemma 1, we deduce that there exists a word w with $|w| = (3^3)^k - 1$, such that $\tau(S_b, w) = S_f$ and w is the shortest word with this property. The letter d is defined for S_f and also $\tau(S_f, d) = q_0^1$. Observe also, that for any proper prefix w' of w we have that $\tau(Q, cw'c) = S_b$ and $\tau(Q, cw'd)$ is not defined. From all that we claim the theorem holds. □

Several families of PFAs with constant alphabet size and exponentially long shortest carefully synchronizing words were presented in [9], but their lengths were asymptotically smaller than in the construction presented here. Observe also that our

construction achieves asymptotically the same lower bound for the shortest carefully synchronizing word as in [32] and the same length of the shortest carefully synchronizing word as in [33] but with reduced alphabet size compared to those results.

3.2 Constant number of letters and subset synchronization

In this section, we provide a construction of automata for a given partition of a number n. Notice that, although this construction does not improve the result established in [9] - where authors obtained the best known lower bounds for $d(n)$ with a restriction of constant alphabet size, it can be used to establish a lower bound for subset synchronization in a DFA.

Let $p = (k_1, .., k_s)$ for $s \geq 1$ be such that $\sum_{i=1}^{s} k_i = n$. Let $n > 1$, $Q^i = \{0_i, 1_i, .., (k_i - 1)_i\}$, $Q = \bigcup_{i=1}^{s} Q^i$ and $\Sigma = \{a, b, c\}$. We define the partial transition function $\delta : Q \times \Sigma \to Q$ for $\mathcal{A}_p = (\Sigma, Q, \delta)$ as follows:

1. $\delta(y_i, a) = 0_i$, $y_i \in Q^i$, $i \in \{1, .., s\}$
2. $\delta(y_i, b) = ((y + 1)\ mod\ k_i)_i$, $y_i \in Q^i$, $i \in \{1, .., s\}$
3. $\delta((k_i - 1)_i, c) = 0_1$, $j \in \{1, .., s\}$

In Fig. 14 we can see $\mathcal{A}_p$ for $p = (2, 2, 3)$ and $\mathcal{P}(\mathcal{A}_p)$ is depicted in Fig. 15.

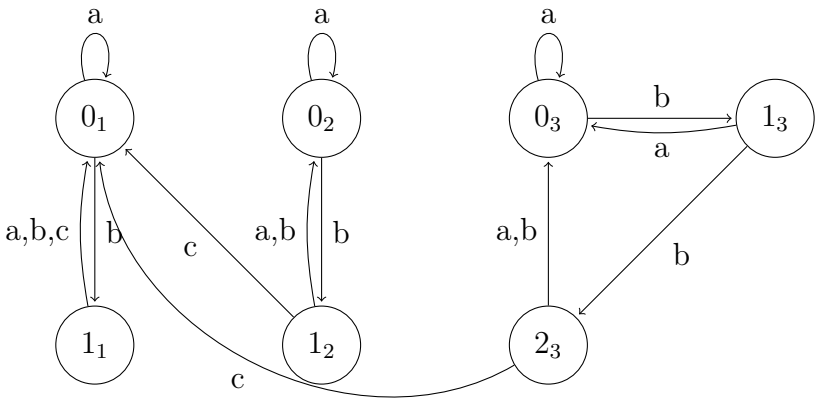


Figure 14: The automaton $\mathcal{A}_p$ for $p = (2, 2, 3)$

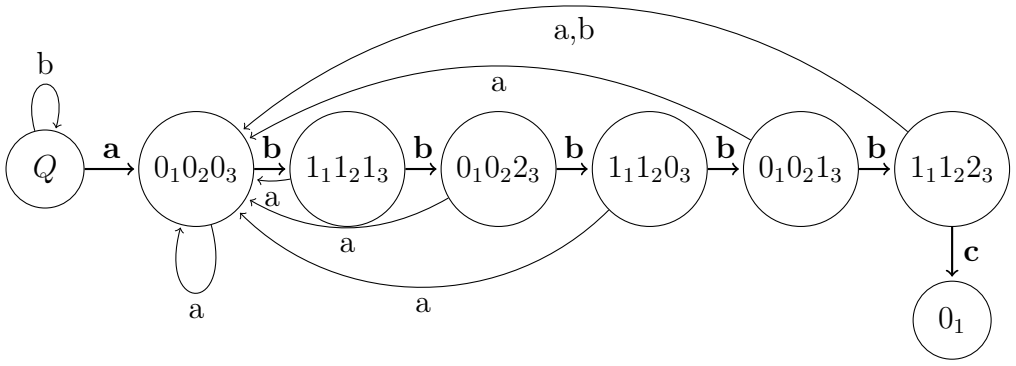


Figure 15: The power automaton $\mathcal{P}(\mathcal{A}_p)$ with the path from Q to 0_1 (bolded arrows)

We state and prove the following theorem.

Theorem 5. *The automaton $\mathcal{A}_p$ is carefully synchronizing and the length of its shortest carefully synchronizing word is $lcm(k_1, .., k_s) + 1$.*

Proof. First, observe that a and b are defined for all states in Q and, since $\delta(\cdot, b)$ is a bijection, we have $\tau(Q, b) = Q$ and $\tau(Q, a) = \{0_1, 0_2, .., 0_s\} = Q_0$. We can also deduce that the permutation type induced by the action of b on the set of states is $[k_1^1 k_2^1 .. k_s^1]$. Denote the order of that permutation by r. From the definition of the order of the permutation we have $\tau(Q_0, b^r) = Q_0$ and, for any $p < q < r$, it holds that $\tau(Q_0, b^p) \neq \tau(Q_0, b^q)$. Denote $\tau(Q, b^p) = Q_p$. From the definition of δ we deduce that, for any p, we have $\tau(Q_p, a) = Q_0$, since for any $i \in \{1, .., s\}$ we have $|Q_p \cap Q_i| = 1$. Letter c is defined only for $Q_p = Q_{r-1} = \{(k_1 - 1)_1, .., (k_s - 1)_s\}$, what can be verified by analysing the definition of δ. On the other hand, r is the order of the permutation induced by b and, by that, we have $r = lcm(k_1, .., k_s)$. So $\tau(Q, ab^{r-1}c) = 0_1$ and the theorem holds. □

Corollary 1. *Let $n \in \mathbb{N}$. There exists $p_n = (k_1, \ldots, k_s)$ such that $k_1 + \ldots + k_s = n$ and $d(\mathcal{A}_{p_n}) \propto e^{(1+o(1))\sqrt{n \ln n}}$.*

Proof. Using Theorem 5, we can construct an automaton for any given cycle decomposition of a permutation. On the other hand, we know that Landau's function [31],

for a given n, is the largest order of an element of S_n. Denote it as $g(n)$. The way to obtain such a family for any n is by taking a permutation p_n with the largest order in S_n and constructing the automaton given in Theorem 5. It is well-known ([31]) that $g(n) \propto e^{(1+o(1))\sqrt{n \ln n}}$ and by that the corollary holds. □

We can also consider the above construction with removed transition labelled with the letter c. Denote such automaton by $\mathcal{B}_p$ for a given partition p. Now we can formulate the following corollary.

Corollary 2. *Let $\mathcal{B}_p$ be defined as above and let $S = \{(k_1-1)_1, .., (k_s-1)_s\}$. Then $\mathcal{B}_p$ is a DFA synchronizable to S, and the shortest word $w \in \Sigma^*$ such that $\delta(Q, w) = S$ is of length $e^{(1+o(1))\sqrt{n \ln n}}$.*

3.3 Summary

We improved the result from [33] and gave a family of automata that has the same asymptotical length of shortest carefully synchronizing word, but with significantly reduced number of letters. We also described how to use our method to construct a family of automata over a three-letter alphabet with the shortest synchronizing word of length $e^{(1+o(1))\sqrt{n \ln n}}$. As a corollary of that construction we gave an infinite series of binary DFAs and the set of states that is reachable with the word of length $e^{(1+o(1))\sqrt{n \ln n}}$) at least.

Notice that the construction from Section 3.2 gives DFAs which are not strongly connected. The case of synchronization to a subset in the class of strongly connected DFAs is worth investigating. In [20] there is a construction of automata having longer words synchronizing to a subset, but the alphabet size is exponential and resulting automata are synchronizing.

4 Automata with coinciding cycles

Real-world processes often exhibit cyclical patterns, which can be effectively modeled using cycles. In numerous applications, segments of one business cycle may also coincide with segments of another business cycle. This scenario can be accurately captured using the following definition.

We say that a DFA is with *coinciding cycles* if there is a set of letters $\Pi \subseteq \Sigma$, $|\Pi| > 1$, such that:

- for every $a \in \Pi$ the function $\delta(\cdot, a)$ is a permutation on Q with exactly one simple directed cycle with more than one vertex, and all other states being self-maps on a (meaning $p.a = p$)
- let $S_a \subseteq Q$ and $S_b \subseteq Q$ be such that S_a induces a directed cycle on the letter $a \in \Pi$ in $\mathcal{A}$, S_b induces a directed cycle on the letter $b \in \Pi$ in $\mathcal{A}$ and $a \neq b$. If $S_a \cap S_b \neq \varnothing$ then $S_a \cap S_b$ induces a directed path on the letter a in $\mathcal{A}$ and on the letter b in $\mathcal{A}$
- $\mathcal{A}$ restricted to Π is strongly connected

An example of an automaton with coinciding cycles is shown in Fig. 16. In this case $\Pi = \{a, b\}$, $S_a = \{q_1, q_2, q_3, q_4, q_5, q_6\}$ and $S_b = \{q_1, q_2, q_3, q_4, q_7, q_8\}$. Observe that $S_a \cap S_b = \{q_1, q_2, q_3, q_4\}$, the letter a induces on that set a directed path $q_4 \to q_3 \to q_2 \to q_1$ and the letter b induces a path $q_2 \to q_1 \to q_4 \to q_3$.

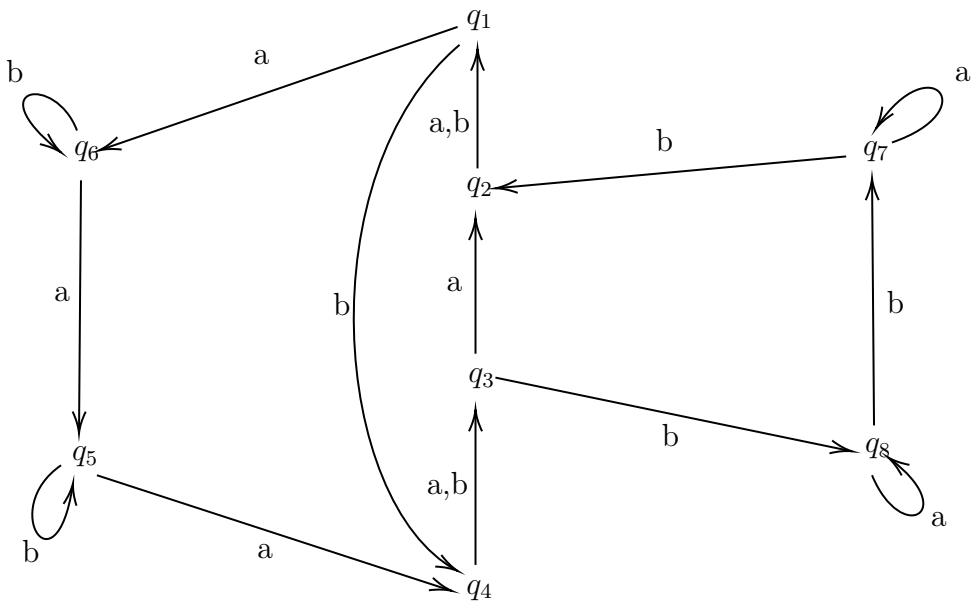


Figure 16: An example of an automaton with coinciding cycles

Let $a, b \in \Pi, p, q \in S_a$. Define $\mathrm{dist}(p, q, a) = \min\{l \in \mathbb{N} : p.a^l = q\}$. Observe that, since S_a induces a cycle, then for any $k \in \mathbb{N}$ holds $\mathrm{dist}(p, q, a) = \mathrm{dist}(p.a^k, q.a^k, a)$. Define also $\mathrm{lastdist}_{a,b}(p)$ as the length of the path from p to the last state in the path induced by a in $S_a \cap S_b$. If $p \notin S_a \cap S_b$ then we set $\mathrm{lastdist}_{a,b}(p) = \infty$. For example, if we take the automaton from Fig. 16, then $\mathrm{lastdist}_{a,b}(q_2) = 1$ (the last element on the path induced by a is q_1) and $\mathrm{lastdist}_{b,a}(q_2) = 3$ (the last element on the path induced by b is q_3). Finally, for $a, b \in \Pi$ we define the functions $\mathrm{mindist}_{\mathrm{a,b}} : Q \times Q \to Q$ and $\mathrm{maxdist}_{\mathrm{a,b}} : Q \times Q \to Q$ as follows:

$$\mathrm{mindist}_{\mathrm{a,b}}(p, q) = \begin{cases} p & \text{if } \mathrm{lastdist}_{a,b}(p) < \mathrm{lastdist}_{a,b}(q), \\ q & \text{otherwise} \end{cases}$$

and

$$\mathrm{maxdist}_{\mathrm{a,b}}(p, q) = \begin{cases} p & \text{if } \mathrm{lastdist}_{a,b}(p) > \mathrm{lastdist}_{a,b}(q), \\ q & \text{otherwise.} \end{cases}$$

First, we make useful observations used in the latter proofs.

Observation 1. *For every $a \in \Pi$, there exists $b \in \Pi$ such that $S_a \cap S_b \neq \varnothing$.*

Proof. It is obvious, since $\mathcal{A}$ restricted to Π is strongly connected and $|\Pi| > 1$. □

Observation 2. *For every $a \in \Pi$, we have $|S_a| < |Q|$.*

Proof. Suppose, for the sake of contradiction, that this is not the case. So there must exist a letter a that is a directed cycle on Q. From Observation 1 we know that there exists $b \in \Pi$ such that $S_a \cap S_b \neq \varnothing$. But then we have $Q \cap S_b = S_a \cap S_b = S_b$ so it is a contradiction with the second condition of definition of automata with coinciding cycles, since $S_a \cap S_b$ should induce a path on S_b. □

Observe that we can identify a letter a of an automaton as a function $a : Q \to Q$. Let Π be a set of permutation letters. As defined in [3], we say that Π has a *synchronization property* if for any non-permutation letter x, an automaton induced by the letters $\Pi \cup \{x\}$ has a synchronizing word.

4.1 Upper bound

In this section, we provide a quadratic upper bound on the length of the shortest synchronizing word in automata with coinciding cycles as well as necessary and sufficient condition for such automata to be synchronizing. From that point, we assume that $|Q| = n$, where $n \in \mathbb{N}$ and, if $a \in \Pi \subset \Sigma$, then $S_a \subset Q$ is as in the second condition of definition of automata with coinciding cycles. First, we prove a couple of lemmas.

Lemma 2. *Let X be a finite set, and $A_1, \ldots, A_k \subset X$ be a sequence of sets such that $A_i \cap A_{i+1} \neq \varnothing$. Then there exists a subsequence of this sequence $B_1, \ldots, B_l$ such that $B_1 = A_1, B_l = A_k$, $B_i \cap B_{i+1} \neq \varnothing$ and for $i, j \in \{1, \ldots, l\}$ such that $j \notin \{i-1, i, i+1\}$ we have $B_i \cap B_j = \varnothing$.*

Proof. We construct the desired subsequence inductively. Let $B_1 = A_1$. Having B_i we choose B_{i+1} to be A_j such that $A_j \cap B_i \neq \varnothing$, $j > i$ and for any $j' > j$ we have that $A_{j'} \cap B_i = \varnothing$. Since for any i holds $A_i \cap A_{i+1} \neq \varnothing$ then the construction ends with B_l equal to A_k and, since in the i-th step we have chosen maximal j for which

$B_i \cap A_j \neq \varnothing$, we know that for $i, j \in \{1, ..., l\}$ such that $j \notin \{i-1, i, i+1\}$ holds $B_i \cap B_j = \varnothing$ and that concludes the proof. □

Lemma 3. *Let $p, q \in Q$, $p \neq q$ and $a, b, c \in \Pi$ such that $S_b \cap S_c \neq \varnothing$ and $S_a \cap S_b = \varnothing$. If $p \in S_a$ and $q \in S_b$, then there exists a word $w \in \Pi^*$ such that $p.w \in S_a$, $q.w \in S_c$ and $|w| \leq |S_b| - |S_b \cap S_c|$.*

Proof. Since $S_a \cap S_b = \varnothing$, we have $p \notin S_b$. Notice that $p.b = p$ and, since $S_b \cap S_c \neq \varnothing$, there exists a word b^l such that $q.b^l \in S_b \cap S_c$ and $l \leq |S_b| - |S_b \cap S_c|$. We set $w = b^l$ and that concludes the proof. □

Lemma 4. *Let $a, b, c \in \Pi$, $p, q \in Q$ and $p \neq q$. If $p \in S_a$, $q \in S_b$, $S_a \cap S_b \neq \varnothing$ and $S_a \cap S_c \neq \varnothing$ then there exists a word $w \in \Pi^*$ such that either $p.w \in S_a \setminus S_c$ and $q.w \in S_c$ or $q.w \in S_a \setminus S_c$ and $p.w \in S_c$, and in both cases $|w| \leq |S_b| + |S_a|$.*

Proof. If $p \notin S_a \cap S_b$, then there exists a word b^l such that $p.b^l = p$, $q.b^l \in S_a \cap S_b$ and $l \leq |S_b| - |S_a \cap S_b|$. Then we set $u = b^l$. Otherwise set

$$r_1 = \text{mindist}_{\text{a,b}}(p, q)$$

and

$$r_2 = \text{maxdist}_{\text{a,b}}(p, q).$$

Observe that then there exists $k > 0$ such that $r_1.a^k \in S_a \setminus S_b$, $r_1.a^{k-1} \in S_a \cap S_b$, $k \leq |S_a \cap S_b|$ and by the definition of r_2 it follows that $r_2.a^k \in S_b$. So there exists $l > 0$ such that $r_1.a^k b^l = r_1.a^k$ and $r_2.a^k b^l \in S_a \cap S_b$ and $l \leq |S_b| - |S_a \cap S_b|$. We set $u = a^k b^l$. Now we have $p.u \in S_a \setminus S_b$ and $q.u \in S_a \cap S_b$ or vice versa ($q.u \in S_a \setminus S_b$ and $p.u \in S_a \cap S_b$) and $|u| \leq |S_b|$.

If $r_1.u \notin S_c$ and $r_2.u \notin S_c$ then we know that, since $r_2.u \neq r_1.u$ and S_a is a cycle, there exists a word a^k such that $r_1.ua^k \in S_a \setminus S_c$, $r_2.ua^k \in S_a \cap S_c$ or vice versa and $k < |S_a| - |S_a \cap S_c|$. Then we set $w = ua^k$ and we have $|w| \leq |S_b| + |S_a| - |S_a \cap S_c|$. Otherwise $r_1.u \in S_a \cap S_c$ or $r_2.u \in S_a \cap S_c$. If also $r_2.u \in S_a \setminus S_c$ or $r_1.u \in S_a \setminus S_c$

then we set $w = u$ and we know that $|w| \leq |S_b|$. If $r_1.u \in S_a \cap S_c$ and $r_2.u \in S_a \cap S_c$ then define

$$s_1 = \text{mindist}_{a,c}(r_1.u, r_2.u)$$

and

$$s_2 = \text{maxdist}_{a,c}(r_1.u, r_2.u).$$

Using a similar argument as above we obtain that there exists a word a^k such that $s_1.a^k \in S_a \setminus S_c$, $s_1.a^{k-1} \in S_a \cap S_c$, $s_2.a^k \in S_a \cap S_c$ and $k \leq |S_a \cap S_c|$. We set $w = ua^k$ with $|w| \leq |S_b| + |S_a \cap S_c|$.

Observe that the word w satisfies $p.w \in S_a \setminus S_c$ and $q.w \in S_c$ or vice versa and $|w| \leq \max\{|S_b|, |S_b| + |S_a| - |S_a \cap S_c|, |S_b| + |S_a \cap S_c|\}$ and from that inequality we deduce $|w| \leq |S_a| + |S_b|$ and the result holds. □

Using the three former lemmas, we are ready to prove the main result of this section. First, we prove the lemma stating that choosing two states and two cycles, we can always find a word of linear length which sends the chosen states to the states belonging to the chosen cycles.

Lemma 5. *Let $a, b \in \Pi$, and $S_a \subset Q$ induces a directed cycle in $\mathcal{A}$ on a and $S_b \subset Q$ induces a directed cycle in $\mathcal{A}$ on b. Let also $p, q \in Q$. Then, there exists a word $w \in \Pi^*$ such that either $p.w \in S_a$ and $q.w \in S_b$ or $q.w \in S_a$ and $p.w \in S_b$, and in both cases $|w| \leq 3n - |S_b|$.*

Proof. Let $\mathcal{B}$ be the automaton $\mathcal{A}$ restricted to Π. We are going to construct the desired word. Let $w = \epsilon$. If $p \in S_a$ and $q \in S_b$ then we are done. Assume without loss of generality that $q \notin S_b$. If also $p \notin S_a$, then, since $\mathcal{B}$ is strongly connected, we know that there exists a word $v \in \Sigma^*$ such that $p.v \in S_a$ and $|v| \leq n - |S_a|$. If also $q.v \in S_b$ then we set $w = v$ and we are done. Assume that $q.v \notin S_b$. Then, since $\mathcal{B}$ is strongly connected, there exists $c_1 \in \Pi$ such that $q.v \in S_{c_1}$. Since $\mathcal{B}$ is strongly connected, there exists a sequence of pairwise different letters $(c_2, \ldots, c_k)$, such that $S_{c_i} \cap S_{c_{i+1}} \neq \varnothing$ for $1 \leq i \leq k-1$ and $S_{c_k} \cap S_b \neq \varnothing$. Using Lemma 2, we choose

a subsequence of the sequence $(c_2, \ldots, c_k, b)$, say $(c'_1, \ldots, c'_l)$, such that $c'_1 = c_2$ and $c'_l = b$. Let us investigate three cases.

Case 1: $S_a \cap S_{c'_i} \neq \varnothing$ and $S_a \cap S_{c'_j} \neq \varnothing$ for $0 < i \neq j < l + 1$

Denote the first and the last index of the sequence $(c'_1, \ldots, c'_l)$ such that $S_{c'_i} \cap S_a \neq \varnothing$ and $S_{c'_j} \cap S_a \neq \varnothing$ as i and j (respectively). Let $w_1 = \epsilon$. We apply Lemma 3 $i - 1$ times inductively with the states $p.vw_1 \ldots w_{m-1}$ and $q.vw_1 \ldots w_{m-1}$, and the letters a, c'_{m-1}, c'_m for $m \in \{2, .., i\}$ to find a word w_m such that $p.vw_1 \ldots w_{m-1}w_m \in S_a$, $q.vw_1 \ldots w_{m-1}w_m \in S_{c'_m}$ and $|w_m| \leq |S_{c'_{m-1}}| - |S_{c'_{m-1}} \cap S_{c'_m}|$. We obtain the word $u = w_1 \ldots w_i$, such that $p.vu \in S_a$ and $q.vu \in S_{c'_i}$. Now, we use Lemma 4 to obtain a word u' such that either $p.vuu' \in S_a \backslash S_{c'_j}$ and $q.vuu' \in S_{c'_j}$ or vice versa and $|u'| \leq |S_a| + |S_{c'_i}|$. Without loss of generality, assume the former. Notice that there exists a word c'^k_j such that $p.vuu'c'^k_j \in S_a$, $q.vuu'c'^k_j \in S_{c'_j} \cap S_{c'_{j+1}}$ and $k \leq |S_{c'_j}| - |S_{c'_j} \cap S_{c'_{j+1}}|$. Now, we can again apply Lemma 3 in the same manner, to obtain a word $u'' = w_{j+2}..w_l$ such that $p.vuu'c'^k_j u'' \in S_a$ and $q.vuu'c'^k_j u'' \in S_b$. We set $w = vuu'c'^k_j u''$ and we only need to bound the length of w from above. We know that

$$|w| = |v| + |u| + |u'| + |c'^k_j| + |u''| = n - |S_a| + |w_1 w_2 \ldots w_i| + |u'| + k + |w_{j+2}..w_l| \leq$$

$$n - |S_a| + \sum_{m=1}^{i-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|) + |S_{c'_i}| + |S_a| +$$

$$|S_{c'_j}| - |S_{c'_j} \cap S_{c'_{j+1}}| + \sum_{m=j+1}^{l-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|) =$$

$$n + \sum_{m=1}^{i-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|) + |S_{c_i}| + \sum_{m=j}^{l-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|)$$

On the other hand, we know that

$$|S_{c'_1} \cup \ldots \cup S_{c'_i}| = \bigcup_{\varnothing \neq K \subseteq \{1,\ldots,i\}} (-1)^{|K|+1} |\bigcap_{k \in K} S_{c'_k}|$$

and

$$|S_{c'_j} \cup \ldots \cup S_{c'_l}| = \bigcup_{\varnothing \neq K \subseteq \{j,\ldots,l\}} (-1)^{|K|+1} |\bigcap_{k \in K} S_{c'_k}|$$

and also

$$|S_{c'_1} \cup \ldots \cup S_{c'_i}| + |S_{c'_j} \cup \ldots \cup S_{c'_l}| \leq 2n.$$

By Lemma 2, it follows that for any $K \subseteq \{1, \ldots, l\}$ different than $\{i, i+1\}$ and for $i \in \{1, \ldots, l-1\}$ we have $|\bigcap_{k \in K} S_{c'_k}| = \varnothing$. So, we know that $\sum_{m=1}^{i-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|) + |S_{c'_i}| = |S_{c'_1} \cup \ldots \cup S_{c'_i}|$ and $\sum_{m=j}^{l-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|) = |S_{c'_j} \cup \ldots \cup S_{c'_l}| - |S_{c'_l}|$ and, since $c'_l = b$, the lemma holds for that case.

Case 2: $S_a \cap S_{c'_i} = \varnothing$ for every $0 < i < l+1$

Since for every $0 < i < l+1$ we have $S_a \cap S_{c'_i} = \varnothing$, then we can apply Lemma 3 $l-1$ times to obtain the word $u = w_1 w_2 \ldots w_{l-1}$ such that $p.vu \in S_a$, $q.vu \in S_{c'_l} = S_b$ and each $|w_i| \leq |S_{c'_i}| - |S_{c'_i} \cap S_{c'_{i+1}}|$, which implies that $|vu| = n - |S_a| + \sum_{i=1}^{l-1}(|S_{c'_i}| - |S_{c'_i} \cap S_{c'_{i+1}}|)$ and, using Lemma 2, one can immediately deduce that the result holds in the similar manner as in the former case.

Case 3: $S_a \cap S_{c'_i} \neq \varnothing$ for exactly one $0 < i < l+1$

As in Case 1, we can construct a word u such that $p.vu \in S_a$, $q.vu \in S_{c'_i}$ and $|u| \leq \sum_{m=1}^{i-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|)$. We are now looking for a word u' such that either $p.vuu' \in S_a \setminus S_{c'_i}$ and $q.vuu' \in S_{c'_i}$ or $q.vuu' \in S_a \setminus S_{c'_i}$ and $p.vuu' \in S_{c'_i}$. Using a similar argument as in the proofs of the former lemmas we can find such a word with $|u'| \leq |S_a \cap S_{c'_i}|$. As in the previous cases, using Lemma 3 we now construct a word u'' such that either $p.vuu'u'' \in S_a$ and $q.vuu'u'' \in S_{c'_l}$ or $q.vuu'u'' \in S_a$ and $p.vuu'u'' \in S_{c'_l}$ with $|u''| \leq \sum_{m=i}^{l-1}(|S_{c'_m}| - |S_{c'_m} \cap S_{c'_{m+1}}|)$. The rest of the argument is analogous to the ones in the former cases. □

The next lemma states that, for any two pairs of states, we can find a word that sends the first pair to the second one.

Lemma 6. *Let $p, q \in Q$. Then, for any $r, s \in Q$ there exists a word $w \in \Pi^*$ such that either $r.w = p$ and $s.w = q$ or $s.w = p$ and $r.w = q$, and $|w| \leq 6n$.*

Proof. We investigate two cases.

Case 1: There exists $a \in \Pi$ such that $p, q \in S_a$.

The idea of the proof, in this case, is to "push" one of the states r, s "outside" the cycle induced by a to the state right behind the "entrance" to this cycle, then to set the other state in a right distance after this entrance, push the first state back to the cycle of a and then rotate those two states to obtain p and q.

Using Observation 1, we can find $b \in \Pi$ such that $S_a \cap S_b \neq \varnothing$. Now, we use Lemma 5 to obtain a word $u \in \Pi^*$ of length at most $3n$ such that $r.u \in S_a$ and $s.u \in S_b$ or $s.u \in S_a$ and $r.u \in S_b$. Without loss of generality, assume the first case. Now, we are looking for a word $v \in \Pi^*$ such that $r.uv \in S_a, s.uv \in S_b \setminus S_a$ and $s.uvb \in S_a$. If $r.u \notin S_a \cap S_b$, then it suffices to take $v = b^l$, where $l < |S_b|$. If $r.u \in S_a \cap S_b$ then define

$$t_1 = \text{mindist}_{a,b}(r.u, s.u)$$

and

$$t_2 = \text{maxdist}_{a,b}(r.u, s.u).$$

Observe that there exists a word a^k where $k \leq |S_a \cap S_b|$ such that $t_1.a^k \in S_a \setminus S_b$ and, by the definition of t_2, it follows that $t_2.a^k \in S_b$. Now, we can find a word b^l in the same manner as before and obtain that $v = a^k b^l \leq |S_b| + |S_a \cap S_b|$.

We are now looking for a word $z \in \Pi^*$, such that $\text{dist}(r.uvzb, s.uvzb, a) = \text{dist}(p, q, a)$. It is obvious that, since $|S_a \cap S_b|$ induces a path on letter b and for $t \in S_a \setminus S_b$ we have that $t.b = t$, there exists $t' \in S_a$ such that $\text{dist}(t'.b, s.uvb, a) = \text{dist}(p, q, a)$. Moreover, there exists a word a^m where $m < |S_a|$ such that $r.uvza^k = t'$, and since $s.uv \in S_b \setminus S_a$ then $s.uva^k = s.uv$. We set $u = a^m$. Since $\text{dist}(r.uvzb, s.uvzb, a) = \text{dist}(p, q, a)$, we know that there exists a word a^k where $k < |S_a|$ such that $r.uvzba^k = p$ and $s.uvzba^k = q$. Set $w = wvuba^k$ and notice that $|w| = |uvzba^k| \leq 3n - |S_b| + |S_b| + |S_a \cap S_b| + 2|S_a| \leq 6n$ and that concludes this case.

Case 2: There exist $a, b \in \Pi$ such that $p \in S_a$ and $q \in S_b$

If also $p \in S_a \cap S_b$ or $q \in S_a \cap S_b$ then we can apply Case 1 of this proof, to obtain the required word. Assume that $p \in S_a \setminus S_b$ and $q \in S_b \setminus S_a$. Using Lemma 5, we obtain a word $u \in \Pi^*$ of length at most $3n$ such that $r.u \in S_a$ and $s.u \in S_b$.

If $S_a \cap S_b = \varnothing$ then there exists a word $a^k b^l$ such that $k < |S_a|$ and $l < |S_b|$, and $r.ua^k b^l = p$ and $s.ua^k b^l = q$. Set $w = ua^k b^l$ and observe that $|w| < 3n$. So let us assume $S_a \cap S_b \neq \varnothing$. If also $r.u \in S_a \setminus S_b$ and $s.u \in S_b \setminus S_a$ then $w = ua^k b^l$ for some $k < |S_a|$ and $l < |S_b|$ and we are done. Otherwise, define

$$t_1 = \mathrm{mindist}_{a,b}(r.u, s.u)$$

and

$$t_2 = \mathrm{maxdist}_{a,b}(r.u, s.u)$$

and we can find a word a^k, where $k \leq |S_a \cap S_b|$, such that $t_1.a^k \in S_a \setminus S_b$ and $t_2.a^k \in S_b$. Then, we apply word b^l, where $l < |S_b|$, to obtain $t_2.a^k b^l = q$. Since $t_1.a^k \in S_a \setminus S_b$, then $t_1.a^k b^l = t_1.a^k$. Now, we apply word a^m, where $m < |S_a|$ to obtain $t_1.a^k b^l a^m = p$. Since $q \in S_b \setminus S_a$, then $t_2.a^k b^l a^m = q.a^m = q$. Set $w = ua^k b^l a^m$. Observe that $k + l + m \leq |S_a| + |S_b| + |S_a \cap S_b|$, which concludes the proof. □

Theorem 6. *Let $\mathcal{A}$ be an automaton with coinciding cycles. Then $\mathcal{A}$ is synchronizing if and only if there exist $x \in \Sigma$ and $p, q \in Q$ such that $p.x = q.x$. The shortest synchronizing word for such an automaton is of length at most $1 + (6n + 1)(n - 2)$.*

Proof. First, we prove "⇒" implication. For the sake of contradiction, suppose that $\mathcal{A}$ is synchronizing, and there is no such letter x. But that would mean that every letter in Σ is a bijection, so for any $a \in \Sigma$ we have that $Q.a = Q$, which is the contradiction.

To prove "⇐" implication, assume that there exist $x \in \Sigma$ and $p, q \in Q$ such that $p.x = q.x$. Notice that then $|Q.x| < n$. Then, we can act as in the proof of Theorem 1 ([56] (Theorem 1)) and use Lemma 6 to obtain words $w_i x$ such that $|Q.w_1 x \dots w_{i-1} x w_i x| < |Q.w_1 x \dots w_{i-1} x|$. Indeed, observe that for any $i \in \{0, \dots, n-1\}$ either there exist $r, s \in Q.w_1 x \dots w_{i-1} x w_i$ such that $\{r, s\}.w_i = \{p, q\}$ or $|Q.w_1 x \dots w_{i-1} x w_i| = 1$. The bound from Lemma 6 at most $n - 1$ times concludes the proof. □

Now we can state a straightforward conclusion from the previous theorem.

Corollary 3. *Any automaton with coinciding cycles has a synchronization property.*

Proof. Immediate from Theorem 6, since any non-permutation letter x has $p, q \in Q$ such that $p.x = q.x$. □

4.2 Worst case lower bound

We use the result from [19] to show, that we cannot significantly improve the result from the previous section. First, we recall the construction and the theorem showing the lower bound for the shortest synchronizing word of a family of automata arising from that construction. Next, we prove that this family fulfills the conditions of automata with coinciding cycles. We start with the construction.

Let $Q_n = \{q_0, .., q_{n-1}\}$ and $\Sigma = \{a_1, ..., a_n\}$ for $n \in \mathbb{N}$. Define an automaton $\mathscr{V}_n = (Q_n, \Sigma_n, \delta_n)$ with the transition function δ_n as follows:

- $\delta_n(q_i, a_j) = q_i$ for $0 \leq i < n$, $1 \leq j < n$, $i \neq j$, $j \neq i + 1$, $i \neq n$
- $\delta_n(q_i, a_i) = q_{i-1}$ for $0 < i \leq n - 1$
- $\delta_n(q_i, a_{i+1}) = q_{i+1}$ for $0 \leq i < n - 1$
- $\delta_n(q_0, a_n) = \delta_n(q_1, a_n) = q_0$
- $\delta_n(q_i, a_n) = q_i$ for $2 \leq i \leq n - 1$

We recall the result from [19] establishing the length of the shortest synchronizing word for the DFA $\mathscr{V}_n$. The DFA $\mathscr{V}_5$ is depicted in Fig. 17 (self-maps $a_1, \ldots, a_4$ are omitted).

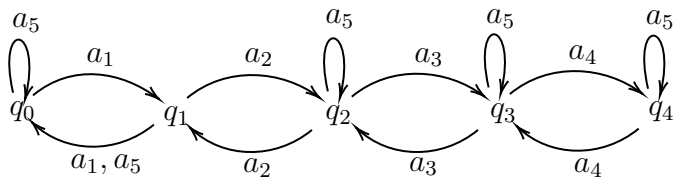


Figure 17: The automaton $\mathscr{V}_5$

Theorem 7 (Theorem 4 in [19]). *The length of the shortest synchronizing word for the automaton $\mathscr{V}_n$ is $\frac{(n-1)n}{2}$.*

Having that, we are ready to prove the following lemma.

Lemma 7. *For $n > 2$ the automaton $\mathscr{V}_n$ is the automaton with coinciding cycles.*

Proof. First, notice that the letter a_n is a non-permutation letter. Define $\Pi_n = \Sigma_n \setminus \{a_n\}$. Since $n > 2$ then, by the definition of $\mathscr{V}_n$, we have that $|\Pi_n| > 1$. Consider the letter $a_i \in \Pi_n$. By the definition, we obtain that for any $j \notin \{i-1, i\}$ letter a_i acts as a self-map on q_j and acts as a cyclic permutation on the set $\{q_{i-1}, q_i\}$ and that proves the first condition.

Define $S_i = \{q_{i-1}, q_i\}$ for $1 \leq i \leq n$. We have already proven that a_i induces a directed cycle on S_i. It is obvious that, for any $i \neq j$, we have that $S_i \cap S_j \neq \varnothing$ if and only if $j = i - 1$ or $j = i + 1$. In any of these cases $|S_i \cap S_j| = 1$, so the intersection of these sets must be an isolated vertex and that proves the second condition.

For the proof of the third condition it suffices to notice that for any $i \neq j$ we have that $q_i.a_{i+1}a_{i+2}..a_j = q_j$ and $q_j.a_ja_{j-1}..a_{i+1} = q_i$, which concludes the proof. □

Since we have shown the existence of a family of automata in the class of automata with coinciding cycles for which the shortest synchronizing words are of length $\Theta(n^2)$, we are certain that we cannot improve the bound from the previous section significantly, i.e., the bound proven in the Section 4.1 is asymptotically strict.

4.3 Summary

We have presented the class of automata with coinciding cycles and we have proven a quadratic upper bound for the length of the shortest synchronizing word for it. Moreover, we have shown that every automaton from this class has a synchronization property. We have also found the evidence that the upper bound for synchronizing word we have established, is asymptotically strict and we cannot improve the result by more than a constant.

5 Careful synchronization of one-cluster automata

Before proceeding, we state the preliminary observation and lemma that can shed more light on the synchronization and careful synchronization of one-cluster automata.

Observation 3. *If a one-cluster automaton $\mathcal{A} = (Q, \Sigma, \delta)$ is carefully synchronizing, C induces a directed cycle on the letter a, the letter a is the only one defined for all states and $|C| > 1$, then there exists a letter $b \in \Sigma$ such that $b \neq a$ and for any $q \in C$ it holds that $\delta(q, b)$ is defined.*

Proof. Obviously, since a is the only letter defined for all states, then every synchronizing word must start with it. On the other hand, for all $k \in \mathbb{N}$ holds $C \subseteq Q.a^k$ and if $k \geq l$, where l is the level of $\mathcal{G}_a$, then $Q.a^k = C$. If $|C| > 1$, then there must exist another letter $b \in \Sigma$ such that $C.b \neq C$ for $\mathcal{A}$ to be carefully synchronizing. □

Suppose that C induces a directed cycle on the set of states of a PFA. We state and prove the lemma that shows that if there exists a word w shrinking C, then there exists a word w' (whose length is polynomial in $|w|$) shrinking C by half.

Lemma 8. *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a one-cluster PFA, $|Q| = n$. Let C induce a directed cycle on the letter a and l be the level of $\mathcal{G}_a$. If there exists a word w such that $|C.w| < |C|$ then there exists a word w' such that $|Q.w'| \leq \lfloor \frac{1}{2}|C| \rfloor$ of length at most $n + \frac{1}{2}|C|(|w| + |C|)$.*

Proof. For any $p, q \in C$ define $\mathrm{dist}_C(p, q) = \min\{k_1, k_2 : p.a^{k_1} = q \wedge q.a^{k_2} = p \wedge k_1, k_2 < |C|\}$. Since $p, q \in C$, then $dist_C(p, q)$ is always well defined and for all p, q holds $dist_C(p, q) \leq \lfloor \frac{1}{2}|C| \rfloor$. Since there exists w such that $|C| > |C.w|$, then there must exist $\bar{p}, \bar{q} \in C$ such that $\bar{p}.w = \bar{q}.w$. Denote $\mathrm{dist}_C(\bar{p}, \bar{q}) = k$. We will show the claim:

Claim 1. *For any $C' \subset C$ such that $|C'| > \frac{1}{2}|C|$, there exist $p', q' \in C$, such that $dist(p', q') = k$.*

Proof. For the sake of contradiction, suppose that there exists $C' \subset C$ and $|C'| > \frac{1}{2}|C|$ such that for all $p, q \in C'$ holds $\mathrm{dist}(p, q) \neq k$. Let $S = \{q_{i+k \bmod m} : q_i \in C'\}$. Since for any $q_i \in C'$ there exists exactly one state $q_{i+k \bmod m} \in C$, then $|C'| = |S|$. On the other hand, since for all $p, q \in C'$ holds $\mathrm{dist}(p, q) \neq k$, then $C \cap S = \varnothing$. But since $|C'| = |S| > \frac{1}{2}|C|$ and $C' \subset C$ and $S \subset C$ then $|C'| + |S| > |C|$, so there must hold $C' \cap S \neq \varnothing$ and we have a contradiction. □

We can construct the word w' in the following way: Obviously $Q.a^l = C$ and $|Q.a^l w| < |C|$. If also $|Q.a^l w| \leq \lceil \frac{1}{2}|C| \rceil$ then the result holds. Otherwise, observe that $Q.a^l w a^l \subset C$ and $|Q.a^l w a^l| > \frac{1}{2}|C|$ so we can apply Claim 1 to find the states $p', q' \in Q.a^l w a^l$ such that $\mathrm{dist}(p', q') = k = \mathrm{dist}(\bar{p}, \bar{q})$. Denote $u = a^l w a^l$ and observe that, for some $m_1 \leq |C| < n$, it must hold that $\bar{p}, \bar{q} \in Q.ua^{m_1}$ so $|Q.ua^{m_1} w| < |Q.u|$. We can apply Claim 1 and the word $a^{m_1} w$ as long as the size of the resulting set is greater than $\frac{1}{2}|C|$. We obtain the word $w' = a^l w (a^{m_1} w)^m_2$ where $m_2 < \frac{1}{2}|C| - 1$ so the result holds. □

5.1 Long carefully synchronizing words

In this section, we construct an infinite family of carefully synchronizing, one-cluster PFAs with exponential shortest carefully synchronizing word. The scheme of the proof is similar to the one given in [25] (Proposition 8), however it differs in details. To simplify proofs, we assume that the size of the set of states is $n = 2k$, where $k \in \mathbb{N}$ but the arguments can be easily adapted to the case $n = 2k+1$. Let $\mathcal{B}_k = (Q_k, \Sigma_k, \delta_k)$. with $Q_k = C_k \cup T_k$ where $C_k = \{c_1, \ldots, c_k\}$, $T_k = \{t_1, \ldots, t_k\}$ and $\Sigma_k = \{a\} \cup \Sigma'_k$ where Σ'_k is a set of letters specified later on. Define the action of the letter a on the set of states as:

- $\delta_k(c_i, a) = c_{i+1}$ for $i < k$
- $\delta_k(c_n, a) = c_1$

- $\delta_k(t_i, a) = c_i$ for $i \in \{1, \ldots, k\}$

It is easy to observe that $\mathcal{B}_k$ is one-cluster with respect to a for every k. An example of that cluster for $k = 3$ is depicted below. 18.

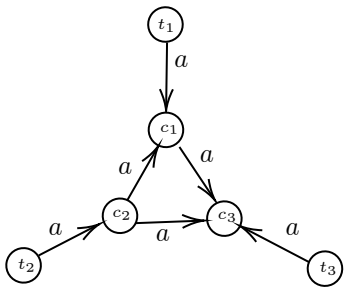


Figure 18: The a-cluster of the automaton $\mathcal{B}_3$

Let $\mathcal{T}_k = \{T \cup (C_k \cap ((T_k \setminus T).a).a^{-1}) : T \in 2^{T_k} \setminus \{\varnothing\}\}$ – (recall that 2^{T_k} stands for the set of all subsets of T_k). Intuitively, it is the set of all sets T such that $|T'| = k$ and $T'.a = C_k$. First, we prove the following lemma.

Lemma 9. *Let $\mathcal{T}_k$ be the family of sets defined above. Then $|\mathcal{T}_k| = 2^{|T_k|} - 1$.*

Proof. Let $T'_1, T'_2 \in 2^{T_k}$ such that $T'_1 \neq T'_2$. Denote $C_k \cap ((T_k \setminus T'_1).a).a^{-1} = P_1$ and $C_k \cap ((T_k \setminus T'_2).a).a^{-1} = P_2$ It suffices to show that, for any T'_1, T'_2, the sets $T'_1 \cup P_1$ and $T'_2 \cup P_2$ are different. But it is easy to notice that $P_1, P_2 \subset C_k$ and since $T'_1 \neq T'_2$ and $T'_1, T'_2 \subset T_k$, then the lemma holds. □

Using Lemma 9, we can enumerate sets of the family $\mathcal{T}_k$ arbitrarily from S_1 to $S_{2^{|T_k|}-1}$. Let us prove some properties of those sets in the next lemma.

Lemma 10. *For any $i \in \{1, \ldots, 2^{|T_k|} - 1\}$ holds $S_i.a = C_k$ and $|S_i| = \frac{n}{2}$.*

Proof. By the definition of S_i, we know that there exists $T \subseteq T_k$ such that $S_i = T \cup (C_k \cap ((T_k \setminus T).a).a^{-1})$. By the construction $T.a \subseteq C$ so it suffices to show that $(C \cap ((T_k \setminus T).a).a^{-1}).a = C \setminus T.a$. Indeed $(C_k \cap ((T_k \setminus T).a).a^{-1}).a = C_k.a \cap (T_k \setminus T).a = C_k \cap T_k.a \setminus T.a = C \cap C \setminus T.a = C \setminus T.a$. To show that $|S_i| = \frac{n}{2}$ first observe that

$(T_k \setminus T).a = \{q \in C_k \setminus T.a\}$. Let $q \in C_k \setminus T.a$ and observe that the set $q.a^{-1}$ contains exactly two states $q_1 \in T_k$ and $q_2 \in C_k$ and by the construction we have that $|((T_k \setminus T).a).a^{-1}| = 2|C_k \setminus T.a|$ and the half of the states must be contained in C_k, which gives us $|S_i| = |T.a| + |C_k \setminus T.a|$, and that concludes the proof. □

Let $\Sigma'_k = \{b_1, \ldots, b_{2^{|T_k|}-1}, c\}$ (this is the subalphabet used in the definition of Σ_k) and $S_i = \{s_1^i, \ldots, s_k^i\}$. Put the transition function δ_k (for Σ'_k) as follows:

- $\delta_k(c_j, b_1) = s_j^1$, the letter b_1 not defined for other states
- for $0 < i < 2^{|T_k|} - 2$ let $\delta_k(s_j^i, b_{i+1}) = s_j^{i+1}$, the letter $b_i + 1$ not defined for other states
- $\delta_k(s_j^{2^{|T_k|}-1}, c) = c_1$, the letter c not defined for other states

Observe that, for any i, holds $\delta_k(S_i, b_{i+1}) = S_{i+1}$. Let us state and prove the main theorem of this section.

Theorem 8. *The automaton $\mathcal{B}_k$ is carefully synchronized by the word $w = ab_1 \ldots b_{2^{|T_k|}-1}c$, w is the shortest word that carefully synchronizes $\mathcal{B}_k$ and $|w| = 2^{\frac{n}{2}} + 1$.*

Proof. Since $\mathcal{B}_k$ is one-cluster with respect to a and the level of that cluster is 1, it is straightforward that $Q.a = C$. Immediately from the construction, we have that, for $w_i = ab_1 \ldots b_i$, the equality $Q.w_i = S_i$ holds. From that, we have that $Q.w = c_1$, so w carefully synchronizes $\mathcal{B}_k$. To prove the minimality of w, first notice that, for each i, holds that $S_i.b_j$ is undefined for $j \neq i-1$ (because each $|S_i| = \frac{n}{2}$ from Lemma 10) and $S_i.a = C$ (by Lemma 10). Then $\mathcal{P}(\mathcal{B}_k)$ forms a path from Q to $\{c_1\}$ labelled with the consecutive letters of w, and for each state on the path, there is only one transition leading to C (visited in the first step of the path) and that ends the proof. □

5.2 Complexity

This section is devoted to the proof that the problem of deciding whether a given one-cluster PFA is carefully synchronizing is NP-hard, even for the binary alphabet.

Notice that Observation 3 implies that the letter b must be defined for at least all the states in the cycle induced by the letter a. State the problem:

2-ONE-CLUSTER-CARSYNC
Input: A one-cluster PFA $\mathcal{A} = (Q, \{a, b\}, \delta)$
Question: Is $\mathcal{A}$ carefully synchronizing?

Let us give the main theorem formally.

Theorem 9. *For a given PFA $\mathcal{A} = (Q, \{a, b\}, \delta)$ that is one-cluster with respect to a, the problem of deciding whether $\mathcal{A}$ has a carefully synchronizing word (2-ONE-CLUSTER-CARSYNC) is NP-hard.*

We construct a polynomial time reduction from 3-SAT to 2-ONE-CLUSTER-CARSYNC to prove the theorem. Let $\{x_1, \ldots, x_n\}$ be a set of variables and $C_1, \ldots, C_m$ be clauses. We also assume that for any pair $\{x_j, \bar{x}_j\}$ if $x_j \in C_i$ then $\bar{x}_j \notin C_i$. That is not really a problem since if both of them belong to C_i, then C_i is always true. For a given formula $\phi = C_1 \wedge \ldots \wedge C_m$, we define the PFA $\mathcal{A}_\phi = (Q, \{a, b\}, \delta_\phi)$. Let $Q = \bigcup_{i=1}^{m} (S_{C_i^t} \cup S_{C_i^f}) \cup P \cup R$ where:

- $P = \{p_1, \ldots, p_m\}$
- $R = \{r_1, \ldots, r_m\}$
- $S_{C_i^t} = \{c_i^{x_1}, \ldots, c_i^{x_n}, x_i^{end}\}$ for each i
- $S_{C_i^f} = \{\bar{c}_i^{x_1}, \ldots, \bar{c}_i^{x_n}, \bar{x}_i^{end}\}$ for each i.

Define δ_ϕ in the following way:

1. $\delta_\phi(p_i, a) = p_{i+1 \bmod m}$
2. $\delta_\phi(p_i, b) = \bar{c}_i^{x_1}$
3. $\delta_\phi(r_i, a) = p_i$, $\delta_\phi(r_i, b)$ undefined

4. $\delta_\phi(c_i^{end}, a) = \delta_\phi(\bar{c}_i^{end}, a) = r_i$

5. $\delta_\phi(c_i^{end}, b) = p_1$ $\delta_\phi(\bar{c}_i^{end}, b)$ undefined

6. if $x_j \in C_i$ then $\delta_\phi(\bar{c}_i^{x_j}, a) = c_i^{x_{j+1}}$ otherwise $\delta_\phi(\bar{c}_i^{x_j}, a) = \bar{c}_i^{x_{j+1}}$ for $0 < j < n$

7. if $\bar{x}_j \in C_i$ then $\delta_\phi(\bar{c}_i^{x_j}, b) = c_i^{x_{j+1}}$ otherwise $\delta_\phi(\bar{c}_i^{x_j}, b) = \bar{c}_i^{x_{j+1}}$ for $0 < j < n$

8. $\delta_\phi(c_i^{x_j}, a) = \delta_\phi(c_i^{x_j}, b) = c_i^{x_{j+1}}$ for $0 < j < n$

9. if $x_n \in C_i$ then $\delta_\phi(\bar{c}_i^{x_n}, a) = c_i^{end}$ otherwise $\delta_\phi(\bar{c}_i^{x_n}, a) = \bar{c}_i^{end}$

10. if $\bar{x}_n \in C_i$ then $\delta_\phi(\bar{c}_i^{x_n}, b) = c_i^{end}$ otherwise $\delta_\phi(\bar{c}_i^{x_n}, b) = \bar{c}_i^{end}$

11. $\delta_\phi(c_i^{x_n}, a) = \delta_\phi(c_i^{x_n}, b) = c_i^{end}$

For the exemplary formula $\phi_{ex} = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4)$, the corresponding $\mathcal{A}_{\phi_{ex}}$ is depicted in Fig. 19. Let us state and prove two lemmas before going further.

Lemma 11. *For any given ϕ, the automaton $\mathcal{A}_\phi$ is one-cluster with respect to a.*

Proof. First, observe that for any ϕ, the set P forms the a-cycle in the automaton $\mathcal{A}_\phi$ (Point 1 of the δ_ϕ definition). It is also easy to see from the definition of $\mathcal{A}_\phi$ that any set $S_{C_i^t}$ together with the states $r_i \in R$ and $p_i \in P$ create a directed path of the form $c_i^{x_1} \to c_i^{x_2} \to \ldots \to c_i^{x_n} \to r_i \to p_i$ labelled with a (Points 3, 4 and 8 of the definition of δ_ϕ). Denote this path as p. Denote by v_j a variable of the formula ϕ and $v_j \in \{x_j, \bar{x}_j\}$. Consider now the clause $C_i \in \phi$ and the set $S_{C_i^f}$. Assume that $C_i = (v_{j_1} \vee v_{j_2} \vee v_{j_3})$ and $j_1 < j_2 < j_3$. Any of the variables $v_{j_3+1}, \ldots, v_n$ do not belong to the clause C_i, so the set $\{\bar{c}_i^{x_{j_3+1}}, \bar{c}_i^{x_{j_3+2}}, \ldots, \bar{c}_i^{x_n}, \bar{c}_i^{end}, r_i\}$ forms a directed path on the letter a (Points 4, 6, 7, 9, 10 of the definition of δ_ϕ). Denote it by p_1. Suppose that $v_{j_3} = \bar{x}_{j_3}$. Then, by Points 6 or 9, state $\bar{c}_i^{x_{j_3}}$ is attached to the first state of the path p_1 with the transition labelled by the letter a. Otherwise, by Points 6 or 9, $\bar{c}_i^{x_{j_3}}$ is attached to the path p by the letter a. Now, we can repeat our argument to the set

$\{\bar{c}_i^{x_{j_2+1}}, \ldots \bar{c}_i^{x_{j_3-1}}\}$ to show that it forms the path p_2 labelled with the letter a which begins in the state $\bar{c}_i^{x_{j_2+1}}$ and ends in the state $\bar{c}_i^{x_{j_3-1}}$. By Point 6, we obtain that this path is attached by its end to the state $\bar{c}_i^{x_{j_3}}$ by the letter a. Now, we can also repeat our arguments to the state $\bar{c}_i^{x_{j_2+1}}$ to show that it is either attached to p_2 or to p. The same reasoning for the sets $\{\bar{c}_i^{x_{j_1+1}}, \ldots \bar{c}_i^{x_{j_2-1}}\}$ and $\{\bar{c}_i^{x_1}, \ldots \bar{c}_i^{x_{j_2-1}}\}$ and the state $\bar{c}_i^{x_{j_1}}$ concludes the proof. □

Lemma 12. *For any given ϕ, if $n_1 < n + 3$ then $\delta_\phi(Q, a^{n_1}b)$ is undefined.*

Proof. Observe that the letter a induces a path on $n + 3$ vertices on the set $S_{C_i^t} \cup \{r_i, p_i\} = S_i$ for each i. That path starts in $c_i^{x_1}$, ends in p_i and the one before last vertex on that path is r_i. Also $Q = \bigcup_{i=1}^{m} S_i$, so for each $n_1 < n + 3$ holds $r_i \in Q.a^{n_1}$ for any $0 < i < m + 1$. On the other hand, $\delta_\phi(r_i, b)$ is undefined for each i (Point 3), so $\delta_\phi(Q, a^{n_1}b)$ is undefined and the lemma holds. □

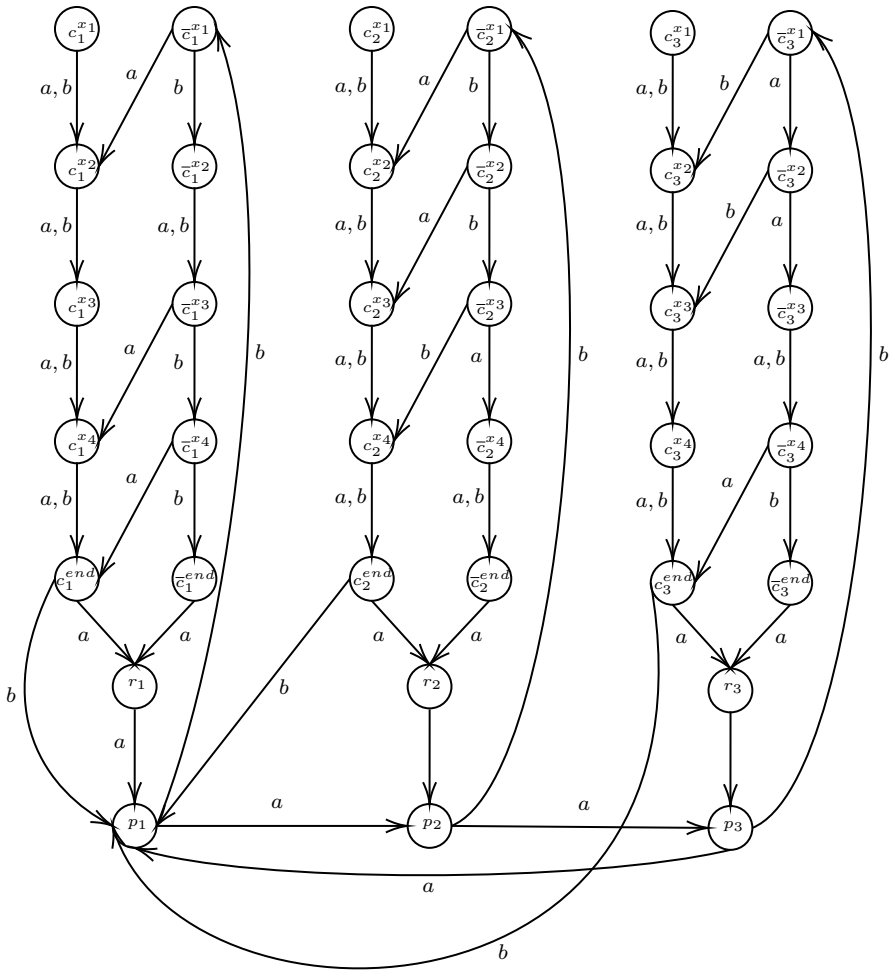


Figure 19: The automaton $\mathcal{A}_{\phi_{ex}}$

Before moving further let us define, for a given $\mathcal{A}_\phi$, two subsets of its states $S_{init} = \{\bar{c}_1^{x_1}, \ldots, \bar{c}_m^{x_1}\}$ and $S_{end} = \{c_1^{end}, \ldots, c_m^{end}\}$ and state the following observation.

Observation 4. *For any $i \in \{1, \ldots, m\}$ and a formula ϕ, if $v \in \{a, b\}^*$ and $|v| = n$, then $\delta_\phi(\bar{c}_i^{x_1}, v) \in \{c_i^{end}, \bar{c}_i^{end}\}$.*

Proof. Immediate from Points 6 to 11 of the definition of δ_ϕ. □

Lemma 13. *For any given ϕ, the automaton $\mathcal{A}_\phi$ is carefully synchronizing if and only if there exists a word w of length n such that $\delta_\phi(S_{init}, w) = S_{end}$.*

Proof. First, assume that there exists a word w of length n such that $\delta_\phi(S_{init}, w) = S_{end}$. Observe that $\delta_\phi(Q, a^{n+3}) = P$ and $\delta_\phi(P, b) = S_{init}$. Also $\delta_\phi(S_{init}, w) = S_{end}$ and $\delta_\phi(S_{end}, b) = p_1$ so we conclude that the word $u = a^{n+3}bwb$ carefully synchronizes $\mathcal{A}_\phi$. Now, assume that $\mathcal{A}_\phi$ is carefully synchronizing. From Lemma 12 we obtain that any carefully synchronizing word for $\mathcal{A}_\phi$, say $u \in \{a, b\}^*$, must start with the word a^k where $k > n + 2$. Since $\delta_\phi(Q, a^{n+3}) = P$ and $\delta_\phi(P, a) = P$ we imply that u must be of the form $a^k bv$. We know that $\delta_\phi(Q, a^k b) = S_{init}$. From Observation 4, for any $v_1 \in \{a, b\}^*$ such that $|v_1| = n$ we have $\delta_\phi(\bar{c}_i^{x_1}, v_1) \in \{c_i^{end}, \bar{c}_i^{end}\}$. On the other hand, observe that for any set of the form $\{t_1, \ldots, t_n\}$, where $t_i \in \{c_i^{end}, \bar{c}_i^{end}\}$ but S_{end}, we have that $\delta_\phi(T, b)$ is undefined, $\delta_\phi(T, a) = R$ and $\delta_\phi(R, b)$ is undefined, and $\delta_\phi(R, a) = P$. So, since u carefully synchronizes $\mathcal{A}_\phi$, there must exist a desired word, and that concludes the proof. □

Lemma 14. *Let ϕ be a formula and $\mathcal{A}_\phi$ be the corresponding automaton. Then, ϕ has a truth assignment if and only if there exists a word w of length n such that $\delta_\phi(S_{init}, w) = S_{end}$.*

Proof. First, assume that ϕ has a truth assignment $e : \{x_1, \ldots, x_n\} \to \{0, 1\}$. We define the word w of length n in the following way: if $e(x_i) = 1$ in that evaluation then the i-th letter of the word w is a, otherwise the i-th letter is b. Note w_i as the $i - 1$ letter prefix of w and $\bar{w}_i$ as the $i - 1$ letter suffix of w. Let $0 < k_1 < k_2 < k_3 < n + 1$ and consider the clause $C_j = y_{k_1} \vee y_{k_2} \vee y_{k_3}$ where $y_{k_i} \in \{x_{k_i}, \bar{x}_{k_i}\}$. Since e is a truth evaluation, at least one of the variables x_{k_1}, x_{k_2} or x_{k_3} must be evaluated as **true** in the sense that if $y_{k_i} = x_{k_i}$ then $e(x_{k_i}) = 1$, otherwise $e(x_{k_i}) = 0$. It is straightforward from the definition of δ_ϕ (Points 6 and 7) that $\delta_\phi(\bar{c}_j^{x_1}, w_{k_1}) = \bar{c}_j^{x_{k_1}}$. Now consider four pairwise exclusive cases:

1. $y_{k_1} = x_{k_1}$ and $e(x_{k_1}) = 1$
2. $y_{k_1} = x_{k_1}$ and $e(x_{k_1}) = 0$
3. $y_{k_1} = \bar{x}_{k_1}$ and $e(x_{k_1}) = 1$

4. $y_{k_1} = \bar{x}_{k_1}$ and $e(x_{k_1}) = 0$.

Now:

- if Case 1 holds, then the i-th letter of w is a and, by Point 6, we know that $\delta_\phi(\bar{c}_j^{x_{k_1}}, a) = c_j^{x_{k_1+1}}$
- if Case 2 holds, then the i-th letter of w is b and, by Point 7, we know that $\delta_\phi(\bar{c}_j^{x_{k_1}}, b) = \bar{c}_j^{x_{k_1+1}}$
- if Case 3 holds, then the i-th letter of w is a and, by Point 6, we know that $\delta_\phi(\bar{c}_j^{x_{k_1}}, a) = \bar{c}_j^{x_{k_1+1}}$
- if Case 4 holds, then the i-th letter of w is b and, by Point 7, we know that $\delta_\phi(\bar{c}_j^{x_{k_1}}, b) = c_j^{x_{k_1+1}}$.

If Case 1 or 4 hold, then $\delta_\phi(\bar{c}_j^{x_1}, w_{i+1}) = c_j^{x_{k_1+1}}$ and, from Points 8 and 11, we obtain that $\delta_\phi(c_j^{x_{k_1+1}}, \bar{w}_{n-k_1}) = c_j^{end}$. Otherwise, we obtain that $\delta(\bar{c}_j^{x_1}, w_{k_2}) = \delta(\bar{c}_j^{x_{k_2}})$ and we can repeat our case analysis to obtain that either $\delta(\bar{c}_j^{x_1}, w_{k_2+1}) = c_j^{x_{k_2+1}}$ (if y_{k_2} is evaluated as **true**) or $\delta(\bar{c}_j^{x_1}, w_{k_2+1}) = \bar{c}_j^{x_{k_2+1}}$ (if y_{k_2} is evaluated as **false**). With the same argument applied to y_{k_3}, since at least one of the variables of C_j must be evaluated as **true**, we obtain that $\delta_\phi(\bar{c}_j^{x_1}, w) = c_j^{end}$. But since e is a truth evaluation, then any C_j must have at least one variable evaluated as **true**, so for any $j \in \{1, \ldots, m\}$ we have that $\delta_\phi(\bar{c}_j^{x_1}, w) = c_j^{end}$, so we obtain that $\delta(S_{init}, w) = S_{end}$.

Now, assume that there exists a word w of length n, such that $\delta(S_{init}, w) = S_{end}$. We define the evaluation of ϕ, $e_w : \{x_1, \ldots, x_n\} \to \{0, 1\}$ in the following way: if the i-th letter of w is a, then $e_w(x_i) = 1$ otherwise $e_w(x_i) = 0$. Since $\delta_\phi(S_{init}, w) = S_{end}$, then from Observation 4 for any $j \in \{1, \ldots, m\}$ holds $\delta_\phi(\bar{c}_j^{x_1}, w) = c_j^{end}$. Consider e_w, let $0 < k_1 < k_2 < k_3 < n+1$ and the clause $C_j = y_{k_1} \vee y_{k_2} \vee y_{k_3}$ where $y_{k_i} \in \{x_{k_i}, \bar{x}_{k_i}\}$. It suffices to prove that for any j if $\delta_\phi(\bar{c}_j^{x_1}, w) = c_j^{end}$, then $e_w(y_{k_1}) = 1$ or $e_w(y_{k_1}) = 1$ or $e_w(y_{k_1}) = 1$ (we assume here that $e(\bar{x}) = 1 - e(x)$). For the sake of contradiction suppose, that there exists $j \in \{1, \ldots, m\}$ such that $\delta_\phi(\bar{c}_j^{x_1}, w) = c_j^{end}$ and $e_w(y_{k_1}) = 0$

and $e_w(y_{k_2}) = 0$ and $e_w(y_{k_3}) = 0$. First, observe that $\delta_\phi(\bar{c}_j^{x_1}, w_{k_1}) = \bar{c}_j^{x_{k_1}}$. Since $e(y_{k_1}) = 0$, then (from Points 6 and 7) we obtain that $\delta_\phi(\bar{c}_j^{x_1}, w_{k_1+1}) = \bar{c}_j^{x_{k_1+1}}$. The same argument for y_2 gives us $\delta_\phi(\bar{c}_j^{x_1}, w_{k_2+1}) = \bar{c}_j^{x_{k_2+1}}$ (from Points 6 and 7) and further for y_3 gives us $\delta_\phi(\bar{c}_j^{x_1}, w_{k_3+1}) = \bar{c}_j^{x_{k_3+1}}$ or $\delta_\phi(\bar{c}_j^{x_1}, w_{k_3+1}) = \bar{c}_j^{end}$ (respectively from Points 6 and 7 or 10 and 11). In the latter case, we obtain a contradiction straightforward. In the former case, we notice that, for any $v \in \{a, b\}^*$ such that $|v| = n - k_3$, we obtain that $\delta_\phi(\bar{c}_j^{x_{k_3}}, v) = \bar{c}_j^{end}$ (from Points 6, 7,10 and 11 and the observation that any variable with the index greater than k_3 does not belong to the clause C_j). That concludes the proof. □

Combining Lemmas 13 and 14, we obtain that the automaton $\mathcal{A}_\phi$ is carefully synchronizing if, and only if, the formula ϕ has a truth evaluation. Taking Lemma 11 into account, the only thing we have to show to finish the proof of Theorem 9 is that the reduction can be performed in polynomial time. Observe that, for any ϕ with n variables and m clauses and corresponding $\mathcal{A}_\phi$, we have that $|P| = |R| = m$ and for any $i \in \{1, \ldots, m\}$ also holds $|S_{C_i^t}| = |S_{C_i^f}| = n + 1$, we obtain, that $|Q| = 2m(n + 2)$ and on the other hand $\mathcal{A}_\phi$ is a partial automaton, what means, that for any state there are at most two outgoing transitions (a and b), so computing the automaton $\mathcal{A}_\phi$ can be done in polynomial time of n and m and that concludes the proof of Theorem 9.

5.3 Remarks

In Section 5.1, we have found an infinite family of PFAs with the shortest carefully synchronizing word of exponential length. However, the size of the alphabet used in the construction is also exponential in terms of the number of states, which makes the construction insufficient to analyze the decision problem stated in Section 5.2. An interesting problem to investigate is whether one can achieve the shortest carefully synchronizing word for one-cluster PFAs of exponential length using a smaller alphabet size.

In Section 5.2, we have proven NP-hardness of the problem of deciding whether a given binary one-cluster PFAs can be carefully synchronized. Let $\mathcal{A}$ be a one-cluster PFA with respect to the letter a and let C be the set of states that induces a cycle on the letter a. Further analysis of the proof leads to a conclusion that even the problem of deciding whether there exists a word w such that $|C.w| < |C|$ is NP-hard. Let us make a simple observation:

Observation 5. *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a PFA, and $w \in \Sigma^*$. There exists an algorithm that, in $O(|w||Q|)$ time, decides whether w is a carefully synchronizing word for $\mathcal{A}$.*

Proof. Consider the algorithm defined below. Notation $w[i]$ in that algorithm stands for the i-th letter of the word w.

Algorithm 2 Algorithm for deciding if a given w carefully synchronizes a given PFA $\mathcal{A}$

```
1: P ← Q
2: for i = 1 to i = |w| do
3:     P ← P.w[i]
4: end for
5: if |P| = 1 then
6:     return true
7: else
8:     return false
9: end if
```

Obviously, the algorithm answers true if and only if w carefully synchronizes $\mathcal{A}$ and requires $O(|Q|)$ additional space complexity and $O(|Q||w|)$ time complexity, what concludes the proof. □

From Observation 5, we can infer that one possibility to prove that the problem 2-ONE-CLUSTER-CARSYNC is in NP is to show that the shortest carefully synchronizing word for binary one-cluster automata is of polynomial length. It is a problem

we want to investigate further. Lemma 8 can be utilized to obtain that result. One must show first that if there exists a word w such that $|C.w| < C$ where C is the set of states inducing a cycle on the letter a, then such a word is of polynomial length in terms of the set of states. The next question is how to synchronize the remaining half of the states of the cycle.

6 Asymmetric cryptosystem utilizing careful synchronization

Cryptography is the essential branch of mathematics since the ancient times. Its main purpose is to ensure the privacy of information between a sender and a receiver sent through a possibly observed channel. Nowadays we distinguish between the symmetric cryptography (where the key used to cipher the message is the same as the one to decipher it) and the asymmetric cryptography (where the key to cipher the message is commonly known and the one to decipher it is known only to the receiver of the message). In other words, the asymmetric cryptography is referred to as the public-key cryptography, or the public-private key cryptography. The idea of the public key cryptography was first mentioned in a confidential report GCHQ [13] (UK Government Communications Headquarters) and later independently by Diffie and Hellman in 1976 [10], along with the first practical public key cryptosystem based on the knapsack problem. The most recognizable asymmetric cryptosystem, RSA, was invented by Rivest, Shamir and Adleman in 1978 [38] and is applicable since then to encryption and digital signatures.

In the *Introduction to Modern Cryptography* ([28]) the authors indicate that "... until the 20th century (and arguably until late in that century), cryptography was an art. Constructing good codes, or breaking existing ones, relied on creativity and personal skill. There was very little theory that could be relied upon and there was not even a well-defined notion of what constitutes a good code. In the late 20th century, this picture of cryptography radically changed. A rich theory emerged, enabling the rigorous study of cryptography as a science ...". Later in the introduction the authors authors state three basic principles of the modern cryptography. We evoke them here as stated in the textbook:

1. *Principle 1* — the first step in solving any cryptographic problem is the formulation of a rigorous and precise definition of security.

2. *Principle 2* — when the security of a cryptographic construction relies on an unproven assumption, this assumption must be precisely stated. Furthermore, the assumption should be as minimal as possible.

3. *Principle 3* — cryptographic constructions should be accompanied with a rigorous proof of security with respect to a definition formulated according to principle 1, and relative to an assumption stated as in principle 2 (if an assumption is needed at all).

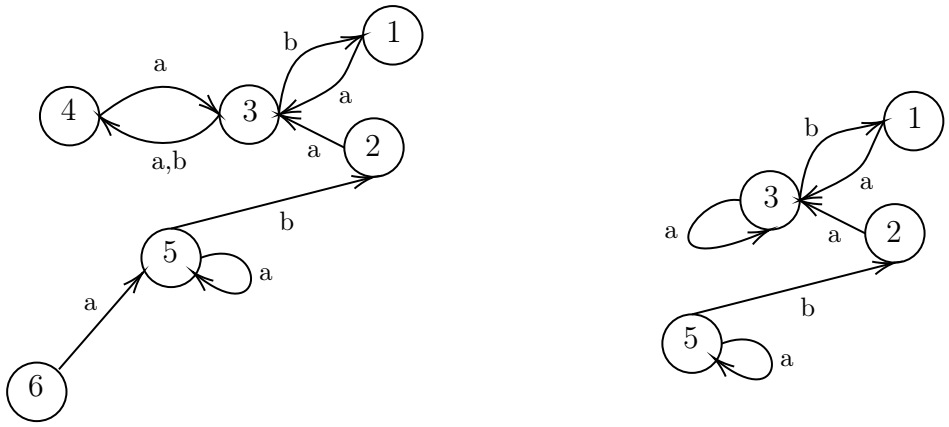
In this section, we study a possibility of designing a public key cryptosystem utilizing the problem of careful synchronization of a PFA. This is, however, not the first attempt to develop the asymmetric cryptosystem with the notion of finite automata (see for example [52]). In Sections 6.1 and 6.2 we describe algorithms of ciphering and deciphering, while Section 6.3 is devoted to the problem of defining the private and public keys for our cryptosystem. Constructions and algorithms in latter sections are rather "proof of concept" than fully applicable solutions and demand further research to adapt them to real-world applications. We still need to delve into the problem of defining the security for our cryptosystem to meet the requirements of Katz and Lindell stated in the previous paragraph and then "fill the gaps" in our method, to fulfill those requirements. As in the previous centuries, to the best of our knowledge, there is "very little theory" that we can rely on while trying to meet the principles of [28], as the modern cryptography utilizes mostly algorithmic number theory rather than languages and automata theory or even graph theory.

6.1 Encryption

Let the plaintext be a word $p = p_1 \dots p_k$ where each $p_i \in \{0, 1\}$. Assume that we have m PFAs: $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1)$, $\dots$, $\mathcal{A}_m = (Q_m, \Sigma, \delta_m)$ such that $\Sigma \cap \{0, 1\} = \varnothing$, that are all carefully synchronized by a common word $w \in \Sigma^*$. First, we describe a construction that is a ciphertext. Encryption consists of five steps:

1. Create a directed graph $G = v_1 \rightarrow_{p_1} v_2 \rightarrow_{p_2} \ldots \rightarrow_{p_k} v_{k+1}$ (the i-th edge is labelled with the i-th letter of the plaintext p)
2. For each vertex v_i in G, choose automaton $\mathcal{B}_i = (Q'_i, \Sigma, \delta'_i)$ from $\{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$
3. For any $\mathcal{B}_i, \mathcal{B}_{i+1}$, choose states $q_1 \in \mathcal{B}_i$ and $q_2 \in \mathcal{B}_{i+1}$ and add a transition $q_1 \rightarrow_{p_i} q_2$
4. For all $\mathcal{B}_i$, choose l pairs of states $p, q \in \mathcal{B}_i$ and add a transition $t \in \{0, 1\}$ between them
5. For all $p \in \mathcal{B}_i$ and for any letter $a \in \Sigma$, choose all states q such that $\delta'_i(q, a)$ is undefined, choose $\mathcal{B}_j$ and state $r \in \mathcal{B}_j$ and define the transition $q \rightarrow_a r$

As a result, we obtain an automaton $\mathcal{C} = (\bigcup_{i=1}^{k} Q'_i, \Sigma \cup \{0, 1\}, \tau)$ - a ciphertext. It is straightforward from the construction, that computing such automaton is polynomial in terms of Q, Σ and the length of the plaintext. Let the automata $\mathcal{A}_1^{ex}$ and $\mathcal{A}_2^{ex}$ depicted in Fig. 20 be the ones used for encryption in our example. They are both carefully synchronized by the word *aba*.



(a) The automaton $\mathcal{A}_1^{ex}$ (b) The automaton $\mathcal{A}_2^{ex}$

Figure 20: The automata that are carefully synchronized by the word *aba*

The procedure of encrypting the word 010 using $\mathcal{A}_1^{ex}$ and $\mathcal{A}_2^{ex}$ is depicted in Figures 21, 22, 23, 24 and 25.

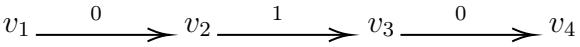


Figure 21: The first step of encryption.

The first step involves creating a directed path labelled with the consecutive letters of the plaintext p.

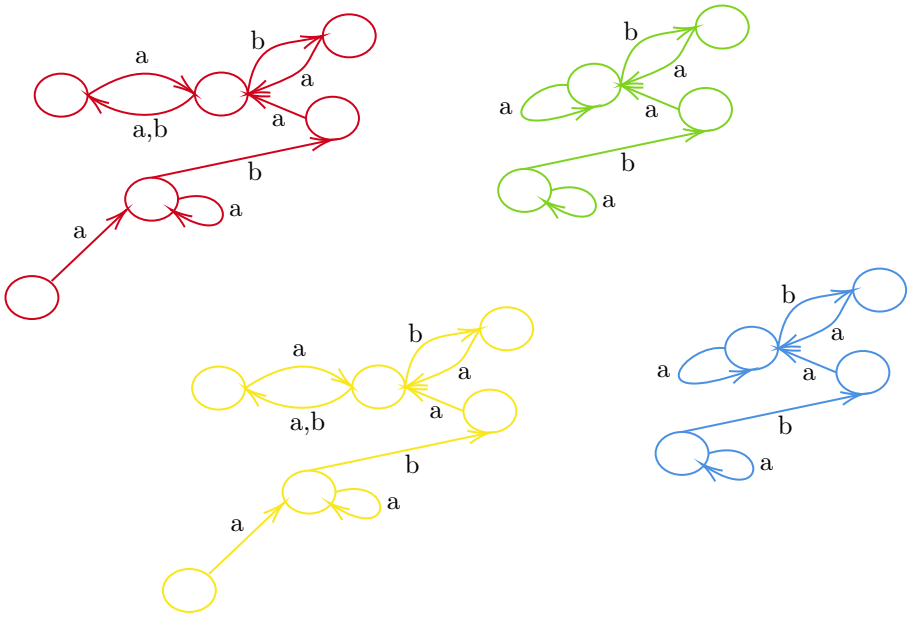


Figure 22: The second step of encryption.

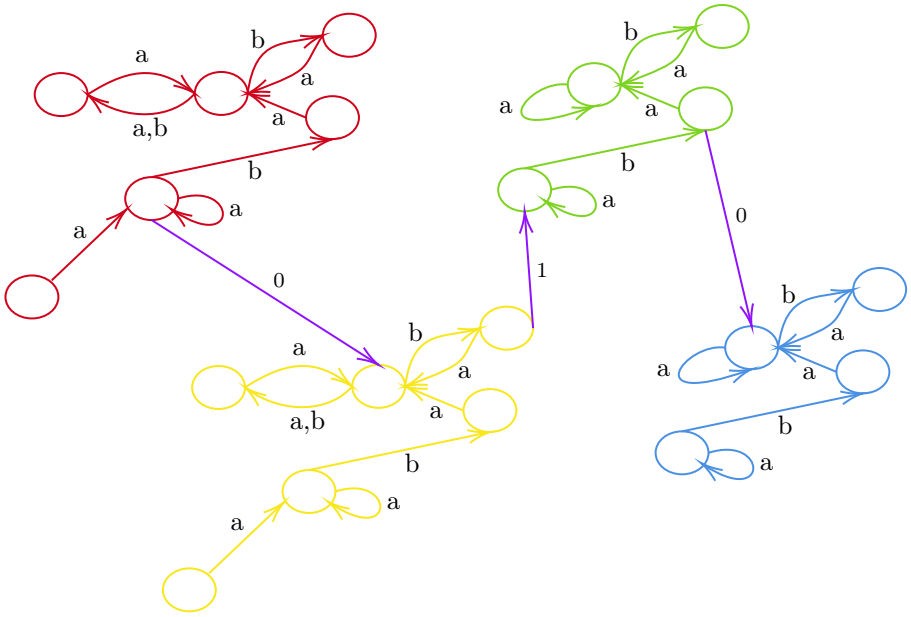


Figure 23: The third step of encryption.

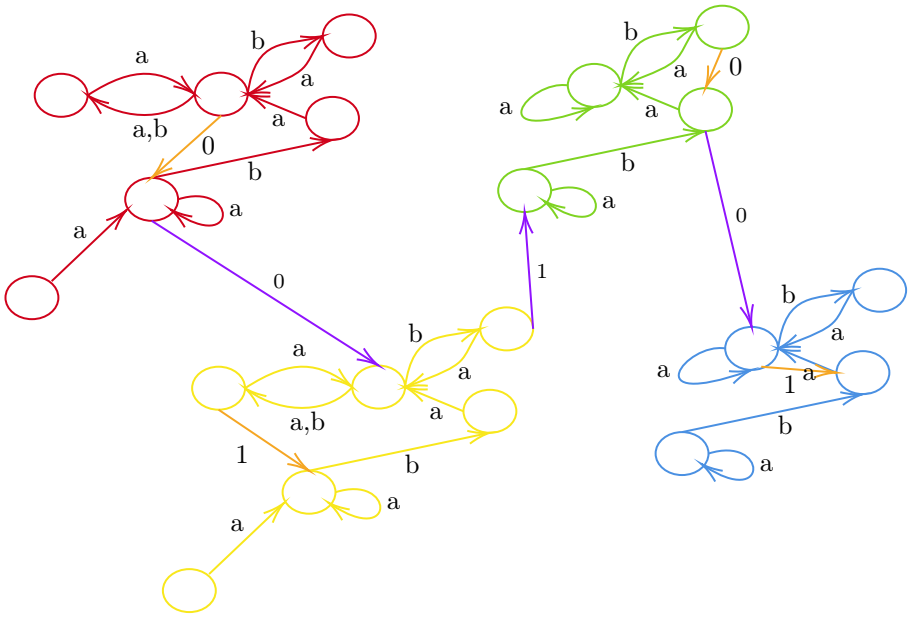


Figure 24: The fourth step of encryption.

In the second step, depicted in Fig. 22, we are replacing vertices of the graph from the first step with randomly chosen automata from the set $\{\mathcal{A}_1^{ex}, \mathcal{A}_2^{ex}\}$. In our

case $v_1 \to \mathcal{A}_1^{ex}$, $v_2 \to \mathcal{A}_1^{ex}$, $v_3 \to \mathcal{A}_2^{ex}$, $v_4 \to \mathcal{A}_2^{ex}$.

In the third step, we are adding transitions $x \in \{0, 1\}$ between automata that correspond to the transitions from the graph from Fig. 21. Since $v_1 \to \mathcal{A}_1^{ex}$ and $v_2 \to \mathcal{A}_1^{ex}$, we are defining the purple 0 transition between the red and yellow automata. The same action is applied to the automata corresponding to vertices v_2, v_3 and v_3, v_4, which results in the automaton depicted in Fig. 23.

The fourth step consists of adding l "obfuscating" $0, 1$ transitions. In our example, these are the orange transitions shown in Fig. 24 with $l = 1$. The last step involves adding "lacking" a, b transitions, so the resulting automaton $\mathcal{C}^{ex}$ restricted to $\{a, b\}$ has one weakly connected component. In our example those are the black transitions.

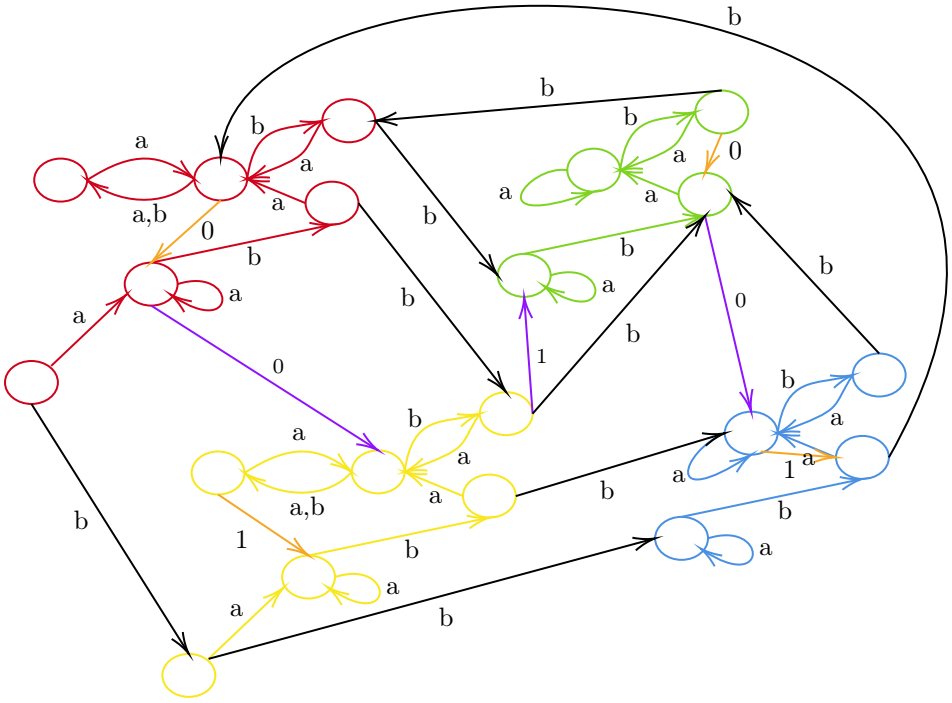


Figure 25: The fifth step of encryption - the ciphertext $\mathcal{C}^{ex}$

6.2 Decryption

In this section, we assume that we have a ciphertext automaton $\mathcal{C} = (P, \Sigma \cup \{0,1\}, \rho)$ constructed by the algorithm defined in Section 6.1 from $|u|+1$ possibly pairwise isomorphic PFAs $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1), \ldots, \mathcal{A}_{|u|+1} = (Q_{|u|+1}, \Sigma, \delta_{|u|+1})$ carefully synchronized by a common word w. Denote $Q_i.w = q_i$ for every $1 \leq i \leq |u|+1$.

Lemma 15. *Let $\mathcal{C}'$ be $\mathcal{C}$ restricted to Σ. We have that $P.w = \{q_1, q_2, .., q_{|u|+1}\}$.*

Proof. It is immediate from the construction defined in Section 6.1, that for any $Q_i \subset P$ we have $Q_i.w = q_i$, since w carefully synchronizes $\mathcal{A}_i$ and each Q_i on Σ induces an isomorphic copy of $\mathcal{A}_i$. On the other hand, $P = Q_1 \cup \ldots \cup Q_{|u|+1}$ and $\mathcal{B}$ is deterministic (because we add only transitions that were previously undefined in Point 5 of the encryption algorithm), so $P.w = (Q_1 \cup \ldots Q_{|u|+1}).w = Q_1.w \cup \ldots \cup Q_{|u|+1}.w = \{q_1, q_2, .., q_{|u|+1}\}$ and the result holds. □

Lemma 16. *There exists an algorithm that, in $O(|P||w|)$ time and $O(|P||w|)$ space, computes a partition of P into subsets Q_1, Q_2, ..., $Q_{|u|+1}$.*

Proof. We describe the desired algorithm. Suppose we have an array with $|P|$ columns and $|w|+1$ rows. Let us put every element of P in a different column of the first row. Next, we fill the i-th row by taking the state from the $(i-1)$-th row of the corresponding column and applying to it the i-th letter of the word w until the end of the row. After this procedure, from Lemma 15, the last row contains only the states from the set $\{q_1, q_2, .., q_{|u|+1}\}$. We can now compute each Q_i by taking these states from the first row that lie in the same columns as the state q_i. □

With these two lemmas we are ready to present a decryption method.

1. Using Lemma 16, compute the partition of P into sets Q_1, Q_2, ..., $Q_{|u|+1}$
2. For every transition $x \in \{0,1\}$ in $\mathcal{B}$, if x joins states from different sets, say Q_i and Q_j, then join q_i and q_j by the transition x, otherwise remove the transition

Observe that after applying this procedure to the ciphertext $\mathcal{B}$ we end up with a graph that was our plaintext. That can be concluded directly from the encryption procedure. It seems like one can decipher the message only by knowing common carefully synchronizing word for automata $\mathcal{A}_1, \ldots, \mathcal{A}_m$ defined in 6.1 or computing every possible induced subautomaton isomorphic to $\mathcal{A}_i$, for $1 \leq i \leq m$.

6.3 Key and security issues

This section is an initial attempt to the cryptanalysis of the presented cryptosystem, assuming the knowledge of the automata used in the encryption algorithm. Our considerations will lead us to the conclusion that taking a tuple of automata with a common carefully synchronizing word as a public key is a rather weak idea either from the practical side (we have no efficient algorithm that generates such a tuple explicitly) and due to security issues (knowing the automata helps attacker to deduce which states of the ciphertext automaton belong to which automaton). Instead, we discuss the possibility of using the reduction construction from Section 5.2 which guarantees, that we obtain carefully synchronizing automaton from the 3-CNF logical formula. We start this section with a simple lemma.

Lemma 17. *If $\mathcal{A}$ is carefully synchronizing, then there exists S that induces a strongly connected component on $\mathcal{A}$ such that, for all of its carefully synchronizing words $w \in \Sigma^*$, we have $Q.w \subseteq S$.*

Proof. Suppose, for the sake of contradiction, that there exist carefully synchronizing words w_1, w_2 such that $Q.w_1 \in S_1$ and $Q.w_2 \in S_2$ and, without the loss of generality, assume that there exists $p \in S_1$ such that, for all $u \in \Sigma^*$, we have $\delta(p, u) \notin S_2$. On the other hand, since w_2 is carefully synchronizing and $Q.w_2 \in S_2$, then $p.w_2 \in S_2$ and we have the contradiction. □

Identifying the set S from Lemma 17 in a carefully synchronizing automaton $\mathcal{A}$ is computationally easy. First, we can use a slight modification of the algorithm that

computes strongly connected components of the directed graph from [53] (Theorem 13), that works in time $O(|\Sigma||Q|)$. Now, it suffices to check which of the components are reachable from all others. This can be computed using a depth-first search algorithm.

Now, we present an algorithm which, for given strongly connected automata $\mathcal{A} = (Q_A, \Sigma, \delta_A)$ and $\mathcal{B} = (Q_B, \Sigma, \delta_B)$, computes in polynomial time (in the size of $\mathcal{A}$ and $\mathcal{B}$) the set $P \subseteq Q_B$ such that the automaton induced by P is isomorphic to $\mathcal{A}$ (the definition can be found in Section 2.1) if such a set exists, otherwise it returns $\varnothing$. In the algorithm below $Q[0]$ stands for an established element of the set of states of $\mathcal{A}$ and M is the dictionary that defines the current candidate for an isomorphism. If $M[p] = q$ then we mean that our candidate isomorphism f satisfies $f(p) = q$.

Algorithm 3 The algorithm for computing an isomorphic strongly connected sub-automaton

Require: $\mathcal{A} = (Q_A, \Sigma, \delta_A)$ - strongly connected, $\mathcal{B} = (Q_B, \Sigma, \delta_B)$

Ensure: $S \subseteq Q_B$ inducing automaton isomorphic to $\mathcal{A}$, or $\varnothing$ if such S does not exist

1: **for each** $q_B \in Q_B$ **do**
2: $p \leftarrow Q_A[0]$
3: $M[p] \leftarrow q_B$
4: S.push(p)
5: **while** not S.empty() **do**
6: $p_1 \leftarrow S$.pop()
7: **for each** $a \in \Sigma$ **do**
8: **if** $\delta_A(p_1, a)$ is defined **then**
9: $p_2 \leftarrow \delta_A(p_1, a)$
10: **if** not M.containsKey(p_2) **then**
11: $M[p_2] \leftarrow \delta_B(M[p_1], a)$
12: S.push(p_2)
13: **else if** $M[p_2] \neq \delta_B(M[p_1], a)$ or $\delta_B(M[p_1], a)$ is not defined **then**
14: $M \leftarrow \varnothing$
15: $S \leftarrow \varnothing$
16: **end if**
17: **end if**
18: **end for**
19: **end while**
20: **if** M.size() $= |Q_A|$ **then return** M.keys()
21: **end if**
22: **end for**
23: **return** $\varnothing$

In Algorithm 3 above S stands for the stack and M for the dictionary. The idea is

to try to "match" every state of automaton $\mathcal{B}$ with every state of automaton $\mathcal{A}$. The dictionary M is used for constructing the desired isomorphism f between $\mathcal{A}$ and some subautomaton of $\mathcal{B}$. The **while** loop in the line 5 is actually a depth-first search of the automaton $\mathcal{A}$. Indeed, a property of being a key of M is equivalent to marking a state as "visited". We interpret **if** clause in the line 10 as defining $f(p_2) = \delta_B(f(p_1), a)$ and we push p_2 on S in order for DFS to keep traversing through $\mathcal{A}$. The **else if** condition in line 13 occurs when DFS comes into a state which was already visited before and the constructed f is not an isomorphism (because for some $p_1, p_2 \in Q_A, a \in \Sigma$, such that $\delta_A(p_1, a) = p_2$, we have that $\delta_B(f(p_1), a) \neq f(p_2)$). In that case, we empty the stack S and the mapping M. Observe that the condition in **while** loop (the line 5) may occur in two cases:

1. Case 1 : we visited all states in $\mathcal{A}$ (it is possible, since we assumed that $\mathcal{A}$ is strongly connected)

2. Case 2 : we emptied S in the line 15 (this indicates that the constructed mapping M is not a desired isomorphism)

Case 1 implies that our isomorphism is valid (because condition in the line 13 was never true), so we return the set of all keys of the dictionary M (line 18). Observe that we can reach the line 23 only if we did not find a proper isomorphism. Time needed for that procedure is bounded by $|Q_B||\Sigma||Q_A|$ (DFS for $\mathcal{A}$ for all states of $\mathcal{B}$). All former considerations lead us to the following result.

Theorem 10. *Algorithm 3 runs in polynomial time and returns $P \subseteq Q_B$ such that $\mathcal{B}_P$ is isomorphic to $\mathcal{A}$, or $\varnothing$ if such P does not exist.*

It remains an open question, if one can drop the assumption of strong connectivity of $\mathcal{A}$ in Algorithm 3 and obtain also a polynomial time algorithm for that problem. Nevertheless, we conclude that Algorithm 3 together with Observation 2 may be enough to break the cryptosytem while knowing the automata used in encryption.

Another difficulty would be to generate such a tuple of automata with a common synchronizing word ensuring that resulting instances are indeed hard, i.e., it is hard to find that common carefully synchronizing word. Taking into account these problems, we propose another solution. Section 5.2 is devoted to the proof that deciding whether a given one-cluster binary automaton is carefully synchronzing is NP-hard. We could rephrase the main theorem (Theorem 9) of that section in the following manner:

Theorem 11. *There exists an algorithm that, for a given 3-CNF formula ϕ, constructs in polynomial time the carefully synchronizing automaton $\mathcal{A}_\phi$ if and only if ϕ has a truth evaluation. Moreover, if e is a truth evaluation, then there exists an algorithm constructing the word w_e which carefully synchronizes $\mathcal{A}_\phi$.*

Observe also the following:

Fact 3. *Let $\{x_1, \ldots, x_n\}$ be variables and $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a 3-CNF formula over those variables. If $e : \{x_1, \ldots, x_n\} \to \{0, 1\}$ is a truth evaluation for the formula ϕ, then e (possibly restricted to some subset of variables) is also a truth evaluation for a formula $C_{i_1} \wedge C_{i_2} \ldots \wedge C_{i_k}$, where each $i_j \in \{1, \ldots, m\}$.*

Having these two observations, we can now modify our encryption procedure in the following way:

1. Public key: the formula ϕ; private key: the truth evaluation e
2. Encryption: randomly generate "subformulas" (utilizing Fact 3) $\phi_1, \ldots, \phi_k$ and use them to construct automata $\mathcal{A}_{\phi_1}, \ldots, \mathcal{A}_{\phi_k}$ using the algorithm from Theorem 11, then encrypt the message as stated in Section 6.1
3. Decryption: construct the word w_e and decrypt as stated in Section 6.2

Of course, we presented here only a high-level idea of the proposed cryptosystem. First of all, using Theorem 11, we can generate only one-cluster binary automata. We cannot use one-cluster PFAs, since then the attacker could find the partition

on sets only by looking for a-clusters in the ciphertext. It is easy to modify the reduction from Section 5.2 to obtain automata which are not one-cluster as a result, but that might not be enough for a cryptosystem to be secure. The crucial issue is to define "robustness" of such algorithm against the attack by computing the partition of ciphertext into automata used in the encryption, and then design the reduction from Section 5.2 to fulfill these requirements.

7 Conclusions

The aims outlined in this thesis in section 1.2 have been addressed, leading to an exploration of various aspects within the realm of automata theory and its applications. Let us review each aim and its corresponding achievements:

1. Investigating the impact of alphabet size on the length of carefully synchronizing words resulted in improvement of Martyugin's results from [33]. Additional results from the Section 3.2 provide insights in a problem of careful synchronizability and subset synchronizability with fixed alphabet size.

2. The introduction of the class of DFAs with coinciding cycles and then proving a quadratic upper bound for the length of the shortest synchronizing word for that class fits into the trend of looking for classes of automata with long shortest synchronizing words that is visible in the literature. The proof of quadratic worst-case lower bound for the length of shortest synchronizing words can result with new test cases for algorithms that are finding shortest synchronizing words.

3. Extending the notion of one-cluster automata to partial finite automata represents a notable theoretical development in the field. Through the examination of extremal properties and complexity issues related to carefully synchronizing automata, this thesis contributes to a deeper understanding of mathematical structures and properties within automata theory.

4. Exploring the intersection of automata theory and cryptography by investigating the possibility of designing an asymmetric cryptosystem based on synchronizability presents a novel approach to cryptographic protocol design. By leveraging synchronizability as a cryptographic primitive, this research offers alternative avenues for enhancing security and efficiency in asymmetric encryption schemes.

In conclusion, each aim set forth in this thesis has been achieved, contributing to advancements in automata theory and its practical applications.

References

[1] D. S. Ananichev and M. V. Volkov. Some results on Černý type problems for transformation semigroups. In *Semigroups and Languages*, pages 23–42.

[2] D. S. Ananichev, M. V. Volkov, and V. V. Gusev. Primitive digraphs with large exponents and slowly synchronizing automata. *J Math Sci*, 192:263–278, 2013.

[3] F. Arnold and B. Steinberg. Synchronizing groups and automata. *Theor. Comput. Sci.*, 359(1-3):101–110, 2006.

[4] M.-P. Béal, M. V. Berlinkov, and D. Perrin. A Quadratic Upper Bound on the Size of a Synchronizing Word in One-Cluster Automata. *Int. J. Found. Comput. Sci.*, 22(2):277–288, 2011.

[5] M. V. Berlinkov, R. Ferens, A. Ryzhikov, and M. Szykula. Synchronizing Strongly Connected Partial DFAs. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[6] M. T. Biskup and W. Plandowski. Shortest synchronizing strings for Huffman codes. *Theor. Comput. Sci.*, 410(38-40):3925–3941, 2009.

[7] G. Chapuy and G. Perarnau. Short synchronizing words for random automata. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 581–604. SIAM, 2023.

[8] M. de Bondt, H. Don, and H. Zantema. DFAs and PFAs with Long Shortest Synchronizing Word Length. In Émilie Charlier, Julien Leroy, and Michel Rigo, editors, *Developments in Language Theory - 21st International Conference, DLT*

2017, Liège, Belgium, August 7-11, 2017, Proceedings, volume 10396 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2017.

[9] M. de Bondt, H. Don, and H. Zantema. Lower Bounds for Synchronizing Word Lengths in Partial Automata. *Int. J. Found. Comput. Sci.*, 30(1):29–60, 2019.

[10] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[11] H. Don and H. Zantema. Finding DFAs with Maximal Shortest Synchronizing Word Length. In Frank Drewes, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 11th International Conference, LATA 2017, Umeå, Sweden, March 6-9, 2017, Proceedings*, volume 10168 of *Lecture Notes in Computer Science*, pages 249–260, 2017.

[12] L. Dubuc. Sur Les Automates Circulaires et la Conjecture de Cerný. *RAIRO Theor. Informatics Appl.*, 32(1-3):21–34, 1998.

[13] J. H. Ellis. The Possibility of Non-Secret Digital Encryption. 1970.

[14] D. Eppstein. Reset Sequences for Monotonic Automata. *SIAM J. Comput.*, 19(3):500–510, 1990.

[15] H. Fernau, V. V. Gusev, S. Hoffmann, M. Holzer, M. V. Volkov, and P. Wolf. Computational Complexity of Synchronization under Regular Constraints. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 63:1–63:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[16] P. Gawrychowski and D. Straszak. Strong Inapproximability of the Shortest Reset Word. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 - 40th International*

Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I, volume 9234 of *Lecture Notes in Computer Science*, pages 243–255. Springer, 2015.

[17] Z. Gazdag, S. Iván, and J. Nagy-György. Improved upper bounds on synchronizing nondeterministic automata. *Inf. Process. Lett.*, 109(17):986–990, 2009.

[18] S. Ginsburg. On the Length of the Smallest Uniform Experiment which Distinguishes the Terminal States of a Machine. *J. ACM*, 5(3):266–280, 1958.

[19] F. Gonze, V. V. Gusev, R. M. Jungers, B. Gerencsér, and M. V. Volkov. On the Interplay Between Černý and Babai's Conjectures. *Int. J. Found. Comput. Sci.*, 30(1):93–114, 2019.

[20] F. Gonze and R. M. Jungers. Hardly Reachable Subsets and Completely Reachable Automata with 1-Deficient Words. *J. Autom. Lang. Comb.*, 24(2-4):321–342, 2019.

[21] V. V. Gusev and E. V. Pribavkina. Reset Thresholds of Automata with Two Cycle Lengths. *Int. J. Found. Comput. Sci.*, 26(7):953–966, 2015.

[22] S. Hoffmann. Completely Reachable Automata, Primitive Groups and the State Complexity of the Set of Synchronizing Words. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications - 15th International Conference, LATA 2021, Milan, Italy, March 1-5, 2021, Proceedings*, volume 12638 of *Lecture Notes in Computer Science*, pages 305–317. Springer, 2021.

[23] S. Hoffmann. Completely Distinguishable Automata and the Set of Synchronizing Words. In Frank Drewes and Mikhail Volkov, editors, *Developments in Language Theory - 27th International Conference, DLT 2023, Umeå, Sweden, June 12-16, 2023, Proceedings*, volume 13911 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2023.

[24] B. Imreh and M. Steinby. Directable Nondeterministic Automata. *Acta Cybern.*, 14(1):105–115, 1999.

[25] M. Ito and K. Shikishima-Tsuji. Some Results on Directable Automata. In Juhani Karhumäki, Hermann A. Maurer, Gheorghe Paun, and Grzegorz Rozenberg, editors, *Theory Is Forever, Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday*, volume 3113 of *Lecture Notes in Computer Science*, pages 125–133. Springer, 2004.

[26] J. Kari. A Counter Example to a Conjecture Concerning Synchronizing Words in Finite Automata. *Bull. EATCS*, 73:146, 2001.

[27] J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theor. Comput. Sci.*, 295:223–232, 2003.

[28] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[29] A. Kisielewicz, J. Kowalski, and M. Szykuła. Experiments with synchronizing automata. In Yo-Sub Han and Kai Salomaa, editors, *Implementation and Application of Automata - 21st International Conference, CIAA 2016, Seoul, South Korea, July 19-22, 2016, Proceedings*, volume 9705 of *Lecture Notes in Computer Science*, pages 176–188. Springer, 2016.

[30] A. Kisielewicz and M. Szykuła. Generating small automata and the černý conjecture. In Stavros Konstantinidis, editor, *Implementation and Application of Automata - 18th International Conference, CIAA 2013, Halifax, NS, Canada, July 16-19, 2013. Proceedings*, volume 7982 of *Lecture Notes in Computer Science*, pages 340–348. Springer, 2013.

[31] E. Landau. Über die Maximalordnung der Permutationen gegebenen Grades. *Arch.Math. Phys.*, 3, 1903.

[32] P. V. Martyugin. A Lower Bound for the Length of the Shortest Carefully Synchronizing Words. *Russian Mathematics*, 54:46–54, 2010.

[33] P. V. Martyugin. Careful Synchronization of Partial Automata with Restricted Alphabets. In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science - Theory and Applications - 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings*, volume 7913 of *Lecture Notes in Computer Science*, pages 76–87. Springer, 2013.

[34] P. V. Martyugin. Computational Complexity of Certain Problems Related to Carefully Synchronizing Words for Partial Automata and Directing Words for Nondeterministic Automata. *Theory Comput. Syst.*, 54(2):293–304, 2014.

[35] E. F. Moore. *Gedanken-Experiments on Sequential Machines*, pages 129–154. Princeton University Press, Princeton, 1956.

[36] B. K. Natarajan. An Algorithmic Approach to the Automated Design of Parts Orienters. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 132–142. IEEE Computer Society, 1986.

[37] J.-E. Pin. Le problème de la synchronisation et la conjecture de Cerný. In A. De Luca, editor, *Non-commutative structures in algebra and geometric combinatorics vol. 109*, Quaderni de la Ricerca Scientifica, pages 37–48. CNR (Consiglio nazionale delle ricerche, Italy), 1981.

[38] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[39] A. Roman. Experiments on synchronzing automata. *Schedae Informaticae*, 19:35–51, 2010.

[40] J. Ruszil. Some Results Concerning Careful Synchronization of Partial Automata and Subset Synchronization of DFA's. In Pascal Caron and Ludovic Mignot, editors, *Implementation and Application of Automata - 26th International Conference, CIAA 2022, Rouen, France, June 28 - July 1, 2022, Proceedings*, volume 13266 of *Lecture Notes in Computer Science*, pages 106–115. Springer, 2022.

[41] J. Ruszil. Synchronizing Automata with Coinciding Cycles. In Frank Drewes and Mikhail Volkov, editors, *Developments in Language Theory - 27th International Conference, DLT 2023, Umeå, Sweden, June 12-16, 2023, Proceedings*, volume 13911 of *Lecture Notes in Computer Science*, pages 208–218. Springer, 2023.

[42] J. Ruszil. Careful Synchronization of One-Cluster Automata. In Joel D. Day and Florin Manea, editors, *Developments in Language Theory - 28th International Conference, DLT 2024, Göttingen, Germany, August 12-16, 2024, Proceedings*, volume 14791 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2024.

[43] I. K. Rystsov. Minimization of the length of synchronizing words for finite automata. *Theoretical questions of design of computer systems*, page 75–82, 1980.

[44] I. K. Rystsov. Reset Words for Commutative and Solvable Automata. *Theor. Comput. Sci.*, 172(1-2):273–279, 1997.

[45] A. Ryzhikov and M. Szykula. Finding short synchronizing words for prefix codes. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 21:1–21:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[46] A. Salomaa. Composition sequences for functions over a finite domain. *Theor. Comput. Sci.*, 292(1):263–281, 2003.

[47] S. Sandberg. Homing and Synchronizing Sequences. In Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner, editors, *Model-Based Testing of Reactive Systems, Advanced Lectures [The volume is the outcome of a research seminar that was held in Schloss Dagstuhl in January 2004]*, volume 3472 of *Lecture Notes in Computer Science*, pages 5–33. Springer, 2004.

[48] Y. Shitov. An Improvement to a Recent Upper Bound for Synchronizing Words of Finite Automata. *J. Autom. Lang. Comb.*, 24(2-4):367–373, 2019.

[49] B. Steinberg. The Černý conjecture for one-cluster automata with prime length cycle. *Theor. Comput. Sci.*, 412(39):5487–5491, 2011.

[50] M. Suomalainen, A. Q. Nilles, and S. M. LaValle. Virtual Reality for Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 11458–11465. IEEE, 2020.

[51] M. Szykula. Improving the Upper Bound on the Length of the Shortest Reset Word. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[52] R. Tao. Finite Automaton Public Key Cryptosystems. In *Finite Automata and Application to Cryptography*, pages 347–393, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[53] R. E. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[54] A. Trahtman. An Efficient Algorithm Finds Noticeable Trends and Examples Concerning the Cerny Conjecture. In Rastislav Kralovic and Pawel Urzyczyn,

editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 789–800. Springer, 2006.

[55] A. Trahtman. The Černý Conjecture for Aperiodic Automata. *Discret. Math. Theor. Comput. Sci.*, 9(2), 2007.

[56] J. Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Mat.-Fyz. Cas. Slovens.Akad. Vied.*, 14:208–216, 1964.

[57] M. V. Volkov. Synchronization of finite automata. *Russian Mathematical Surveys*, 77(5):819–891, 2022.