Russian Math. Surveys **77**:5 819–891 *Uspekhi Mat. Nauk* **77**:5 53–130

DOI: https://doi.org/10.4213/rm10005e

Synchronization of finite automata

M. V. Volkov

Abstract. A survey of the state-of-the-art of the theory of synchronizing automata is given in its part concerned with the case of complete deterministic automata. Algorithmic and complexity-theoretic aspects are considered, the existing results related to Černý's conjecture and methods for their derivation are presented.

Bibliography: 193 titles.

Keywords: finite automaton, synchronizability, algorithm, computational complexity, reset threshold, Černý's conjecture.

Contents

This research was supported by the Russian Foundation for Basic Research under grant no. 19-11-50120 and by the Ministry of Science and Higher Education of the Russian Federation (project no. FEUZ-2020-0016).

AMS 2020 *Mathematics Subject Classification.* Primary 03D15, 20F10; Secondary 20M35, 68Q25, 68Q45, 68Q70, 68R15.

Introduction

Synchronization of finite automata is a branch of discrete mathematics whose original concepts are rather transparent since they explicitly appeal to commonplace intuition. At the same time, this area of research exhibits diverse and often unexpected relationships with many other areas of pure mathematics and computer science, as well as real-world applications. It is particularly attractive since the researcher quickly faces interesting and hard problems, including a great variety of open problems remarkable for their simple formulations. This is perfectly illustrated by Černý's conjecture, which we discuss in § 3.1 below: the concepts involved in its formulation are absolutely elementary, but for more than 50 years researchers have nevertheless been unable to prove or disprove this conjecture.

In the course of the study of synchronizing automata a huge body of material has been accumulated and the amount of related information keeps increasing impressively rapidly. The surveys on synchronizability [98], [166], and [187], which were published more than 10 years ago, have noticeably got out of date. Therefore, it seems reasonable to put together and classify the recent advances in the theory in order to distinguish the trends of its further development. At the same time, the simplicity of the original concepts mentioned above also makes it attractive to give an outline of the elements of the theory in Russian in a way comprehensible to non-specialists. These two intentions have determined the two-layer structure of the text: a detailed presentation of basic concepts and fundamental facts alternates with compactly formulated addenda, including overviews of recent results and the statements of new research problems. It is supposed that newcomers to the topic can leave out these addenda, whereas competent experts can only look through these addenda in search of interesting facts and questions.

In the process of preparation of the paper its length increased rapidly and, finally, I had to divide the text into two papers. In this paper the study is restricted to the classical case of complete deterministic automata. In § 1 we introduce the notion of synchronizability and give an account of where and when it appeared — synchronizing automata were rediscovered several times by mathematicians, computer scientists, and engineers. In § 2 we present algorithms for detecting synchronizability; we also discuss the computational complexity of the main problems related to synchronizability. In § 3 we formulate Černý's conjecture and give a review of the related results. In the concluding section we announce the second paper of the cycle, which is devoted to some other important aspects of synchronizability.

The author attempted to present the key results so as to make them comprehensible for the reader familiar with elements of algorithmics at the level of the textbook [55]. In some parts of the discussion in § 2 we employ notions and results of the theory of computational complexity within the scope of the beginning chapters of the textbook [130]; however, the presentation in § 3 is almost free of these details.

The list of references includes only the titles mentioned in the paper; any attempt to compile a more or less complete bibliography of the results related to synchronizing automata would be in contradiction to Kozma Prutkov's maxim: "One cannot embrace the unembraceable". We offer apologies to the authors whose publications are not mentioned in this survey. The list of references does not comprise preprints and papers in proceedings of conferences that were subsequently covered by publications in journals.

This work is based on the notes of the lecture courses given by this author over the years at the Ural Federal University and abroad. The author is grateful to the listeners of these courses for their interest and valuable comments; special thanks are due to Gregory Javens and Leonid Lagunov. From 2010 on we worked in tandem with Jarkko Kari on the chapter [101] in a reference book on automata theory; I express my deep appreciation to Jarkko both for our collaborative work on that chapter and for allowing me to include some of the materials from that work in this paper.

1. Synchronizing automata: what, where, when?

1.1. Finite automata and their graphs. A *finite automaton* is a simple, but rather resourceful concept, which reflects a quite important idea of interaction between an object and its environment. There are a lot of species of finite automata; here we deal with the simplest version, namely, with deterministic finite automata. A *complete deterministic finite automaton* (DFA) is a triple $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, where Q is the *state set*, Σ is the *input alphabet*, and $\delta\colon Q \times \Sigma \to Q$ is the *transition function*[1]. Elements of Q and Σ are called *states* and (*input*) *symbols*, respectively. It is assumed that the DFA evolves in discrete time. At each instant it has a certain current state $q \in Q$. At the next instant, in response to an input symbol $a \in \Sigma$ the automaton changes from its current state to the state $\delta(q, a)$.

Let us comment on the three adjectives (complete, deterministic, and finite) used in the above definition. The adjective 'finite' is to indicate that both the state set and the input alphabet are always assumed to be finite; it is also assumed that both of these sets are non-empty. The attribute 'deterministic' points out that the state $\delta(q, a)$ is uniquely determined by the pair $(q, a) \in Q \times \Sigma$, so that the transition rule is indeed a function. Finally, 'complete' means that the transition function is assumed to be totally determined: $\delta(q, a)$ is defined for each pair $(q, a) \in Q \times \Sigma$.

Finite automata can be visualized conveniently by means of labelled graphs. We expect the reader to be familiar with the concept of a graph; let us specify more exactly what kind of graph is used here. Throughout this paper a *graph* is a quadruple of sets and maps: a *vertex set* V, an *edge set* E, a map $h\colon E \to V$ taking each edge to its *head*, and a map $t\colon E \to V$ taking each edge to its *tail*. Note

[1] In this form a finite automaton was apparently defined for the first time by Medvedev [123].

that our graphs are allowed to have edges sharing both the tail and head; such edges are said to be *parallel*. Also, loops are allowed (a *loop* is an edge whose tails and heads coincide). Thus, our graphs are in fact directed multigraphs. However, we call them graphs for short, since no other types of graphs appear in this work. A *labelled graph* is equipped with one more map $E \to \Lambda$, where the set Λ is called the *alphabet of labels*; this map assigns a *label* to each edge of the graph. An edge with tail v, head v', and label a is denoted by $v \xrightarrow{a} v'$.

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is represented as a labelled graph with vertex set Q, the label alphabet Σ, and edge set

$$\{q \xrightarrow{a} q' \mid q,\, q' \in Q,\ a \in \Sigma,\ \delta(q,a) = q'\}.$$

Thus, the transition from a state q to a state q' caused by an input symbol a is depicted by the edge with tail q, head q', and label a. For example, Fig. 1 shows a DFA $\mathscr{C}_4$ with four states 0, 1, 2, 3, and two input symbols a and b and transition function

$$\delta(i,a) = \begin{cases} 1, & \text{if } i = 0, \\ i, & \text{if } i = 1,2,3; \end{cases} \qquad \delta(i,b) = i + 1 \pmod 4 \text{ for } i = 0,1,2,3.$$

Here and below we use edges with multiple labels to denote bundles of parallel edges. For instance, the edge $0 \xrightarrow{a,b} 1$ in Fig. 1 is used to denote the parallel edges $0 \xrightarrow{a} 1$ and $0 \xrightarrow{b} 1$.

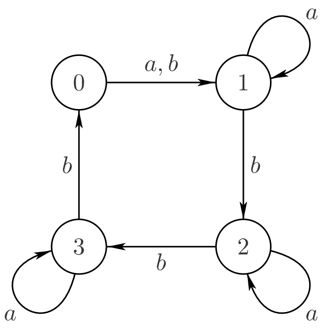


Figure 1. The automaton $\mathscr{C}_4$.

Omitting labels on the graph of a DFA produces a graph called the *support* of this DFA. We use the notions and terms of graph theory in the context of DFAs by suggesting that they be applied to their supports. For example, the automaton $\mathscr{C}_4$ shown in Fig. 1 is strongly connected, as so is its support.

A *word* over the alphabet Σ is a finite (perhaps empty) sequence of letters of Σ. The *empty word* is denoted by the symbol ε. The set of all words over the alphabet Σ, including ε, is denoted by Σ^*. It forms a *monoid*, a semigroup with unity with respect to the operation of word concatenation; the role of the unity in this monoid is played by ε. The set of all non-empty words over Σ is denoted by Σ^+. If $w = a_1 \cdots a_\ell \in \Sigma^+$, then the integer ℓ is called the *length* of the word w and is denoted by $|w|$. It is assumed that $|\varepsilon| = 0$.

The transition function $\delta\colon Q \times \Sigma \to Q$ extends to a function $Q \times \Sigma^* \to Q$ (still denoted by δ) in the following natural way: for every $q \in Q$ we set $\delta(q,\varepsilon) := q$, and if $w = a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$, then

$$\delta(q,w) := \delta(\dots\delta(\delta(q,a_1),a_2),\dots,a_\ell).$$

When we deal with a fixed DFA, we simplify our notation by suppressing the sign of the transition function: we write $\langle Q,\Sigma\rangle$ for $\langle Q,\Sigma,\delta\rangle$ and $q\,.\,w$ for $\delta(q,w)$.

1.2. Synchronizing automata.

Definition 1.1. A complete deterministic finite automaton $\mathscr{A} = \langle Q,\Sigma\rangle$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ and a state $s \in Q$ such that $q\,.\,w = s$ for all $q \in Q$. In this case w is said to be a *reset word*; it *resets the automaton* $\mathscr{A}$, that is, it *leaves the automaton in state* s.

Example 1.1. The automaton $\mathscr{C}_4$ shown in Fig. 1 is synchronizing.

The claim of Example 1.1 is not that trivial: a reset word for $\mathscr{C}_4$ is not so easily guessed by a look at Fig. 1. One of the possible reset words is *abbbabbba*: it resets $\mathscr{C}_4$ by leaving it in state 1. (We will see later on that *abbbabbba* is in fact the shortest reset word for $\mathscr{C}_4$.) This follows from direct computations presented in Table 1.

Table 1. Applying the sequence *abbbabbba* of actions to the states of the automaton $\mathscr{C}_4$

State	a	b	b	b	a	b	b	b	a
0	1	2	3	0	1	2	3	0	1
1	1	2	3	0	1	2	3	0	1
2	2	3	0	1	1	2	3	0	1
3	3	0	1	2	2	3	0	1	1

The computations shown in Table 1 explain the choice of the term ‘synchronization’. Think of the situation as if we had four copies of the automaton $\mathscr{C}_4$ which originally have different current states. If we apply the sequence *abbbabbba* to all four copies simultaneously, then they eventually get ‘synchronized’, that is, go to the same state (and exhibit the same behaviour in response to any other input word applied simultaneously to all of them).

Some terminological observations are appropriate here. In the literature *synchronizing automata* are sometimes called *synchronizable automata* and *reset words* are sometimes called *synchronizing words*. One also comes across other terms like ‘directable’, ‘cofinal’, ‘collapsible’, ‘resettable’, ‘recurrent’, ‘initialisable’ for synchronizing automata and ‘directing’, ‘recurrent’, ‘initializing’ for reset words. On the other hand, the term ‘synchronization’ is often used in other meanings.

Let us mention an obvious but quite useful property of reset words.

Lemma 1.1. *If the word $w \in \Sigma^*$ resets a DFA $\mathscr{A} = \langle Q,\Sigma\rangle$ by leaving it in state $s \in Q$, then for any $u, v \in \Sigma^*$ the word uwv resets $\mathscr{A}$ by leaving it in state $s\,.\,v$.*

In terms of algebra the claim of Lemma 1.1 means that for any DFA $\mathscr{A} = \langle Q,\Sigma\rangle$ the set $\operatorname{Sync}\mathscr{A}$ of its reset words forms an ideal in the monoid Σ^*. (A subset I of a monoid M is called an *ideal* if $m_1 i m_2 \in I$ for all $i \in I$ and $m_1, m_2 \in M$.)

1.3. The story behind the notion of a synchronizing automaton. The concept of a synchronizing automaton presented above took its final shape in the early 1960s. The most often mentioned reference related to synchronizing automata is Černý's paper [44] published in 1964. In fact, at least two researchers came to introduce the same notion slightly earlier. Liu's thesis [117], prepared at the Massachusetts Institute of Technology in 1962, contained a whole chapter devoted to the systematic study of synchronizing automata. Moreover, the term 'synchronizing automaton' seems to have originally been introduced in this thesis: Liu used the term 'synchronizable', whereas Černý called such automata 'directable'. In his e-mail message dated March 10, 2016, Liu, whom (as well as Černý) I consulted in tracing back the notion of synchronization, emphasized the role of his scientific advisor Arden. Synchronizing automata also appeared in the reports [112] and [113] presented by Laemmel at the United States Department of the Army in 1956 and 1963, respectively; Laemmel used the term 'resettable machines'. Laemmel's reports are less thoroughly elaborated in comparison to [117] and [44], but the second of them contains an equivalent of Definition 1.1, the key one (though only for the case of strongly connected automata), and several valuable observations. It may happen that an analysis of some Soviet reports of research projects, when they get declassified, will add the names of Soviet researches[2] to the list of inventors of synchronizing automata.

In Černý's paper [44] the notion of a synchronizing automaton appeared within the classical framework of Moore's 'Gedanken-experiments' [124]. Moore studied finite automata *with outputs*. In automata with outputs each pair (a state, an input symbol) determines both the next state to which the automaton transits, and a symbol of a certain output alphabet that is produced by the automaton in response to the action of the input symbol. Automata with outputs were used as mathematical models of devices operating in discrete modes (such as computers). One of the natural problems that arises in such modelling consists in recovering the unknown current state of the device from the outputs produced by this device in response to various actions. Moore [124] showed that under certain conditions one can uniquely determine the state at which the automaton arrives after a suitable sequence of actions (called an *experiment*). Moore's experiments were adaptive, which means that each next action was selected on the basis of the outputs caused by the previous actions. Ginsburg [77] considered more restricted experiments, which he called *uniform*. A uniform experiment[3] is just a fixed sequence of actions, that is, a word over the input alphabet; thus, in Ginsburg's experiments outputs were only used to calculate the resulting state of the automaton at the end of an experiment. Just one further step from this was required to come to the setting considered by Černý, namely, the setting of the problem of recovering the current state of an automaton without using outputs at all. It should be noted that this setting is by no means artificial — there exist many practical situations in which it is hardly possible or even technically impossible to observe output signals of a control system.

[2]For example, similar ideas can be found in Levenshtein's papers of the same period (see, for instance, [115]).

[3]After the publication of [76] the name of *homing sequence* became standard for this notion.

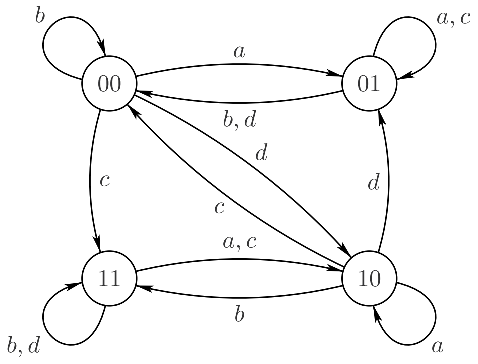


Figure 2. The Ashby automaton.

In Liu's thesis [117] the idea of synchronization was motivated by three problems. The first is the problem of bringing the automaton from an unknown current state to a given state, that is, just the same problem that motivated Černý. The second problem is a version of the first: there are several copies of a fixed automaton which must be synchronized from the originally different current states, The third problem is related to coding theory: Liu shows how synchronizing automata generate codes that can be used to get the sender and the receiver of the encoded message resynchronized after a failure in the message transmission channel. The fact that synchronizing automata are related to codes is indeed of great importance; we discuss this relation in the next section.

Laemmel [113] was similarly motivated: like Černý, he mentioned Ginsburg's experiments (with a reference to [78]) and, like Liu, he associated 'resettable machines' with codes, in particular, with so-called ergodic codes considered by Schützenberger [167]. Moreover, Laemmel drew an analogy between the notion of synchronizability and the ergodic property in physics.

It is no wonder that synchronizing automata were independently and simultaneously (perhaps we should say 'synchronously') invented by several researchers: the notion itself is quite natural and fits well in the topical lines of research in automata theory of the early 1960s. Implicitly, however, the concept of synchronizability appeared in investigations on automata theory since the mid-1950s. For example, we describe an automaton considered in Ashby's classic book [17] (see pp. 60–61). Ashby presented a puzzle dealing with taming two ghostly noises, Singing and Laughter, in a haunted mansion. Each noise can either be on or off, and their behaviour depends on combinations of two possible actions: organ-playing and incense-burning. Under a suitable encoding, this situation is described by the DFA with 4 states and 4 input letters shown in Fig. 2. Here 00 encodes the state when both Singing and Laughter are silent, 01 stands for the state when Singing is off but Laughter is on, and so on. The letter a stands for the transition that happens when neither the organ is played nor incense is burned, b encodes the transition caused by organ-playing in the absence of incense-burning, and so on. The problem is to ensure silence, in other words, to take the automaton in Fig. 2 to state 00. Ashby only solved this problem under the assumption that both noises are on, which means that the automaton is in state 11; his suggested solution is encoded by the word *acb*. However, the reader can easily check that Ashby's automaton is synchronizing[4], and the word *acb* takes it in fact to state 00. Therefore, the additional assumption is inessential, and applying the corresponding sequence of actions makes the house quiet for any initial configuration!

Since the 1960s the notion of a synchronizing automaton was rediscovered repeatedly. One reason for this was that the pioneering works [44], [112], [113], and [117] were hardly available. Černý's paper [44] was published in Slovak in a local journal. Liu's thesis [117] has never been published in full; it is partly contained in [116] but, unfortunately, Lui did not include there his results on synchronization. For this reason his contribution to the theory of synchronizing automata had not been widely recognized and was eventually forgotten for a long period of time[5]. Laemmel's reports [112] and [113], as well as the later report by Laemmel and Rudner [114], were prepared under a contract with the United States Department of the Army, and were not for public distribution (although they were not classified).

A more conceptual reason for the rediscovery of synchronizing automata is that from time to time they 'showed up' unexpectedly in various areas of mathematics, computer science, and engineering, quite far away from the problems that motivated [44], [112], [113], and [117]. We present two examples of this kind in §§ 1.5 and 1.6, after a brief discussion of the role of synchronizing automata in coding theory. In the framework of this theory synchronization of automata has been studied from the mid-1950s (see, for example, the works of Schützenberger [167] and Laemmel [112] already mentioned), that is, before the currently generally accepted definition of a synchronizing automaton was proposed.

1.4. Synchronizing automata and codes. In this section we discuss the relationship between synchronizing automata and codes at the level of general ideas, without assuming any prior acquaintance with coding theory[6].

Speaking about coding we mean the following. Some data are represented by a long word over an alphabet Θ. A good example is a text in a natural language, say, Proust's monumental novel *In search of lost time*, whose French original contains approximately 9 609 000 characters (counting letters, punctuation marks, and space marks). The storage or transmission of such data requires encoding them, that is, replacing each symbol of the alphabet Θ by a word over some other alphabet Σ, usually, the *binary alphabet* $\{0,1\}$. Thus, *coding* consists in applying a map $\chi\colon \Theta \to \Sigma^+$ such that the extension of χ to Θ^+ is injective, which means that each word $w \in \Theta^+$ is uniquely determined by the word in Σ^+ obtained by replacing each letter a in w by the word $a\chi$. The set $\Theta\chi$ is called a *code*, and its elements are *codewords*.

A coding $\chi\colon \Theta \to \{0,1\}^+$ is usually called a *binary coding*. Any binary coding of an alphabet Θ by words of constant length (for instance, ANSII codes) requires $\lceil \log_2 |\Theta| \rceil$ bits for each character. Consequently, encoding a word $w \in \Theta^+$ requires

[4]It is not clear from that book whether or not Ashby realized this nice feature of his automaton.

[5]This, however, did not impede Liu's outstanding career progress as a researcher and a scientific administrator: see the Wikipedia page devoted to him, https://en.wikipedia.org/wiki/Chung_Laung_Liu.

[6]A thorough discussion of these deep and useful connections can be found in the monograph [29] by Berstel, Perrin, and Reutenauer.

$|w| \cdot \lceil \log_2 |\Theta| \rceil$ bits. However, in many cases letters are unequal in a certain sense: some of them occur more frequently, while some others are rare. (For instance, in Proust's novel the letter 'e' occurs twice as often as 'a', and the frequency of the occurrence of 'k' is less than 0.9% of that of 'l'.) Therefore, the memory (in the case of data storage) and/or time (in the case of data transmission) can be saved by withdrawing the requirement that the codewords should have the same length and coding frequently occurring letters of Θ by shorter words over Σ. This simple idea was used as early as the 19th century by Morse in his telegraph code: for example, 'e', the letter most commonly used in English, is assigned the shortest Morse codeword (a single dot).

On the other hand, when codewords have different lengths, the problem of decoding, that is, recovering the word $w \in \Theta^+$ from the sequence of codes encoding its characters, can be non-trivial. Here is a simple example. It is easily verified that the map $\chi\colon \{a, b\} \to \{0, 1\}^+$ defined by $a\chi := 0$ and $b\chi := 01$ is a coding. Assume that a sequence of zeros and ones, starting with a zero, is fed at the input of the decoder. Having read the first bit, the decoder cannot decide yet whether this zero is the code of the character a or the first bit of the code of the character b, and it has to put off the decision until the next bit is read, while the current bit should be stored meanwhile. In the case of a more complicated coding such a delay in decoding can be rather significant, both in time and in the memory required for storing the intermediate data.

However, there is an important class of codes that can be decoded synchronously, which means that the decoding rate is the same as that at which the codewords are fed to the input. A word $x \in \Sigma^*$ is called a *prefix* of a word $y \in \Sigma^+$ if y can be written as $y = xz$ for an appropriate $z \in \Sigma^+$. (Note that in accordance with this definition ε is a prefix of the word y, whereas the word itself is not a prefix of itself.) A set $X \subset \Sigma^+$ is called a *prefix code* if the codewords in X are not prefixes of each other; a coding $\chi\colon \Theta \to \Sigma^+$ is called a *prefix coding* if $\Theta\chi$ is a prefix code.

The procedure for decoding a prefix code is rather simple: the decoder reads the input stream of symbols in Σ from left to right until it recognizes a codeword. This word cannot be a prefix of any other codeword by the definition of a prefix code; therefore, the word that has been recognized is immediately replaced by the corresponding character in Θ and is removed from the stream, after which the process repeats. A classical result of coding theory (the Kraft–McMillan theorem, see [29], Theorem 2.4.12) guarantees that for any code there exists a prefix code over the same alphabet with the same lengths of codewords. This means that the most efficient data compression that can be achieved by employing codewords of different lengths can be implemented with the use of a prefix code. This fact, along with the delay-free character of the decoding, explains why the codes used in applications are usually prefix ones.

The decoding algorithm described above is elegantly implemented by a finite automaton, generally speaking, a partial one[7]. We discuss the corresponding construction in the second paper of our cycle; here we limit our considerations to the special case of finite maximal prefix codes. (A prefix code $X \subset \Sigma^+$ is said to be *maximal* if X is contained in no other prefix code over Σ.) If X is such a code,

[7] *Partial* deterministic automata differ from complete ones only in that their transition functions may not be defined at some pairs (a state, an input symbol).

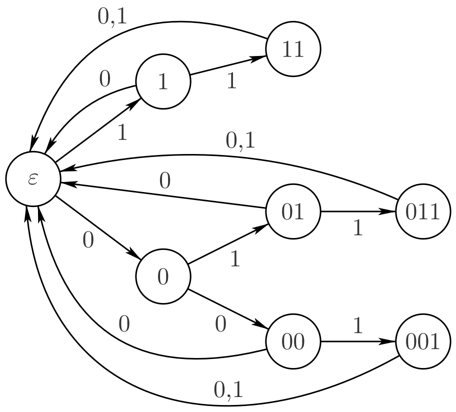


Figure 3. The decoder for the code $\{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$.

then it is decoded by the DFA $\mathscr{A}_X = \langle Q, \Sigma\rangle$, where Q is the set of all prefixes of all words in X and the transition function is defied by

$$q \,.\, a := \begin{cases} qa & \text{if } qa \text{ is a prefix of a word in } X, \\ \varepsilon & \text{if } qa \in X. \end{cases}$$

The automaton $\mathscr{A}_X$ starts its operation at the state ε and partitions the input stream of symbols in Σ into codewords that correspond to getting back to this state. In Fig. 3 we present the DFA $\mathscr{A}_C$ for the maximal prefix code

$$C := \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}. \tag{1}$$

A maximal prefix code X over Σ is said to be *synchronized* if there exists a word $z \in \Sigma^+$ such that for any $y \in \Sigma^*$ the word yz can be written as a product of words in X. Such z is called a *synchronizing word* for X. The advantage of synchronized codes consists in the fact that in case synchronization between the decoder and the coder fails due to transmission errors, it suffices to transmit the synchronizing word, and after that all symbols that follows are decoded correctly. Moreover, since the probability of the event that the word $x \in \Sigma^*$ contains a fixed block z tends to 1 with the increase of the length of x, synchronized codes bring about resynchronization on their own, provided that sufficiently many symbols are transmitted. (As shown in [39], the last property characterizes synchronized codes in fact.)

Let us illustrate this notion with the example of the code C presented by formula (1). It is easily verified that each of the words 010, 011110, 011111110, ... is synchronizing for C. Suppose that in the process of transmission of the codeword 000 an error occurs in the first bit and, as a result, the word 100 is received. The decoder, which is unaware of this error, interprets 10 as a codeword and thus runs out of synchronization, since it starts processing the next bit as a part of another codeword. It can seem that, from this moment on, all decoding will be erroneous, but this is not so. With a high probability the asynchronous mode of operation will not be too long: the decoder will be resynchronized as soon as it reads one of the blocks 010, 011110, 011111110, ... Several examples are presented in Table 2. (The vertical lines in Table 2 are not part of the data transmitted; they only show the partition of each stream into codewords.) The boldface codewords show the blocks starting from which synchronization is reestablished.

Table 2. Reestablishing synchronization in the case of the code C in (1)

Sent	0 0 0 | 0 0 1 0 | **0 1 1 1** | . . .
Received	1 0 | 0 0 0 | 1 0 | **0 1 1 1** | . . .
Sent	0 0 0 | 0 1 1 1 | 1 1 0 | 0 0 1 1 | 0 0 0 | 1 0 | **1 1 0** | . . .
Received	1 0 | 0 0 1 1 | 1 1 1 | 0 0 0 | 1 1 0 | 0 0 1 0 | **1 1 0** | . . .
Sent	0 0 0 | 0 0 0 | 1 1 1 | **1 0** | . . .
Received	1 0 | 0 0 0 | 0 1 1 1 | **1 0** | . . .

The similarity of names suggests that synchronized codes and synchronizing automata are closely related, and this is indeed true. In the special case considered here the relation between these concepts is most intimate, as the following easily verifiable result shows.

Proposition 1.1. *A maximal prefix code is synchronized if and only if its decoder is synchronizing. The synchronizing words for a code X are exactly the ones that reset the DFA $\mathscr{A}_X$ by leaving it in state ε.*

The decoders of synchronized codes form an important subclass of the class of synchronizing automata, and this subclass is still under active investigation (see, for example, the recent publications [28], [30]–[33], [89], and [161]).

1.5. Rediscovery in industrial mechanics. Synchronization problems arise naturally in industrial mechanics in the development of handling devices for various manipulations with components (feeding, sorting, packaging, and so on). Within the framework of this agenda the notion of a synchronizing automaton was independently rediscovered by Natarajan [126], [127] in the mid-1980s, who showed how such automata can be employed in the design of orienters for planar rigid polygonal objects. We explain the idea of this approach using an illustrative example from [10].

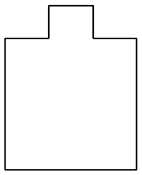


Figure 4. A polygonal component.

Assume that some device has a part of the shape shown in Fig. 4. Such parts arrive at manufacturing sites in boxes, and they need to be properly oriented before assembly. For simplicity, assume that only four initial orientations of a part are possible, namely, the ones shown in Fig. 5.

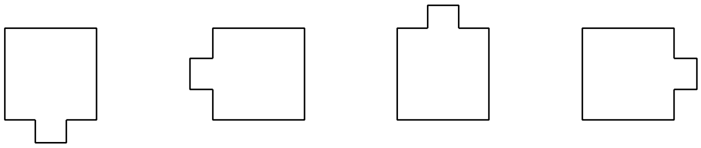


Figure 5. The four possible orientations.

Further, assume that prior the assembly the component must take the 'bump-left' orientation (the second one in Fig. 5). The orienter should put it in the prescribed position regardless of its initial orientation.

There are various approaches to the design of orienters, but practical considerations favour simple and robust mechanic devices with little or no sensing or computer vision. For our particular case these goals can be achieved as follows. We put the parts on a conveyer belt taking them to the assembly point and let the stream of details encounter a series of passive obstacles placed along the belt. We need two types of obstacle, tall and short, A tall obstacle should be high enough in order that any component on the belt encounters this obstacle by its rightmost low angle (we assume that the belt is moving from left to right). Then the component carried by the belt is forced to turn through 90° clockwise, as shown in Fig. 6. A short obstacle makes the same whenever the part is in the 'bump-down' orientation (the first one in Fig. 5); otherwise it does not touch the part, which therefore passes by without changing its orientation.

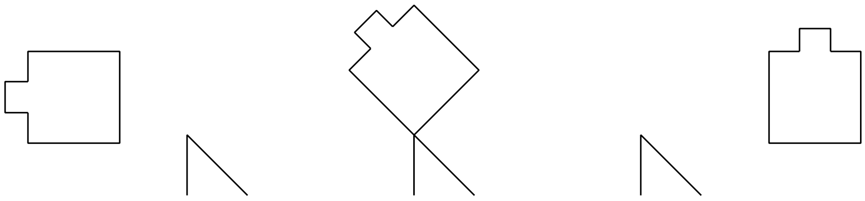


Figure 6. The action of a tall obstacle.

The scheme in Fig. 7 summarizes how the obstacles affect the orientation of the component. We hope that the reader recognizes immediately the automaton $\mathscr{C}_4$ (cf. Fig. 1). Since the word *abbbabbba* resets $\mathscr{C}_4$ leaving it in state 1 (see the discussion of Example 1.1 in § 1.2), the series of obstacles

short–TALL–TALL–TALL–short–
TALL–TALL–TALL–short

yields the required sensor-free orienter.

Since the 1990s the development of feeding and sorting devices on the basis of synchronizing automata has grown into a prolific research direction, but it would be fair to say that publications in this direction deal mostly with implementation technicalities. However, amongst them there are papers of theoretical importance such as [46], [64], [79], and [143].

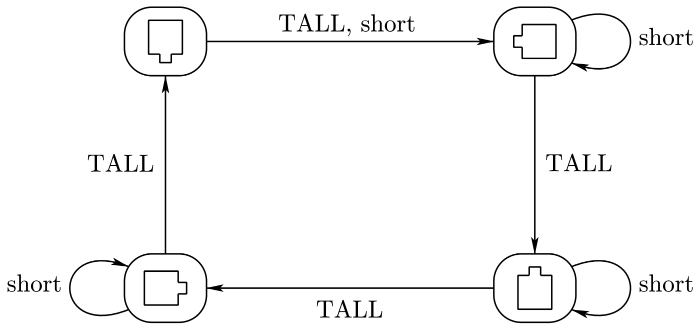


Figure 7. The action of obstacles.

1.6. Rediscovery in the theory of substitution systems. In the 1990s synchronizing automata were rediscovered again in substitution theory. A *substitution* on a finite alphabet X is a map $\sigma\colon X \to X^+$. Substitution theory studies the dynamical aspects of iterations of substitutions on sets of various nature (see the joint monograph [140]). A substitution $\sigma\colon X \to X^+$ is said to be of *constant length* if all words $x\sigma$, $x \in X$, have the same length. Such substitutions are used, for example, to produce fractal objects; by way of illustration, we present an elegant plot of the Peano curve (a continuous curve that passes through every point in the unit square) due to Moore [124].

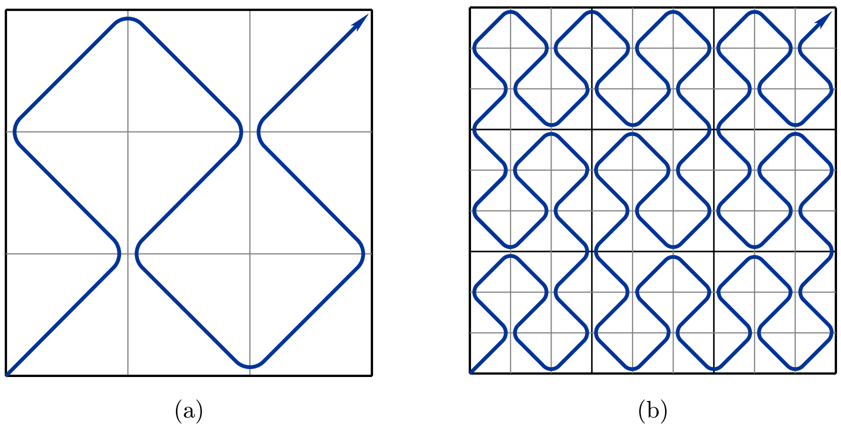


Figure 8. An illustration from Moore's paper [124].

Moore's idea is clear from Fig. 8 ([125], Fig. 3). To indicate the directions more conveniently we rotate the left-hand unit square in Fig. 8 through 45° clockwise, as shown in Fig. 9. We see that the Moore's starting polygonal line consists of 9 links (denoted by arrows in Fig. 9). These links are traversed from the bottom-left angle of the square in the order ENESWSENE, where the letters E, N, W, and S are the initials of the four cardinal directions. At each subsequent iteration each arrow is replaced by a similar 9-link polygonal line, which means applying a substitution of

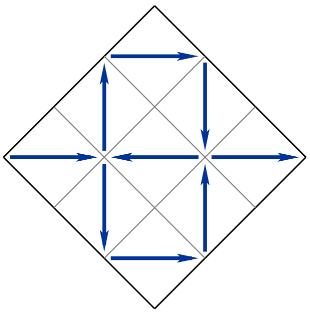


Figure 9. Moore's initial square rotated through 45° clockwise.

constant length 9 by the rule

$$\begin{aligned} \mathrm{E} &\mapsto \mathrm{ENESWSENE},\\ \mathrm{N} &\mapsto \mathrm{NWNESENWN},\\ \mathrm{W} &\mapsto \mathrm{WSWNENWSW},\\ \mathrm{S} &\mapsto \mathrm{SESWNWSES}. \end{aligned}$$

The required Peano curve is the limit curve of the sequence of polygonal lines constructed above.

A substitution $\sigma\colon X \to X^+$ of constant length is said to satisfy the *coincidence condition* if there exist positive integers m and k such that all words $x\sigma^k$, $x \in X$, contains the same letter at the mth position. For example, consider the substitution τ on $X := \{0, 1, 2\}$ defined by

$$0 \mapsto 11, \quad 1 \mapsto 12, \quad 2 \mapsto 20. \tag{2}$$

Calculating the iterations of τ up to τ^4 (see Table. 3) we observe that τ satisfies the coincidence condition for $k = 4$ and $m = 7$.

Table 3. A substitution satisfying the coincidence condition

0	$\mapsto$	11	$\mapsto$	1212	$\mapsto$	12201220	$\mapsto$	122020**1**112202011
1	$\mapsto$	12	$\mapsto$	1220	$\mapsto$	12202011	$\mapsto$	122020**1**120111212
2	$\mapsto$	20	$\mapsto$	2011	$\mapsto$	20111212	$\mapsto$	201112**1**212201220

Dekking [59] showed that the coincidence condition is fundamental for characterizing a number of important properties of dynamical systems determined by constant length substitutions (see [7], Chap. 7, for a survey). For us, however, the coincidence condition is primarily interesting as yet another incarnation of synchronizability. To see this, let us establish a straightforward bijection between the DFAs and the constant length substitutions. Each DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with alphabet $\Sigma = \{a_1, \dots, a_\ell\}$ defines a length ℓ substitution on Q that maps every $q \in Q$ to the word $(q\,.\,a_1)\cdots(q\,.\,a_\ell) \in Q^+$. For instance, the automaton $\mathscr{C}_4$ in Fig. 1 induces the substitution

$$0 \mapsto 11, \quad 1 \mapsto 12, \quad 2 \mapsto 23, \quad 3 \mapsto 30.$$

Conversely, each substitution $\sigma\colon X \to X^+$ such that $|x\sigma| = \ell$ for all $x \in X$ gives rise to a DFA with state set X and ℓ input letters $a_1, \dots, a_\ell$, say, acting on X as follows: $x\,.\,a_i$ is the symbol at the ith position of the word $x\sigma$. For instance, the substitution (2) defines the automaton shown in Fig. 10.

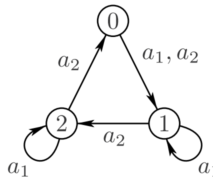


Figure 10. The automaton defined by the substitution (2).

It is easily seen that under the bijection described above substitutions satisfying the coincidence condition correspond precisely to synchronizing automata. Moreover, the number of the iteration at which the coincidence occurs for the first time is equal to the minimum length of reset words for the corresponding DFA. These facts seem to be first noticed by Frettlöh and Sing [71], who used other terminology since they were unfamiliar with the notion of a synchronizing automaton (and they rediscovered some of Černý's results in [44]).

1.7. Algebraic aspects. Given a DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, each word $w \in \Sigma^*$ defines a transformation $[w]\colon Q \to Q$ by the rule $[w]\colon q \mapsto q\,.\,w$. Denote the set of all such transformations by $M(\mathscr{A})$. It is clear that the set $M(\mathscr{A})$ always contains the identity transformation $[\varepsilon]$ and is closed under multiplication of transformations: for any words $u, v \in \Sigma^*$ we have $[u][v] = [uv]$, where uv is the product of u and v in Σ^*, that is, the result of appending v to u. Therefore, $M(\mathscr{A})$ is a submonoid of the monoid $\mathcal{T}(Q)$ of all transformations of the set Q; it is called the *transition monoid of the automaton* $\mathscr{A}$. If a finite automaton is treated as a computational device, then the transition monoid can be thought of as the library of programs of this device which contains descriptions of all available computations.

The *rank of a transformation* of a finite set is the number of elements in the transformation image. Thus, transformations of rank 1 are exactly *constant transformations* which send all elements of the original set to the same element. Now it is clear how the synchronization of a DFA is expressed in terms of its transition monoid: a DFA $\mathscr{A}$ is synchronizing if and only if the monoid $M(\mathscr{A})$ contains a transformation of rank 1.

It is often useful to study the synchronizability phenomenon both from the combinatorial and algebraic points of view. In particular, the algebraic approach admits a natural linearization, which allows the use of tools of linear algebra. Namely, let $Q = \{q_1, \dots, q_n\}$. Assign to each state q_i the vector

$$[q_i] = (\underbrace{0, \dots, 0}_{i-1}, 1, \underbrace{0, \dots, 0}_{n-i})$$

in the space $\mathbb{R}^n$ of n-dimensional row vectors over the field $\mathbb{R}$ of real numbers. Then the vectors $[q_1], \dots, [q_n]$ form a basis of $\mathbb{R}^n$, and for each word $w \in \Sigma^*$ the transformation $[w] \in M(\mathscr{A})$ extends uniquely to a linear operator on $\mathbb{R}^n$. Assigning to each transformation $[w]$ the matrix of this operator in the above basis makes $M(\mathscr{A})$ isomorphic to a monoid of $n \times n$ matrices. The synchronizability of the automaton $\mathscr{A}$ means exactly that this monoid contains a matrix in which one column consists of ones and all other columns consist of zeros. This linearization was efficiently used in the proof of several fundamental results discussed in § 3.4.

The linearization described above is nothing else but a linear representation of the monoid $M(\mathscr{A})$ over the field $\mathbb{R}$, which can also be regarded as a representation over the field of complex numbers. Investigations of synchronizing automata using tools of representation theory have produced a number of informative results. The pioneering studies in this direction were apparently carried out by Rystsov [152]; among other publications in which synchronizing automata were investigated using methods of representation theory we should mention the papers [3]–[5], [16], [147], and [172] by Almeida, Rodaro, Steinberg, and their coauthors.

Transition monoids or, more exactly, their groups of invertible elements, are closely related to one area in the theory of synchronizing automata which became rather popular recently among experts in permutation groups. Denote by $\mathcal{S}(Q)$ the *symmetric group on the set* Q, that is, the group of all permutations of Q. A subgroup $\mathcal{G}$ of $\mathcal{S}(Q)$ is said to be *synchronizing* if for any transformation $\theta \in \mathcal{T}(Q) \setminus \mathcal{S}(Q)$ the submonoid generated by $\mathcal{G}$ and θ contains a transformation of rank 1. In terms of automata theory this means that if the group of invertible elements of the monoid $M(\mathscr{A})$ contains $\mathcal{G}$ as a subgroup, then the DFA $\mathscr{A}$ is synchronizing, provided that the action induced by at least one input symbol of this automaton is not a permutation on the state set. Examples of synchronizing groups are the symmetric group itself, the alternating group, and, more generally, any doubly transitive[8] group of permutations. This follows easily from the synchronizability criterion (see Proposition 2.1 below). On the other hand it is easy to see that a synchronizing subgroup in $\mathcal{S}(Q)$ must be *primitive*, which means that it preserves no partition of the set Q into several blocks containing more than one element each. Thus, the notion of a synchronizing group distinguishes a subclass of the class of primitive groups that contains the class of all doubly transitive groups. The problem of finding an exact characterization of this subclass turns out to be quite profound; it has revealed a number of deep ties with the theory of classical combinatorial configurations (such as Latin squares, Steiner systems, Hadamard matrices, and others) and became the subject of a large number of publications. We do not include a survey of the studies on the (still unfinished) classification of synchronizing groups in this paper, since this material was exhaustively presented in the recent memoir by Araújo, Cameron, and Steinberg [15].

In conclusion of this brief insight into algebra let us note that, in fact, automata can naturally be treated as algebraic structures. Recall that an *algebra of type* $(n_1, n_2, \dots)$ *with underlying set* Q is a non-empty set Q endowed with operations

$$f_1 \colon \underbrace{Q \times Q \times \dots \times Q}_{n_1} \to Q \quad \text{and} \quad f_2 \colon \underbrace{Q \times Q \times \dots \times Q}_{n_2} \to Q, \quad \dots .$$

[8]Recall that a group $\mathcal{G}$ of permutations on a set Q is said to be *doubly transitive* if for any $p, q, r, s \in Q$ such that $p \neq q$ and $r \neq s$ there exists a permutation $\gamma \in \mathcal{G}$ such that $p\gamma = r$ and $q\gamma = s$.

An algebra $(Q; f_1, f_2, \dots)$ of type $(1, 1, \dots)$ is said to be *unary*. It is obvious that the DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is just a unary algebra with underlying set Q and unary operation defined by $[a]\colon q \mapsto q\,.\,a$ for each symbol $a \in \Sigma$. This standpoint allows one to apply the standard algebraic notions of a subalgebra (*subautomaton*), *congruence*, a *quotient automaton*, and so on to automata; we employ these notions below. Note that the class of synchronizing DFAs is closed under taking subautomata and quotients; moreover, each word that resets a DFA also resets all of its subautomata and quotient automata.

When DFAs are treated as unary algebras, the notion of a synchronizing automaton is naturally interpreted in algebraic terms. Recall that a *unary term* is an expression τ of the form $x\,.\,w$, where x is a variable and w is a word over an alphabet Σ. An *identity* is a formal equality between two unary terms. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ treated as a unary algebra *satisfies* an identity $\tau_1 = \tau_2$ if the terms τ_1 and τ_2 evaluate to the same element for each interpretation of their variables in the set Q. Identities in unary algebras can either have the form $x\,.\,u = x\,.\,v$ (*homotypical* identities) or the form $x\,.\,u = y\,.\,v$ for $x \neq y$ (*heterotypical* identities). It is easy to see that a DFA is synchronizing if and only if it satisfies a heterotypical identity. Indeed, if a word w resets a DFA $\mathscr{A}$, then $\mathscr{A}$ satisfies the heterotypical identity $x\,.\,w = y\,.\,w$. Conversely, let $\mathscr{A}$ satisfy a heterotypical identity $x\,.\,u = y\,.\,v$. Substituting in y for x we obtain the identity $y\,.\,u = y\,.\,v$, and by transitivity the identity $x\,.\,u = y\,.\,u$ also holds in $\mathscr{A}$. The last identity means that the word u resets $\mathscr{A}$. Thus, synchronizing automata can be studied in the framework of the equational logic of unary algebras. It is fair to say, however, that so far this approach has not proved to be really useful for understanding the combinatorial nature of synchronizability; however, the number of publications devoted to it is rather considerable (see the survey [34] by Bogdanović, Imreh, Ćirić, and Petković).

1.8. Other applications and relationships. Since the reader has reached this subsection, we hope that they have already been convinced that the notion of the synchronizing automaton is well motivated and arises naturally in a wide range of areas. Here are some further examples of problems which are related to the synchronization of DFAs in one way or another. The fact that these problems are only formulated and not discussed in detail like in the previous subsections by no means evidences that they are less important or less intriguing!

Reset words have long been used in system and protocol testing (see [36], [53], [54], [136], and [145] for typical examples of research in this area; also see the survey [166]). When the system under verification has undergone a test, it should be taken to the original state before doing the next test, and this is the role of reset words. Reset words are used in a similar role in problems of motion planning in robotics, namely, making a robot ‘gone astray’ get back to a prescribed location (see, for example, [175]).

The other, theoretical end of the application spectrum presents relationships between synchronizing automata and algebra and logic. The relationships between synchronizing automata and functions of multivalued logic have actively been studied by Salomaa and his school (see his own papers [162]–[165] and the survey [122]). Some problems in the theory of finite semigroups [6], [137] have motivated interest to *universal reset words*, that is, words that reset all synchronizing automata with a fixed number of states over a fixed alphabet. A survey of results on universal synchronization obtained by the mid-2000s can be found in [9] and [47]; among the later publications in this direction are [48]–[51].

In conclusion of this section we clear up a misunderstanding. It is often mentioned in the literature (and supported by references to [20] and [21]) that synchronizing automata find applications to biocomputing. In these works, undoubtedly very interesting and important, the authors presented the results of successful experiments on the development of automata of nanoscaling size in which short DNA chains were used as states and input symbols alike. For instance, in [20] the authors produce a 'soup of automata', that is, a solution containing $3 \cdot 10^{12}$ identical automata per ml. The idea that such systems of molecular automata can be controlled by 'spicing' the soup with sufficiently many copies of a DNA molecule whose nucleotide sequence encodes a reset word was put forward in [187] as a conjecture. However, it was by no means claimed in [187] that the this idea, which was accounted there as 'yet imaginary', had been implemented in [20], [21], or any other real experiment. (Actually, the molecular automata considered in [20] and [21] are not synchronizing!)

It should be noted that, in fact, there are substantial relations between synchronizing automata and biocomputing (see, for example, [35]); they are discussed in the corresponding section of the second paper of our cycle, which is devoted to the synchronization of partial deterministic automata.

2. Algorithmic and complexity issues

2.1. Checking the synchronizability of an automaton. It should be clear that not all DFAs are synchronizing, and therefore the very first question that arises is how one can determine whether or not a given automaton is synchronizing. This question can be answered with the use of the construction of the subset automaton proposed in the classic work [141] by Rabin and Scott. In monographs on automata theory this construction is usually used to translate non-deterministic finite automata to equivalent deterministic automata (it returns a DFA recognizing the same formal language as the given non-deterministic automaton), but here we apply it to deterministic automata.

Denote the set of non-empty subsets of a set Q by $\mathcal{P}(Q)$. The transition function of the automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ extends to a function $\mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$, which we also denote by δ, by the formula $\delta(P, a) := \{\delta(p, a) \mid p \in P\}$ for any set $P \in \mathcal{P}(Q)$ and any symbol $a \in \Sigma$. We obtain a DFA $\mathcal{P}(\mathscr{A}) = \langle \mathcal{P}(Q), \Sigma, \delta \rangle$, which is called the *subset automaton* of the automaton $\mathscr{A}$. Fig. 11 shows the subset automaton of $\mathscr{C}_4$ (see Fig. 1).

We apply the notational convention from § 1.1 to the subset automaton $\mathcal{P}(\mathscr{A}) = \langle \mathcal{P}(Q), \Sigma, \delta \rangle$ and write $P \,.\, v$ for $\delta(P, v)$ (here P is a non-empty subset of Q and v is a word over Σ). Then it is obvious that a word $w \in \Sigma^*$ resets the DFA $\mathscr{A}$ if and only if the set $Q \,.\, w$ consists of one element. To express the last property in terms of the underlying set of $\mathcal{P}(\mathscr{A})$ recall some notions of graph theory.

In accordance with the definition in § 1.1, a graph is a quadruple $\langle V, E, h, t \rangle$ of sets and maps, where V and E are the vertex set and edge set, respectively, and the maps $h, t \colon E \to V$ take every edge to its tail and head, respectively. Two

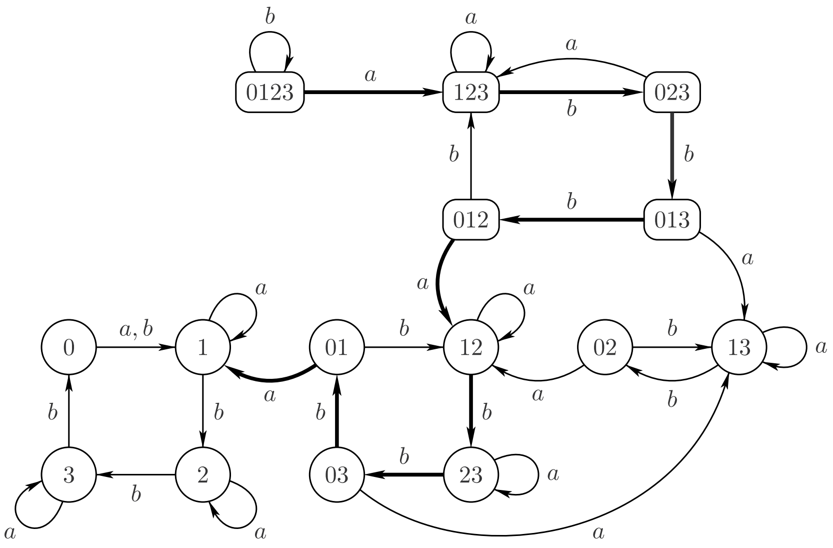


Figure 11. The subset automaton $\mathcal{P}(\mathscr{C}_4)$.

edges $e, e' \in E$ are *successive* if $t(e) = h(e')$. A *path* in a graph is a word over the alphabet E of the edges of this graph any two successive symbols in which correspond to successive edges; the path *length* is the length of this word. In particular, an empty word over E is a path (of length 0) called an *empty path*. A path *starts* at the tail of its first edge and *ends* at the head of its last edge. It is convenient to assume that an empty path can start at any vertex. If a path starts at a vertex v and ends at a vertex v', then we call it a *path from v to v'*. A vertex v' is said to be *reachable* from a v if there exists a path from v to v'.

Now the condition $|Q\,.\,w| = 1$ can be expressed in the following way: the word w is a sequence of labels that are read in the graph of the subset automaton $\mathcal{P}(\mathscr{A})$ along the path starting at Q and ending at a singleton. For example, the labels of edges along the thickened path in Fig. 11 form the reset word for $\mathscr{C}_4$ we have already mentioned. At the same time, it is easily seen that there is no shorter path from $Q = \{0, 1, 2, 3\}$ to a singleton, which means that the automaton $\mathscr{C}_4$ has no shorter reset word, as noted already in the discussion of Example 1.1.

Thus, the question of whether or not a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing reduces to the following problem of reachability in the underlying graph of the subset automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? Clearly, the presence or absence of such a path does not depend on edge labels, and therefore this question relates in fact to the underlying graph of the automaton $\mathcal{P}(\mathscr{A})$. Problems of reachability in graphs are solved using standard methods such as the breadth-first search (see [55], § 22.2). We see that by applying the breadth-first search to the underlying graph of the subset automaton we can solve the problem of synchronizability of the original automaton.

The procedure described above is conceptually very simple but rather inefficient because the state set of the subset automaton is exponentially larger than that

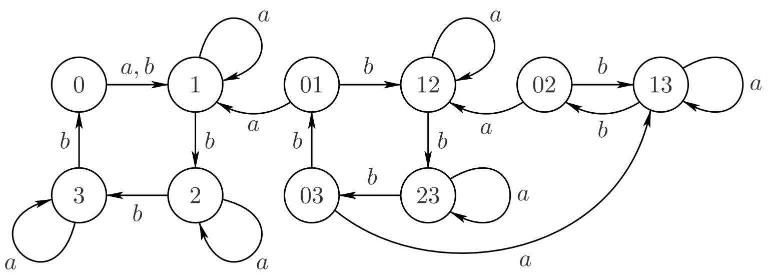


Figure 12. The 2-subset automaton $\mathcal{P}^{\leqslant 2}(\mathscr{C}_4)$.

of the original automaton. However, the following criterion of synchronizability (established independently in the pioneering works by Liu ([117], Theorem 15) and Černý ([44], Theorem 2) and re-established repeatedly afterwards) gives rise to a much more efficient algorithm.

Proposition 2.1. *A DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is synchronizing if and only if for every $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $q \,.\, w = q' \,.\, w$.*

Proof. Of course, only sufficiency requires a proof. For it we consider two arbitrary states $q, q' \in Q$ and a word w_1 such that $q \,.\, w_1 = q' \,.\, w_1$. Then $|Q \,.\, w_1| < |Q|$. If $|Q \,.\, w_1| = 1$, then w_1 is a reset word for $\mathscr{A}$. If $|Q \,.\, w_1| > 1$, then take two states $p, p' \in Q \,.\, w_1$ and consider a word w_2 such that $p \,.\, w_2 = p' \,.\, w_2$. Then $|Q \,.\, w_1w_2| < |Q \,.\, w_1|$. If $|Q \,.\, w_1w_2| = 1$, then the word w_1w_2 resets $\mathscr{A}$; otherwise we repeat the process. Clearly, this procedure produces a reset word for $\mathscr{A}$ in at most $|Q| - 1$ steps. □

If Q is a finite set and $1 \leqslant k \leqslant |Q|$, then for brevity we call k-element subsets of Q *k-subsets*. The set of all k-subsets of Q such that $1 \leqslant k \leqslant m$ for some $m \leqslant |Q|$ is denoted by $\mathcal{P}^{\leqslant m}(Q)$. If $\mathscr{A} = \langle Q, \Sigma\rangle$ is a DFA, then $|P \,.\, a| \leqslant |P|$ for any subset $P \in \mathcal{P}(Q)$ and any symbol $a \in \Sigma$. Therefore, for any $m \leqslant |Q|$ the set $\mathcal{P}^{\leqslant m}(Q)$ is closed under the action of symbols of the input alphabet Σ, and therefore it determines a subautomaton of the subset automaton. This subautomaton is called the *automaton of m-subsets* of the DFA $\mathscr{A}$ and is denoted by $\mathcal{P}^{\leqslant m}(\mathscr{A})$. In Fig. 12 we show the automaton of 2-subsets of the automaton $\mathscr{C}_4$.

Now Proposition 2.1 can be formulated as follows.

Corollary 2.1. *A DFA $\mathscr{A}$ is synchronizing if and only if for each 2-subset D of its state set the underlying graph of the automaton $\mathcal{P}^{\leqslant 2}(\mathscr{A})$ contains a path from D to a 1-subset.*

As above, the question of whether a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is synchronizing reduces to the question of reachability in a certain graph. However, this time we have to deal with the underlying graph of the automaton $\mathcal{P}^{\leqslant 2}(\mathscr{A})$, and this graph has $\dfrac{|Q|(|Q|+1)}{2}$ vertices and $\dfrac{|Q|(|Q|+1)}{2}|\Sigma|$ edges. It is known that the breadth-first search explores a graph with n vertices and m edges in time $\Theta(n+m)$ (see, for example, [55], § 22.2). Therefore, the algorithm based on Corollary 2.1 checks whether a DFA $\mathscr{A}$ is synchronizing in time $O(|Q|^2|\Sigma|)$. We call it the *Liu–Černý algorithm* after its developers.

Based on a deep study of the structure of random automata, Berlinkov suggested an idea for an algorithm that checks the synchronizability of a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ in time $O(|Q|\,|\Sigma|)$ *on the average* (see [25], § 2, and [26], § 4). This algorithm consists of a series of 'true or false' tests. A detailed description of these tests is presented in the second paper of our cycle, in a section devoted to the synchronization of random automata; here we only present the three key properties of this series:

- the testing time for a DFA with n states is $O(n)$;
- the proportion of the DFAs that fail the tests among all DFAs with n states is $O(1/n)$;
- a DFA that has successfully passed all tests is synchronizing.

If a DFA fails to pass a single test, then it can be either synchronizing or not. In this situation the Liu–Černý algorithm should be applied. Therefore, the worst-case complexity of Berlinkov's algorithm is still $O(|Q|^2\,|\Sigma|)$. However, since the proportion of the automata with n states that require the use of the Liu–Černý algorithm is $O(1/n)$, the average running time of Berlinkov's algorithm on a DFA chosen equiprobably among all DFAs with n states and a fixed alphabet is $O(n)$. We emphasize that the algorithm described above is not probabilistic (although it is based on the statistical properties of random automata); it is deterministic and returns a correct answer for any DFA.

Ageev [2] reported a successful implementation of an improved version of Berlinkov's algorithm[9]. Experiments in [2] show that, starting with a DFA with 31 states, the implementation of Berlinkov's algorithm outperforms the implementation of the Liu–Černý algorithm and that the advantage of Berlinkov's algorithm grows with the number of states.

The further discussion in § 2 assumes that the reader is acquainted with some basics of computational complexity[10], in particular, the standard complexity classes **P**, **NP**, **PSPACE**, and so on. In the framework of this theory the problem of checking whether a DFA is synchronizing is formulated in the form of a decision problem:

DFA-SYNC: synchronizability of a complete deterministic automaton.
INSTANCE: a complete deterministic finite automaton $\mathscr{A}$.
ANSWER: YES if $\mathscr{A}$ is synchronizing, NO otherwise.

Corollary 2.1 reduces the DFA-SYNC problem to the known problem of reachability in graphs:

PATH: (directed) reachability in a graph.
INSTANCE: a graph Γ and two of its vertices, s_0 and s_1.
ANSWER: YES if Γ contains a path from s_0 to s_1, NO otherwise.

[9] See https://github.com/birneAgeev/AutomataSynchronizationChecker.
[10] The reader unfamiliar with these basics can skip this part and proceed to § 3.

The PATH problem is one of the 'canonical' complete problems for the complexity class **NL**, which is the class of problems that can be solved on a non-deterministic Turing machine with the use of $O(\log n)$ additional space for an input of length n (see [130], Theorem 16.2). Combining Corollary 2.1 with the non-deterministic algorithm that solves the PATH problem using logarithmic additional space, one can easily show that the DFA-SYNC problem belongs to the class **NL**. In fact, it is **NL**-complete just as the PATH problem is. This result was obtained in 2010 by the Indian computer scientist Sreejith but has not been published. We present the proof here courtesy of the author.

Proposition 2.2. *The problem* DFA-SYNC *is* **NL***-complete.*

Proof. Clearly, it is sufficient to L-reduce some **NL**-complete problem to DFA-SYNC. (L-*reducing* means reducing with the use of logarithmic additional space.) We L-reduce the restriction of PATH to instances of the form (Γ, s_0, s_1), where Γ is the configuration graph of a non-deterministic Turing machine that employs $O(\log n)$ extra space for an input of length n, the vertex s_0 corresponds to the initial configuration of the machine at a given input, and the vertex s_1 is the only accepting configuration at which the machine stops. These very instances of the PATH problem were actually used in the proof of Theorem 16.2 in [130], and therefore the induced subproblem is **NL**-complete. Any non-deterministic Turing machine can be modified by introducing at most logarithmically many new states to a machine with at most two transitions from each configuration. Therefore, without loss of generality we can assume that each vertex in the graph Γ is the tail of at most two edges.

Thus, let $(\Gamma = \langle V, E, h, t\rangle, s_0, s_1)$ be the instance of the subproblem under study. Augmenting the set E with loops at the vertices that are the tails of fewer than two edges, we modify the graph Γ in such a way that all vertices in V have two outgoing edges. (In particular, both edges starting at s_1 can be loops.) It is obvious that this modification does not affect the presence/absence of a path from s_0 to s_1. Now we label the edges of the graph Γ by two letters, a and b, in such a way that for each vertex $v \in V$ one of the edges starting at v is labelled by a and the other by b; in other respects the labels are chosen arbitrarily. The resulting labelled graph corresponds to the DFA $\mathscr{G} := \langle V, \{a, b\}, \zeta\rangle$, where the transition function ζ is defined by the labels: $\zeta(v, c) = v'$ for $c \in \{a, b\}$ and $v, v' \in V$ if and only if the edge $v \to v'$ is labelled by c.

Let $\overline{V} := \{\overline{v} \mid v \in V\}$. Put $Q := V \cup \overline{V}$ and construct a DFA $\mathscr{A}(\Gamma) := \langle Q, \{0, 1\}, \delta\rangle$ with the function δ defined by

$$\delta(v, 0) := \overline{v}, \quad \delta(v, 1) := \zeta(v, a), \quad \delta(\overline{v}, 0) := \overline{s}_1, \quad \delta(\overline{v}, 1) := \zeta(v, b),$$

for $v \in V \setminus \{s_1\}$ and

$$\delta(s_1, 0) = \delta(s_1, 1) := s_1, \quad \delta(\overline{s}_1, 0) := \overline{s}_1, \quad \delta(\overline{s}_1, 1) := s_0.$$

The diagram of the automaton $\mathscr{A}(\Gamma)$ is shown in Fig. 13.

It is clear that constructing the automaton $\mathscr{A}(\Gamma)$ from the initial triple (Γ, s_0, s_1) requires logarithmic-size extra space. Note also that if the coding $\chi\colon \{a, b\} \to \{0, 1\}^+$ is defined by $a\chi := 1$ and $b\chi := 01$, then it is easy to see that $\zeta(v, x) = \delta(v, x\chi)$ for any $v \in V$ and $x \in \{a, b\}^*$.

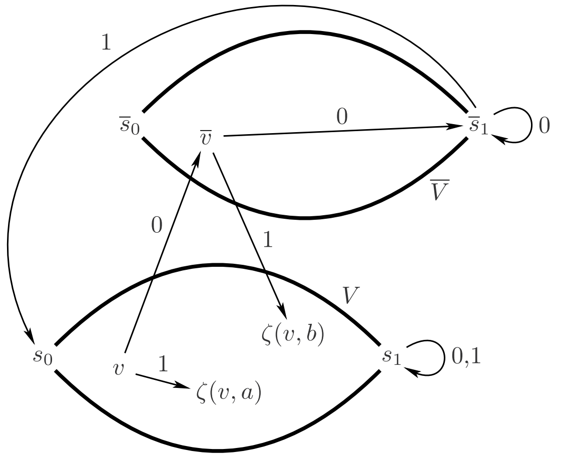


Figure 13. The diagram of the automaton $\mathscr{A}(\Gamma)$; v denotes an arbitrary state in $V \setminus \{s_1\}$.

We claim that the graph Γ contains a path from s_0 to s_1 if and only if the DFA $\mathscr{A}(\Gamma)$ is synchronizing. Indeed, if such a path exists and the labels of its edges in the automaton $\mathscr{G}$ form a word $x \in \{a,b\}^+$, then it can be verified directly that the word $001 \cdot x\chi$ resets $\mathscr{A}(\Gamma)$. Conversely, suppose a DFA $\mathscr{A}(\Gamma)$ is synchronizing. Since $\delta(s_1,0) = \delta(s_1,1) = s_1$, any word in $\{0,1\}^*$ keeps the state s_1 unchanged, and therefore any reset word for $\mathscr{A}(\Gamma)$ leaves it in this state. In particular, there are words $w \in \{0,1\}^+$ (for example, the reset word is one) such that $\delta(s_0,w) = s_1$ Take the shortest word w with such property. The path from s_0 to s_1 along w in the automaton $\mathscr{A}(\Gamma)$ does not visit the state s_0 a second time, therefore, this path does not visit the state $\overline{s}_1$, of which s_0 is the only adjacent successor. This means that w does not contain the block 00, since the repeated action of 0 takes the automaton to $\overline{s}_1$ from any initial state except s_1. It is easily seen that words over $\{0,1\}$ that are free of blocks 00 belong to the image of χ, and therefore $w = x\chi$ for some $x \in \{a,b\}^+$. The path in $\mathscr{G}$ along the word x goes from s_0 to s_1. □

2.2. Synchronizing automata and regular languages. A (*formal*) *language* over a finite alphabet Σ is an arbitrary subset of Σ^*. Finite automata distinguish an important type of languages, namely, *regular languages*; in this case automata play the role of *recognizers*. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ becomes a recognizer once in its state set Q we have distinguished an *initial* state q_0 and a non-empty subset F, elements of which are called *accepting* states. A recognizer $(\mathscr{A}, q_0, F)$ *accepts* a word $w \in \Sigma^*$ if w is a sequence of labels along a path in $\mathscr{A}$ that starts at the state q_0 and ends at one of the states in F, that is, if $q_0 \,.\, w \in F$. The set of all words accepted by $(\mathscr{A}, q_0, F)$ is called the *language recognizable by the automaton* $\mathscr{A}$. A regular language in Σ^* is the language recognizable by some DFA with input alphabet Σ. Kleene's classical theorem [106] characterizes the class of regular languages over Σ as the minimal class of languages that

- contains the empty language and all languages of the form $\{a\}$, where $a \in \Sigma$;

- together with any language L, contains its *Kleene star* L^*, that is, the set of all finite products of words of L (including an empty product, which is assumed to be equivalent to the empty word ε);
- together with any two languages L and K contains their set-theoretic union $L \cup K$ and their *product* LK, that is, the set of all products of words from L and words from K.

Recall that, given a synchronizing DFA $\mathscr{A} = \langle Q, \Sigma\rangle$, we let $\operatorname{Sync}\mathscr{A}$ denote the set of its reset words. The construction of the subset automaton in § 2.1 enables one to establish the following fact easily.

Proposition 2.3. $\operatorname{Sync}\mathscr{A}$ *is a regular language for a synchronizing DFA* $\mathscr{A}$.

Proof. Let $\mathscr{A} = \langle Q, \Sigma\rangle$. A word in Σ^* resets $\mathscr{A}$ if and only if the corresponding labelled path in the subset automaton $\mathcal{P}(\mathscr{A})$ goes from Q to a single-element subset. Therefore, the language $\operatorname{Sync}\mathscr{A}$ is recognized by the DFA $\mathcal{P}(\mathscr{A})$ with the initial state Q and the set of accepting states $\{\{q\} \mid q \in Q\}$. □

Recall that $\operatorname{Sync}\mathscr{A}$ is an ideal of the monoid Σ^* (Lemma 1.1). In view of this Proposition 2.3 means that for any synchronizing DFA $\mathscr{A}$ the language $\operatorname{Sync}\mathscr{A}$ is a non-empty regular ideal. It is easily seen that the converse assertion also holds.

Proposition 2.4. *For any regular ideal I of the monoid Σ^* there exists a synchronizing automaton $\mathscr{A}$ such that $I = \operatorname{Sync}\mathscr{A}$.*

Proof. Let $(\mathscr{A}, q_0, F)$ be the recognizer for I with the smallest number of states. Then any state q in $\mathscr{A}$ is *reachable from* q_0, which means that there exists a word $u \in \Sigma^*$ such that $q_0 \,.\, u = q$. (Otherwise we could obtain a smaller recognizer by deleting the states that cannot be reached from q_0.) Therefore, all transitions in $\mathscr{A}$ from any accepting state $f \in F$ lead to a state in F: if $w \in \Sigma^*$ is a word such that $q_0 \,.\, w = f$, then $w \in I$ and for any symbol $a \in \Sigma$ we have $f \,.\, a = (q_0 \,.\, w) \,.\, a = q_0 \,.\, wa \in F$ since $wa \in I$. This means that the set F must consist of one element, say, s, since otherwise identifying all states in F produces a smaller recognizer. For any $w \in I$ and any state q we have $q \,.\, w = (q_0 \,.\, u) \,.\, w = q_0 \,.\, uw \in F = \{s\}$ since $uw \in I$. Hence any word in I resets the automaton $\mathscr{A}$ by leaving it in state s, which means that $I \subseteq \operatorname{Sync}\mathscr{A}$. The inverse inclusion follows from the fact that s is the unique state to which $\mathscr{A}$ can be brought by a reset word, and therefore any word in $\operatorname{Sync}\mathscr{A}$ should map q_0 to s. □

For $|\Sigma| \geqslant 2$ any regular ideal in Σ^* can be interpreted as a language $\operatorname{Sync}\mathscr{A}$ for an appropriate *strongly connected* synchronizing automaton $\mathscr{A}$ — this (highly non-trivial) refinement was obtained by Reis and Rodaro [144].

In the 2010s the investigation of relationships between synchronizing automata and regular ideals of the monoid Σ^* developed into an actively growing research field in the spirit of so-called descriptional complexity, the theory that measures the complexity of objects in terms of the sizes of their descriptions. Non-empty regular ideals of the monoid Σ^* are infinite objects, and synchronizing automata can be used to describe them in terms of finite sets; moreover, such a description is often much more compact than the 'standard' description using recognizers. For example, it can be verified that any DFA that recognizes the language

Sync $\mathscr{C}_4$, where $\mathscr{C}_4$ is the synchronizing DFA with four states presented in Example 1.1, has at least 12 states. Among the publications in this area we should mention [72], [88], [92], [93], [119]–[121], [139], and [146].

Now we turn back to the computational complexity of synchronizability recognition. It was found out recently (see [66]) that this complexity can vary significantly under most simple constraints. In [66] a family of problems parametrized by regular languages was systematically studied:

DFA-SYNC[L]: synchronizability of a DFA under the restriction to a regular language L.
INSTANCE: a complete deterministic finite automaton $\mathscr{A}$.
ANSWER: YES if $\mathscr{A}$ is synchronizable by a word in L, NO otherwise.

The problem DFA-SYNC considered in § 2.1 belongs to this family: it corresponds to the case where the regular language L is the language of all words over the input alphabet of the automaton, which imposes no constraints on the structure of reset words. However, it often occurs in practice that reset words must have a particular form. For instance, if the device modelled by an automaton can operate in two modes, the regular one and the debug one, then debugging regulations must start and end by a special command which first switches the device to the debug mode and then switches it back to the regular operation mode. (A TeX user recalls of course how TeX switches to the mathematical mode and back to the plain text mode by means of the symbol \$.) Such a restriction corresponds to a regular language of the form $a\Xi^*a$, where a is a special command and Ξ is the set of all other commands of the device. It turns out that even this natural restriction makes the problem of synchronizability computationally hard.

Proposition 2.5. *The problem* DFA-SYNC[$a\Xi^*a$], *where* $a \notin \Xi$, *is* **NP**-*complete for* $|\Xi| = 1$ *and* **PSPACE**-*complete for* $|\Xi| > 1$.

In both cases ($|\Xi| = 1$ and $|\Xi| > 1$) the hardness of the problem DFA-SYNC[$a\Xi^*a$] is a consequence of an easy reduction from the problem FA-INTERSECTION, where, given a finite set of recognizers over the alphabet Ξ, it is required to find out whether there is a word in Ξ^* acceptable to all these recognizers. (This reduction was described in the proof of Proposition 1 in [66].) The problem FA-INTERSECTION is **PSPACE**-complete for $|\Xi| > 1$ (see [110]) and **NP**-complete for $|\Xi| = 1$ (see, for example, [68]). It was shown in [66] that for any regular language L the problem DFA-SYNC[L] belongs to the class **PSPACE**; this yields the **PSPACE**-completeness of the problem DFA-SYNC[$a\Xi^*a$] for $|\Xi| > 1$. In the case where Ξ consists of a single letter, say, b, the fact that the problem DFA-SYNC[$a\Xi^*a$] belongs to the class **NP** is not that obvious, since the synchronizing DFA $\mathscr{A} = \langle Q, \{a, b\}\rangle$ may have no reset words in ab^*a whose length is polynomial in $|Q|$. However, it was verified in [66], Proposition 1, that this problem belongs to **NP**.

The paper [66] presents a lot of examples of regular languages L over two- and three-symbol alphabets for which the problem DFA-SYNC[L] is **PSPACE**-complete. So far, a complete classification of such languages remains an open problem. The complexity of synchronization under various regular constraints was also investigated in [91] and [94]–[97].

2.3. The complexity of computing the reset threshold. The minimum length of reset words for a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is called its *reset threshold* and denoted by $\mathrm{rt}(\mathscr{A})$. This parameter can be computed with the use of the subset automaton, since $\mathrm{rt}(\mathscr{A})$ is the length of the shortest path from Q to a singleton in the underlying graph of the automaton $\mathcal{P}(\mathscr{A})$. Of course, in the worst case the required time is exponential in $|Q|$. However, there have been attempts to implement this approach (see, for example, [102], [145], and [179]). In contrast to the problem of deciding synchronizability, going over to a 'polynomial' subautomaton of 2-subsets does not help here. Moreover, it can be shown under the standard assumptions of the theory of computational complexity that the problem of computing the reset threshold is intractable.

We start our discussion with the following decision problem:

SHORT-SW: synchronization by a word of a given length.
INSTANCE: a DFA $\mathscr{A}$ and a positive integer ℓ.
ANSWER: YES if $\mathrm{rt}(\mathscr{A}) \leqslant \ell$, NO otherwise.

In the formulation of the problem SHORT-SW the form — unary or binary — in which the number ℓ is represented is not specified. This is because the two versions of this problem are equivalent. (This follows from the fact that if a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing, then $\mathrm{rt}(\mathscr{A}) < |Q|^3$; see the detailed discussion in § 3.2 below.)

Rystsov ([150], Theorem 3) established the **NP**-completeness of the problem SHORT-SW. As is so often the case in the theory of synchronizing automata, the pioneering work had gone unnoticed and this important fact was subsequently rediscovered several times (see, for example, [64], [82], and [165]).

Proposition 2.6. *The problem* SHORT-SW *is* **NP**-*complete.*

Proof. To demonstrate that the problem SHORT-SW lies in the class **NP**, first we check whether or not a given DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing. As shown in § 2.1, this can be done in time polynomial in $|Q|$. If $\mathscr{A}$ is not synchronizing, then the answer to the instance $(\mathscr{A}, \ell)$ of the problem SHORT-SW is negative. If $\mathscr{A}$ is synchronizing, then we check the inequality $\ell \geqslant |Q|^3$; if it holds, then the answer to the instance $(\mathscr{A}, \ell)$ is affirmative. Finally, if $\ell < |Q|^3$, then a non-deterministic algorithm guesses the word $w \in \Sigma^*$ of length $\leqslant \ell$ and then checks whether w resets the automaton $\mathscr{A}$. Such a verification can be performed in time $\leqslant \ell|Q|$.

The **NP**-hardness of the problem is demonstrated by a polynomial reduction from the classical SAT problem, which is known to be **NP**-complete (see [130], Theorem 8.2). Recall that an instance of SAT is a system of *clauses* (disjunctions of *literals*, that is, Boolean variables and their negations); it is required to decide

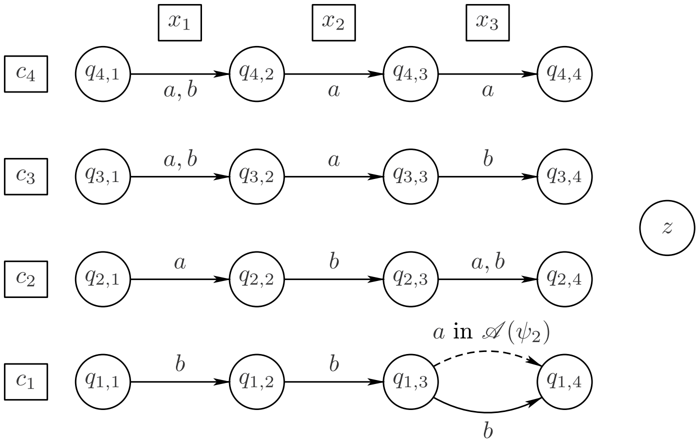

x_1
x_2
x_3
c_4
c_3
c_2
c_1
$q_{4,1}$ $q_{4,2}$ $q_{4,3}$ $q_{4,4}$
$q_{3,1}$ $q_{3,2}$ $q_{3,3}$ $q_{3,4}$
$q_{2,1}$ $q_{2,2}$ $q_{2,3}$ $q_{2,4}$
$q_{1,1}$ $q_{1,2}$ $q_{1,3}$ $q_{1,4}$
z
a, b
a
b
a in $\mathscr{A}(\psi_2)$

Figure 14. The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$.

whether there exists a truth assignment of the variables such that all clauses evaluate to TRUE. For an arbitrary instance ψ of the SAT problem with n variables $x_1, \dots, x_n$ and m clauses $c_1, \dots, c_m$ we construct a DFA $\mathscr{A}(\psi) := \langle Q, \{a, b\}\rangle$ in the following way. The set Q consists of the $(n+1)m$ symbols $q_{i,j}$, where $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n+1$, and the special symbol z. The symbols act in accordance with the following rules:

$$q_{i,j} \,.\, a := \begin{cases} z & \text{if the literal } x_j \text{ occurs in the clause } c_i, \\ q_{i,j+1} & \text{otherwise,} \end{cases} \qquad 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n;$$

$$q_{i,j} \,.\, b := \begin{cases} z & \text{if the literal } \neg x_j \text{ occurs in the clause } c_i, \\ q_{i,j+1} & \text{otherwise,} \end{cases} \qquad 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n;$$

$$q_{i,n+1} \,.\, a = q_{i,n+1} \,.\, b = z \,.\, a = z \,.\, b := z, \qquad 1 \leqslant i \leqslant m.$$

Figure 14 shows two automata of the form $\mathcal{A}(\psi)$ built for two systems of clauses:

$$\psi_1 = \{c_1 := x_1 \vee x_2 \vee x_3,\ c_2 := \neg x_1 \vee x_2,\ c_3 = \neg x_2 \vee x_3,\ c_4 := \neg x_2 \vee \neg x_3\},$$
$$\psi_2 = \{c_1 := x_1 \vee x_2,\ c_2 := \neg x_1 \vee x_2,\ c_3 = \neg x_2 \vee x_3,\ c_4 := \neg x_2 \vee \neg x_3\}.$$

Each ‘row’ in Fig. 14 is labelled by the corresponding clause and each ‘column’ by the corresponding variable. If there is no outgoing edge labelled by $c \in \{a, b\}$ at some state $q \in Q$ in Figure 14, then the edge $q \xrightarrow{c} z$ is implied (such edges are suppressed to improve readability). The two instances ψ_1 and ψ_2 differ only in the first clause: in ψ_1 it contains the literal x_3, whereas in ψ_2 it does not. Correspondingly, the automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$ differ only by the outgoing edge labelled be a at the state $q_{1,3}$: in $\mathscr{A}(\psi_1)$ it leads to z (and is therefore not shown in Fig. 14), while in $\mathscr{A}(\psi_2)$ it leads to the state $q_{1,4}$ and is shown by the dashed line.

Observe that ψ_1 is satisfiable for the truth assignment $x_1 = x_2 := 0$, $x_3 := 1$, while the system of clauses ψ_2 is not satisfiable. It is not hard to check that the word bba resets $\mathcal{A}(\psi_1)$, while $\mathcal{A}(\psi_2)$ is not reset by any word of length 3 (but is reset by every word of length 4 over $\{a,b\}$).

In general, it is easy to see that the DFA $\mathscr{A}(\psi)$ is reset by every word of length $n+1$ over $\{a,b\}$, and it is reset by a word of length n if and only if the system ψ is satisfiable. Thus, assigning the instance $(\mathscr{A}(\psi),n)$ of the problem SHORT-SW to an arbitrary instance ψ of SAT, where n is the number of variables in ψ, one obtains a polynomial reduction of SAT to SHORT-SW. □

Now we introduce the counting problem that corresponds to SHORT-SW:

#SHORT-SW: counting reset words of prescribed length.
INSTANCE: a DFA $\mathscr{A}$ and a positive integer ℓ.
ANSWER: the number of reset words of length ℓ for $\mathscr{A}$.

The reduction used in the proof of Proposition 2.6 establishes a one-to-one correspondence between the satisfying assignments for the system of clauses ψ with n variables and the reset words of length n for the DFA $\mathscr{A}(\psi)$. In particular, the number of satisfying assignments for ψ is equal to the number of reset words of length n for $\mathscr{A}(\psi)$, which means that the reduction constructed above is *parsimonious* in the sense of counting complexity theory. Therefore, it can be regarded as a reduction of the #SAT problem (the problem of counting the satisfying assignments for a fixed system of clauses) to the problem #SHORT-SW. Since the problem #SAT is #**P**-complete (see [130], Theorem 18.1), the problem #SHORT-SW is #**P**-hard. On the other hand, it is easily seen that #SHORT-SW belongs to the class #**P**. As a consequence, we arrive at the following result, which was perhaps explicitly mentioned for the first time by Olschewski and Ummels ([129], Remark 3).

Corollary 2.2. *The problem* #SHORT-SW *is* #**P**-*complete.*

In the literature one can occasionally find speculations like 'the problem of finding the length of the shortest reset word is **NP**-complete' or even 'the problem of finding a shortest reset word is **NP**-complete', in which the authors actually mean the result formulated in Proposition 2.6. Of course, such an interpretation of Proposition 2.6 is not correct; moreover, the construction employed in the proof of this proposition makes it obvious that ascribing these problems to the class **NP** disagrees with the standard assumptions of the theory of computational complexity. To illustrate this, let us introduce the corresponding decision problem:

SHORTEST-SW: checking whether the reset threshold is equal to a given number.
INSTANCE: a DFA $\mathscr{A}$ and a positive integer ℓ.
ANSWER: YES if $\mathrm{rt}(\mathscr{A})=\ell$, NO otherwise.

With a system of clauses ψ in n variables we associate the instance $(\mathscr{A}(\psi),n+1)$ of the problem SHORTEST-SW. As noted in the proof of Proposition 2.6, the DFA $\mathscr{A}(\psi)$ is reset by any word of length $n+1$ over $\{a,b\}$, and it is reset by a word of length n if and only if the system ψ is satisfiable. So $\mathrm{rt}(\mathscr{A}(\psi)) = n+1$ if and only if the system ψ is unsatisfiable. Thus, we reduce to SHORTEST-SW the **negation** of SAT, which is a **coNP**-complete problem. Consequently, the problem SHORTEST-SW is **coNP**-hard. Hence, if SHORTEST-SW belongs to the class **NP**, then the equality **NP** = **coNP** holds, which is considered doubtful. On the hypothesis that **NP** $\neq$ **coNP**, the problem SHORTEST-SW does not belong to **NP**, which means that even a non-deterministic algorithm cannot find the reset threshold of a given DFA in polynomial time.

The complexity class for which SHORTEST-SW is a complete problem was found by Gawrychowski [73] and, independently, Olschewski and Ummels ([129], Theorem 1). It is the class **DP** (Difference Polynomial-Time) introduced by Papadimitriou and Yannakakis [131] (see also [130], § 17.1). It consists of all problems Z that can be reduced polynomially to a pair of problems Z_+ and Z_- in **NP** in such a way that the instances of the problem Z with the answer YES are reduced exactly to pairs of the form (an instance of Z_+ with answer YES, an instance of Z_- with answer NO). **DP** is a wide class, which includes the union **NP** $\cup$ **coNP**, and it is believed that this inclusion is strict. A typical **DP**-complete problem is the problem SAT-UNSAT, where, given a pair of clause systems ψ, φ, one must find out whether it is true that ψ is satisfiable and φ is not. A reduction from SAT-UNSAT was used in [73] and [129] to establish the following fact.

Proposition 2.7. *The problem* SHORTEST-SW *is* **DP***-complete.*

Now we localize the problem of *computing* the reset threshold. It is stated as follows.

COMPUTE-RT: computing the reset threshold.
INSTANCE: a synchronizing DFA $\mathscr{A}$.
ANSWER: the value of $\mathrm{rt}(\mathscr{A})$.

The difference between the problems COMPUTE-RT and SHORTEST-SW is that now it is required to find the reset threshold, rather than to check whether it is equal to a given number.

The functional complexity class $\mathbf{FP}^{\mathbf{NP}[\log]}$ consists of all functions computable by a deterministic polynomial-time Turing machine that has an access to an oracle for an **NP**-complete problem, with the number of queries logarithmic in the size of the input (see [130], § 17.1). It is easily seen that the function $\mathscr{A} \mapsto \mathrm{rt}(\mathscr{A})$ belongs to this class. Indeed, as already mentioned, if a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing, then $\mathrm{rt}(\mathscr{A}) < |Q|^3$. Therefore, one can organize a binary search and find $\mathrm{rt}(\mathscr{A})$ by accessing the oracle $O(\log |Q|)$ times for the **NP**-complete problem SHORT-SW. As shown by Olschewski and Ummels ([129], Theorem 4), this obvious upper bound for the complexity of the problem COMPUTE-RT is sharp.

Proposition 2.8. *The problem* COMPUTE-RT *is* $\mathbf{FP}^{\mathbf{NP}[\log]}$*-complete.*

The proof is based on the reduction from the problem MAX-SAT, whose $\mathbf{FP}^{\mathbf{NP}[\log]}$-completeness was established in [111]. (In the problem MAX-SAT, given a system of clauses, one must find the maximum number of simultaneously satisfiable clauses.)

Finally, we discuss the problem of finding a reset word of minimum length:

COMPUTE-SW: finding a reset word of minimum length.
INSTANCE: a synchronizing DFA $\mathscr{A}$.
ANSWER: reset word of minimum length for $\mathrm{rt}(\mathscr{A})$.

Obviously, finding a reset word of minimum length is not easier than finding its length, that is, the reset threshold. Therefore, it follows from Proposition 2.8 that the problem COMPUTE-SW is $\mathbf{FP}^{\mathbf{NP}[\log]}$-hard. On the other hand it is easily verified that a reset word of minimum length for a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ can be computed in polynomial time (in $|Q|$) on a deterministic Turing machine with access to an oracle for the following **NP**-complete problem (see [129], Theorem 5):

SHORT-SUBSET-SW: reset of a subset by a word of given length.
INSTANCE: a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$, a subset $S \subseteq Q$ and a number ℓ in unary representation.
ANSWER: YES if there exists a word $w \in \Sigma^*$ such that $|S\,.\,w| = 1$ and $|w| \leqslant \ell$,
NO otherwise.

The **NP**-hardness of the problem SHORT-SUBSET-SW follows from Proposition 2.6, since the problem SHORT-SW is a particular case of it (for $S = Q$), and the fact that SHORT-SUBSET-SW belongs to the class **NP** is obvious[11].

The class of all functions that can be computed in polynomial time on a deterministic Turing machine with access to an oracle for an **NP**-complete problem is denoted by $\mathbf{FP}^{\mathbf{NP}}$; this class is important for the theory of computational complexity as the 'residence' of the travelling salesman problem (see [130], Theorem 17.5). The class $\mathbf{FP}^{\mathbf{NP}}$ contains the class $\mathbf{FP}^{\mathbf{NP}[\log]}$, and this inclusion is believed to be strict.

Thus, the complexity of the problem COMPUTE-SW is estimated in terms of the class $\mathbf{FP}^{\mathbf{NP}[\log]}$ from below and by the class $\mathbf{FP}^{\mathbf{NP}}$ from above. It is not yet known whether this problem is complete for any functional complexity class introduced earlier in the literature.

Let us also mention the works [67] and [191], in which the problem SHORT-SW was studied in the framework of the currently popular theory of parametrized complexity. In these works the most natural parameters for instances $(\mathscr{A} = \langle Q, \Sigma\rangle, \ell)$ of the problem SHORT-SW are considered; namely, the size of the state set Q, the size of the alphabet Σ, and the integer ℓ, as well as their combinations.

[11]Note that in the formulation of the problem SHORT-SUBSET-SW the assumption that the integer ℓ is represented in unary is significant! The same problem with ℓ represented in binary is **PSPACE**-complete (see [126] and [151]).

2.4. Complexity of the approximation of the reset threshold. Since the exact value of the reset threshold is hard to compute, it is quite natural to ask whether there exist polynomial-time approximate algorithms. It turns out that under the assumption $\mathbf{P} \neq \mathbf{NP}$, for any approximation of the reset threshold computable in polynomial time there exist series of automata on which this approximation exhibits a large relative error. This fact was established by Gawrychowski and Straszak [74]; it covers all earlier results on the complexity of approximation of the reset threshold (see [24], [25], [75]).

Let us give the necessary definitions. A *polynomial-time algorithm for approximation of the reset threshold* is any algorithm U that for every synchronizing automaton $\mathscr{A}$ returns a positive integer $U(\mathscr{A}) \geqslant \operatorname{rt}(\mathscr{A})$ and takes time polynomial in the number of states. Let $f\colon \mathbb{N} \to \mathbb{R}$ be an arbitrary function. We say that U *approximates the reset threshold to within an accuracy of* $f(n)$ if the following inequality holds for any $n \in \mathbb{N}$ and any synchronizing automaton $\mathscr{A}$ with n states:

$$\frac{U(\mathscr{A})}{\operatorname{rt}(\mathscr{A})} \leqslant f(n).$$

Proposition 2.9 ([74], Theorem 16). *Let $\epsilon > 0$. If $\mathbf{P} \neq \mathbf{NP}$, then no polynomial-time algorithm approximates the reset threshold to within an accuracy of $n^{1-\epsilon}$.*

The proof of Proposition 2.9 involves powerful methods of the modern theory of computational complexity, in particular, Zuckerman's derandomization [193] of the classical result by Håstad [90] stating that the maximum clique problem is hard to approximate.

The result of Proposition 2.9 is sharp: there exist polynomial-time algorithms approximating the reset threshold with accuracy n. For example, Gerbush and Heeringa [75] proposed an algorithm that, given a synchronizing automaton with reset threshold ℓ, finds a reset word of length at most $\lceil (n-1)/(k-1) \rceil \ell$ for it in time $O(kmn^k + n^4/k)$. In other words, such an algorithm gives an $\lceil (n-1)/(k-1) \rceil$-approximation of the reset threshold; in particular, for $k = 2$ we obtain an approximation with accuracy $n-1$.

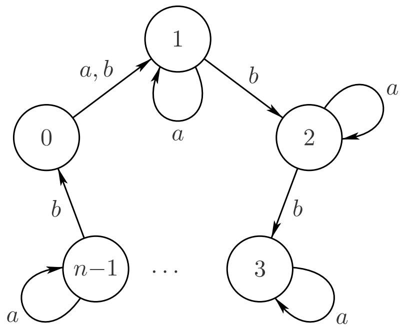
The algorithms from [75] are modifications of the Greedy Compression algorithm, which is discussed in detail in § 3.2 below (see Algorithm 1). Ananichev and Gusev [8] showed that the upper bound $\lceil (n-1)/(k-1) \rceil$ cannot be improved within a very wide family of polynomial-time algorithms that approximate the reset threshold; this family includes the algorithms from [75].

3. Černý's conjecture

3.1. Černý's conjecture: formulation and some history. For the sake of brevity, a DFA with n states is referred to as an *n-automaton*. In 1964 Černý [44] found a series of synchronizing n-automata with reset threshold $(n-1)^2$. Černý's series consists of the automata $\mathscr{C}_n = \langle \{0, 1, \dots, n-1\}, \{a, b\} \rangle$, where the action of the input letters a and b is defined by

$$i \,.\, a := \begin{cases} i & \text{if } i > 0, \\ 1 & \text{if } i = 0; \end{cases} \qquad i \,.\, b = i + 1 \pmod{n}.$$

Our first example of a synchronizing automaton (see Example 1.1 and Fig. 1) is, in fact, the 4-automaton in Černý's series; a generic n-automaton $\mathscr{C}_n$ is shown in Fig. 15 on the left.



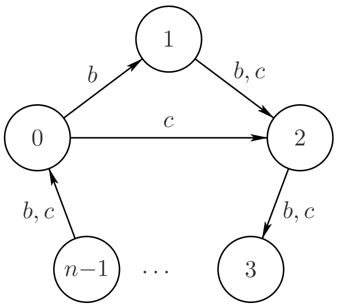


Figure 15. The DFA $\mathscr{C}_n$ and the DFA $\mathscr{W}_n$ induced by the actions of the words b and $c = ab$.

The series $\{\mathscr{C}_n\}_{n=2,3,\dots}$ was rediscovered many times (see, for example, [64], [69], [71], and [114]). It is easily seen that the word $(ab^{n-1})^{n-2}a$ of length $n(n-2)+1 = (n-1)^2$ resets the automaton $\mathscr{C}_n$ by leaving it in state 1.

Proposition 3.1 ([44], Lemma 1). *The reset threshold of the automaton* $\mathscr{C}_n$ *is equal to* $(n-1)^2$.

There are several nice proofs of this result. Here we present the proof from [13]; it is based on a transparent idea and reveals an interesting connection between the Černý automata and an extremal series of graphs discovered in Wielandt's classic paper [192].

Proof of Proposition 3.1. It is sufficient to show that if w is a reset word of minimum length for $\mathscr{C}_n$, then $|w| \geqslant (n-1)^2$. Since the letter b acts on Q as a cyclic permutation, the word w cannot end with b. (Otherwise removing the last letter gives a shorter reset word.) Thus, $w = w'a$, where the prefix w' is such that $Q\,.\,w' = \{0, 1\}$.

Since the letter a fixes each state in its image $\{1, 2, \dots, n-1\}$, every occurrence of a in w, except the last one, is followed by an occurrence of b. (Otherwise the word w contains two consecutive occurrences of a, and removing one of them results in a shorter reset word.) Therefore, if we let $c := ab$, then the word w' can be rewritten as a word v over the alphabet $\{b, c\}$, and the actions of b and c induce a new DFA on the state set Q; we denote this induced DFA (shown in Fig. 15 on the right) by $\mathscr{W}_n$. Since the words w' and v act on Q in the same way, the word vc is a reset word for $\mathscr{W}_n$ and takes the automaton to state 2.

By Lemma 1.1, for any $u \in \{b, c\}^*$ the word uvc is also a reset word for $\mathscr{W}_n$ and it also takes the automaton to state 2. Hence for every $\ell \geqslant |vc|$ there is a path of length ℓ in $\mathscr{W}_n$ from any given state i to 2. In particular, setting $i = 2$ we conclude that for every $\ell \geqslant |vc|$ there is a cycle of length ℓ in $\mathscr{W}_n$. The underlying graph of the automaton $\mathscr{W}_n$ has simple cycles of only two lengths, n and $n-1$. Each cycle of $\mathscr{W}_n$ must consist of simple cycles of these two lengths, so each integer $\ell \geqslant |vc|$

can be expressed as a non-negative integer combination of n and $n-1$. Now we invoke the following well-known elementary result from number theory.

Lemma 3.1 ([142], Theorem 2.1.1). *If positive integers k_1 and k_2 are coprime, then $k_1k_2 - k_1 - k_2$ is the largest integer that is not expressible as a non-negative integer combination of k_1 and k_2.*

Lemma 3.1 yields the inequality $|vc| > n(n-1) - n - (n-1) = n^2 - 3n + 1$. Assume that $|vc| = n^2 - 3n + 2$. Then there must be a path of this length from the state 1 to the state 2. In the DFA $\mathscr{W}_n$ any edge that starts at 1 leads to 2, and thus, in the path it must be followed by a cycle of length $n^2 - 3n + 1$. By Lemma 3.1 no cycles of such length can exist. Hence $|vc| \geqslant n^2 - 3n + 3$.

Since the action of the letter b on any state set $S \subseteq Q$ cannot change the cardinality of S, and the action of c can decrease the cardinality by one at most, the word vc must contain at least $n-1$ occurrences of c. Hence the length of v over $\{b, c\}$ is at least $n^2 - 3n + 2$, and v contains at least $n - 2$ occurrences of c. Since each occurrence of c in v corresponds to an occurrence of the factor ab in w', we conclude that the length of w' over $\{a, b\}$ is at least $n^2 - 3n + 2 + n - 2 = n^2 - 2n$. Thus, $|w| = |w'a| \geqslant n^2 - 2n + 1 = (n-1)^2$. □

We define the *Černý function* $\mathfrak{C}(n)$ as the maximum length of shortest reset words for synchronizing n-automata. Proposition 3.1 establishes the inequality $\mathfrak{C}(n) \geqslant (n-1)^2$. *Černý's conjecture* is the claim that equality actually holds: $\mathfrak{C}(n) = (n-1)^2$.

Černý's Conjecture. *Any synchronizing DFA with n states has a reset word of length $(n-1)^2$.*

In the literature devoted to Černý's conjecture and related issues one often refers to Černý's paper [44] as the source of Černý's conjecture. In fact, that fundamental paper is remarkable in many respects, but the conjecture was *not yet* formulated there. Of course, it is clear that very few people have read the Slovak original of [44], but now an authorized English translation of that work is publicly available. Thus, the interested reader can verify that Černý only made an observation ([44], Theorem 3) that, in our notation, can be expressed as

$$(n-1)^2 \leqslant \mathfrak{C}(n) \leqslant 2^n - n - 1, \tag{3}$$

and he concluded the paper with the following remark:

> "The difference between the bounds increases rapidly and it is necessary to sharpen them. One can expect an improvement mainly for the upper bound".

It appears that the conjecture was first published in 1966 by Starke [170]. Starke improved the upper bound in (3) to $1 + n(n-1)(n-2)/2$, which was the first polynomial upper bound for $\mathfrak{C}(n)$, and concluded the paper [170] with a remark, which we quote here using our notations:

> "Černý's proof of the statement (3) in [44] suggests the conjecture that $\mathfrak{C}(n) = (n-1)^2$ holds for every $n \geqslant 1$. Unfortunately, efforts to either prove or refute the hypothesis have been unsuccessful".

It can be added to this that the situation described in the last phrase has not changed[12] for 56 years!

It is known that Černý gave talks devoted to synchronizing automata (he used the term 'directable' for them) and the conjecture about the length of their reset words at several Czechoslovak conferences held in the late 1960s. Some authors date Černý's conjecture back to his talk at the Bratislava Cybernetics Conference in 1969. In print Černý's conjecture was explicitly stated for the first time in 1971, in his joint paper [45] with Pirická and Rosenauerová. Bearing this in mind it seems that the term 'Černý–Starke conjecture' would be more correct from the historical point of view, but now it is too late to change the terminology which has been in use for more than 40 years[13]. Turning back to the evaluation of Starke's contribution, we also mention his monograph [171], where in § I.9 he discussed results from [44] and [170]. This monograph played an important role in attracting researchers' attention to synchronizing automata.

3.2. Upper estimates. The Pin–Frankl bound. Until recently the best known upper estimate for the Černý function was $\mathfrak{C}(n) \leqslant (n^3-n)/6$. For an arbitrary synchronizing n-state automaton a reset word of length at most $(n^3-n)/6$ can be obtained by the following algorithm.

Algorithm 1. An algorithm for calculating a reset word for the DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ by compression

```
GreedyCompression($\mathscr{A}$)
 1: $w \leftarrow \varepsilon$                                  ▷ initialising the current word
 2: $P \leftarrow Q$                                  ▷ initialising the current set
 3: while $|P| > 1$ do
 4:    if $|P \cdot u| = |P|$ for all $u \in \Sigma^*$ then
 5:       return Failure
 6:    else
 7:       take a word $v \in \Sigma^*$ of minimum length with $|P.v| < |P|$
 8:    $w \leftarrow wv$                               ▷ updating the current word
 9:    $P \leftarrow P.v$                               ▷ updating the current set
10: return $w$
```

Algorithm 1 searches for a reset word according to the 'top-down' greedy strategy: it attempts to compress the current set by a shortest possible word. (We say that a word $u \in \Sigma^*$ *compresses* a subset $S \subseteq Q$ in a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ if $|S\,.\,u| < |S|$.) Let us estimate the running time of the algorithm and the length of the word that it produces.

If $|Q| = n$, then the main loop of Algorithm 1 is executed at most $n-1$ times, since after each execution the size of the current set P decreases at least by 1. The

[12]From time to time preprints by Trahtman appear that contain arguments which the author proposes as a proof of Černý's conjecture: see [184]–[186]. After a while it turns out that the argument contains some errors and the preprints are withdrawn. By the date of publication of this issue of *Uspekhi Matematicheskikh Nauk* (September 2022) the preprint [186] had not been withdrawn, but, according to the common opinion of experts, it unfortunately put forward no novel ideas in comparison to the already withdrawn preprints.

[13]I have tried to find the earliest mention of the term 'Černý's conjecture' in print, but had little success with this. It definitely appeared in the literature in the late 1970s (see, for example, [133]).

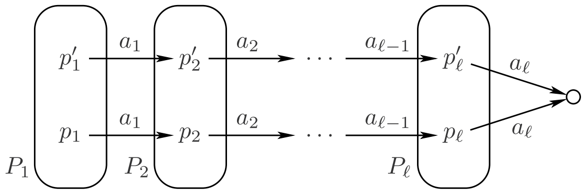


Figure 16. The combinatorial configuration at a generic step of Algorithm 1.

word v in row 7 is the sequence of labels along a shortest path from a 2-element subset of P to a 1-element subset of Q in the automaton $\mathcal{P}^{\leqslant 2}(\mathscr{A})$ (see Corollary 2.1 and the discussion of the Liu–Černý algorithm in § 2.1). The breadth-first search does this in time $O(n^2|\Sigma|)$. Thus, Algorithm 1 is polynomial in the size of $\mathscr{A}$. A detailed analysis of the running time of Algorithm 1 was carried out by Eppstein ([64], Theorem 5); he showed that a precalculation of path lengths in $\mathcal{P}^{\leqslant 2}(\mathscr{A})$ allows Algorithm 1 to be executed in time $O(n^3 + n^2|\Sigma|)$. The paper [64], which also contains a number of other important results, became widely known, and for this reason both Algorithm 1 and its modifications are often referred to as *Eppstein's algorithm*. In [64] Eppstein mentioned Natarajan's work [126] as the source of the algorithm but, in fact, the algorithm had already appeared 20 years earlier, in Starke's note [170], which was not known to either Natarajan or Eppstein.

To estimate the length of the word w returned by Algorithm 1 let us estimate the lengths of the words v appended to w at each execution of the main loop.

Consider a typical step at which $|P| = k > 1$, and let $v = a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$. Then by the choice of v each of the sets

$$P_1 := P, \quad P_2 := P_1.a_1, \quad \dots, \quad P_\ell := P_{\ell-1} \,.\, a_{\ell-1} \tag{4}$$

contains exactly k states. Since $|P_\ell \,.\, a_\ell| = |P \,.\, v| < |P| = |P_\ell|$, there are two distinct states $p_\ell, p'_\ell \in P_\ell$ such that $p_\ell \,.\, a_\ell = p'_\ell \,.\, a_\ell$. Set $R_\ell := \{p_\ell, p'_\ell\}$ and for any $i = 1, \dots, \ell - 1$ consider the 2-element subsets $R_i := \{p_i, p'_i\} \subseteq P_i$ defined by choosing the elements p_i and p'_i in such a way that $p_i.a_i = p_{i+1}$ and $p'_i.a_i = p'_{i+1}$ (see Fig. 16). If $R_i \subseteq P_j$ for some $j < i$, then the word $a_1 \cdots a_j a_i \cdots a_\ell$ of length $j + (\ell - i) < \ell$ satisfies the inequality $|P \,.\, a_1 \cdots a_j a_i \cdots a_\ell| < |P|$, which contradicts the choice of v as a shortest word such that $|P \,.\, v| < |P|$. Therefore, $R_i \nsubseteq P_j$ for any i, j such that $1 \leqslant j < i \leqslant \ell$.

Let $k > 1$. A sequence of k-element subsets $P_1, P_2, \dots$ of an n-element set is said to be 2-*renewing* if each subset P_i contains a 'new' 2-element subset, that is, a 2-element subset R_i such that $R_i \nsubseteq P_j$ for every $j < i$. The argument just presented proves the following assertion.

Lemma 3.2. *Let $\mathscr{A} = \langle Q, \Sigma \rangle$ be a DFA, P be a subset of Q such that $|P| = k > 1$, and let $v = a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$, is a shortest word with $|P.v| < |P|$. Then the sequence of k-element subsets* (4) *is* 2*-renewing.*

By Lemma 3.2, to establish an upper estimate for the length of the word v for $\mathscr{A} = \langle Q, \Sigma \rangle$ with $|Q| = n$, it suffices to find the maximum length of 2-renewing sequences of k-element subsets of an n-element set as a function of n and k. This problem, pertaining to the combinatorics of finite sets, is rather sophisticated. It is easy to give an example of a 2-renewing sequence of k-element subsets of length $\binom{n-k+2}{2}$: one should fix arbitrarily $k-2$ elements of a fixed n-element set and then add all 2-element subsets of the set of remaining $n-k+2$ elements one by one. However, it is by no means obvious whether the sequence constructed in this way has the maximum possible length.

This problem was solved by Frankl [70], who confirmed the optimality of the construction described above[14].

Proposition 3.2. *The maximum length of a* 2*-renewing sequence of k-element subsets of an n-element set is equal to* $\binom{n-k+2}{2}$.

Proof. With each k-element subset $I=\{i_1,\dots,i_k\}$ of the set $\{1,2,\dots,n\}$ we associate a polynomial $D(I)$ in the variables $x_{i_1},\dots,x_{i_k}$ over the field $\mathbb{R}$ by the following formula:

$$I=\{i_1,\dots,i_k\}\mapsto D(I):=\begin{vmatrix}1 & i_1 & i_1^2 & \dots & i_1^{k-3} & x_{i_1} & x_{i_1}^2\\ 1 & i_2 & i_2^2 & \dots & i_2^{k-3} & x_{i_2} & x_{i_2}^2\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots\\ 1 & i_k & i_k^2 & \dots & i_k^{k-3} & x_{i_k} & x_{i_k}^2\end{vmatrix}_{k\times k}.$$

Suppose that k-element subsets of $P_1,\dots,P_\ell$ form a 2-renewing sequence and the polynomials $D(P_1),\dots,D(P_\ell)$ are linearly dependent over $\mathbb{R}$. In this case some polynomial $D(P_j)$ is expressible as a linear combination of the preceding polynomials $D(P_1),\dots,D(P_{j-1})$. By the definition of a 2-renewing sequence, the subset P_j contains elements p and p' such that $\{p,p'\}\nsubseteq P_i$ for all $i<j$. If we substitute $x_p=p$, $x_{p'}=p'$, and $x_t=0$ for $t\neq p,p'$ into each polynomial $D(P_1),\dots,D(P_j)$, then the polynomials $D(P_1),\dots,D(P_{j-1})$ vanish since the last two columns in each of the resulting determinants are proportional. So also does any linear combination of these polynomials. On the other hand, for the indicated values of the variables the value of the polynomial $D(P_j)$ is distinct from 0, since it is the product of a $(k-2)\times(k-2)$ Vandermonde determinant and the determinant $\begin{vmatrix}p & p^2\\ p' & (p')^2\end{vmatrix}=pp'(p'-p)$. Hence $D(P_j)$ cannot be equal to a linear combination of the polynomials $D(P_1),\dots,D(P_{j-1})$, which is a contradiction.

Thus, the polynomials determined by k-element subsets of a 2-renewing sequence are linearly independent. Hence the length of such a sequence does not exceed the dimension of the linear space L, spanned by all polynomials of the form $D(I)$. Let us show that this dimension is at most $\binom{n-k+2}{2}$.

Let $W=\{1,2,\dots,k-2\}$. Consider all k-element subsets $T_1,\dots,T_s$ in $\{1,2,\dots,n\}$ that can be obtained by augmenting W by a 2-element subset of $\{k-1,k,\dots,n\}$; it is clear that $s=\binom{n-k+2}{2}$. We claim that the polynomials $D(T_1),\dots,D(T_s)$ span the space L, which yields the required upper estimate for the dimension of L. We perform induction on the cardinality of the set $I\setminus W$

[14]Actually, Frankl [70] considered and solved a more general problem concerning the maximum length of (analogously defined) m-renewing sequences of k-element subsets of an n-element set for any fixed $m\leqslant k$.

to show that for any k-element subset I of $\{1, 2, \dots, n\}$ the polynomial $D(I)$ is a linear combination of $D(T_1), \dots, D(T_s)$.

If $|I \setminus W| = 2$, then I is one of the subsets T_i. and $D(I) = D(T_i)$.

If $|I \setminus W| > 2$, then there is an element $i_0 \in W \setminus I$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(i_0) = 1$ and $p(i) = 0$ for all $i \in W \setminus \{i_0\}$. Let $I = \{i_1, \dots, i_k\}$ and $I' = I \cup \{i_0\}$. The determinant

$$\begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \dots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \dots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \dots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \dots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}$$

is equal to 0, since the first column is the sum of the next $k-2$ columns with coefficients $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{k-3}$. Thus, expanding this determinant along the first column yields the equality

$$\sum_{j=0}^{k} (-1)^j p(i_j) D(I' \setminus \{i_j\}) = 0.$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, this equality can be written as

$$D(I) = \sum_{j=1}^{k} (-1)^{j+1} p(i_j) D(I' \setminus \{i_j\}), \tag{5}$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$ the non-trivial terms on the right-hand side of (5) correspond to the polynomials $D(I' \setminus \{i_j\})$ such that $i_j \notin W$. The set $I' \setminus \{i_j\}$ is obtained from I by replacing the element $i_j \notin W$ by $i_0 \in W \setminus I$, hence $I' \setminus \{i_j\}$ shares more common elements with W than I. Therefore,

$$|(I' \setminus \{i_j\}) \setminus W| < |I \setminus W|,$$

and by the inductive hypothesis any polynomial $D(I' \setminus \{i_j\})$, where $i_j \notin W$, is a linear combination of $D(T_1), \dots, D(T_s)$. By (5) this also holds for the polynomial $D(I)$. □

Combining Lemma 3.2 with Proposition 3.2 gives the following corollary.

Corollary 3.1. *Let $\mathscr{A} = \langle Q, \Sigma \rangle$ be a DFA, P be a subset of Q with $|P| = k > 1$, and $v \in \Sigma^*$ be a shortest word with the property $|P\,.\,v| < |P|$. Then $|v| \leqslant \binom{n-k+2}{2}$.*

Thus, if ℓ_k is the length of the word that Algorithm 1 appends to the current word after the iteration step the algorithm enters when the current set contains k states, then Proposition 3.1 guarantees that $\ell_k \leqslant \binom{n-k+2}{2}$. Taking the identity $\binom{m}{2} + \binom{m}{3} = \binom{m+1}{3}$ into account, one calculates easily:

$$\sum_{k=2}^{n} \binom{n-k+2}{2} = \binom{n+1}{3} = \frac{n^3 - n}{6} \tag{6}$$

(see, for example, [83], formula (5.10)). Hence, summing all inequalities $\ell_k \leqslant \binom{n-k+2}{2}$ from $k=2$ to $k=n$, one arrives at the bound

$$\mathfrak{C}(n) \leqslant \frac{n^3-n}{6} \tag{7}$$

mentioned above.

The inequality (7) is usually called the Pin–Frankl bound in the literature, making references to [135] by Pin and [70] by Frankl. Pin proved Lemma 3.2 and conjectured the estimate $\binom{n-k+2}{2}$ for the maximum length of 2-renewing sequences in his talk at the Colloquium on Graph Theory and Combinatorics held in Marseille in 1981; the paper [135] is based on that talk. Frankl learned this conjecture from Pin – and immediately proved it – during another colloquium on combinatorics held in Bielefeld in November 1981. The full story is, however, more complicated. Actually, the bound (7) appeared for the first time in the paper [69] by Fischler and Tannenbaum, submitted to the 11th Annual Symposium on Switching and Automata Theory held in Santa Monica, California, in October 1970. In [69] this estimate was deduced from a combinatorial conjecture equivalent to Pin's one; Fischler and Tannenbaum presented several particular cases in which they managed to prove the conjecture and express a hope that a full proof would be available for presentation at the symposium. However, it seems that their hope has never come true: I have found no trace of the proof in the subsequent publications. After that (7) was rediscovered by Kohavi and Winograd [108], [109], but the argument justifying it in their papers was insufficient. In 1987 both Lemma 3.2 and Proposition 3.2 were independently rediscovered by Klyachko, Rystsov, and Spivak [107], who were aware of [69], [108], and [109], but not of the papers by Pin and Frankl. The proof of Proposition 3.2 presented here is taken from[15] [107]; the proof in [70] is based on the same idea.

The word produced by Algorithm 1 is not always a shortest reset word for the DFA under consideration. This should come as no surprise since, as already mentioned in § 2.3, no polynomial-time algorithm, nor even a non-deterministic one, can calculate the reset threshold for an arbitrary synchronizing DFA[16]. A particular example is the DFA $\mathscr{C}_4$. Being executed on this automaton, Algorithm 1 returns the word ab^2abab^3a of length 10, which is composed of the labels along the thickened path in Fig. 17. As we know, this word is not a shortest reset word for $\mathscr{C}_4$. This reveals one of the main intrinsic difficulties of the synchronization problem: the standard optimality principle does not apply here, since it is not true that the optimal solution behaves optimally at all intermediate steps as well. For $\mathscr{C}_4$ the optimal solution is the word ab^3ab^3a, but it cannot be found using Algorithm 1, because at the second execution of the main loop the algorithm chooses a shortest word compressing the 3-element subset $\{1,2,3\}$, that is, the word b^2a, whereas the correct choice is the longer word b^3a (cf. Fig. 11). The algorithm suffers the consequences of such a 'haste' at the next step already: it arrives at the 2-element subset $\{1,3\}$ which is not compressible by any word of length less than 6.

[15]In [107], as well as in [70], the general case of m-renewing sequences of k-element subsets for any $m \leqslant k$ was considered.

[16]Algorithm 1 is not really deterministic. In the general case there can be several words satisfying the conditions in line 7 of the algorithm, and to obtain a deterministic version of Algorithm 1 one should specify a method to choose one of these words.

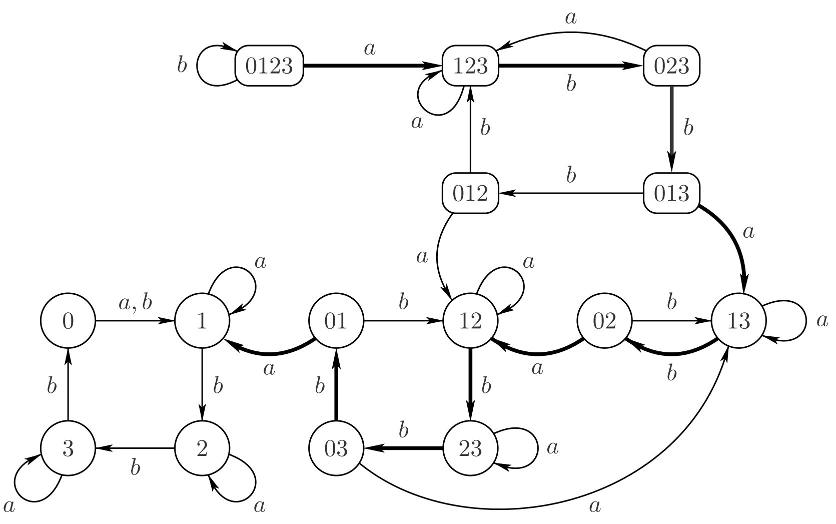


Figure 17. The output of Algorithm 1 executed on the DFA $\mathscr{C}_4$.

Various approaches have been taken in the literature to improve Algorithm 1, for example, it was allowed to 'look ahead' and, in place of a shortest word compressing the k-element subset, choose a word optimizing some other reasonable parameter at each step (see, for example, [75] and the discussion in [148], § 5). Such approaches are of definite practical interest but, of course, they cannot overcome the theoretical barrier established by the results presented in § 2.3.

The gap between the reset threshold of a DFA and the length of the reset word returned by Algorithm 1 on this DFA can be arbitrarily large. For example, one can calculate that for the Černý n-automaton $\mathscr{C}_n$, whose reset threshold is $(n-1)^2$ (see Proposition 3.1), Algorithm 1 produces a reset word of length $\Omega(n^2 \log n)$. Algorithm 1 belongs to the family of algorithms whose approximation properties have been studied thoroughly by Ananichev and Gusev [8]. The reader can find in [8] examples of series of n-automata on which Algorithm 1 exhibits a relative accuracy of $\Omega(n)$. The behaviour of Algorithm 1 on the average has never been considered from the theoretical standpoint; however, it performs quite well in practice.

Concluding the discussion of the Pin–Frankl bound (7) we note that for $n = 1, 2, 3$ it provides the same values as the lower bound $(n-1)^2$, and therefore these three values (0, 1, and 4, respectively) are sharp. Pin [135] verified that for any synchronizing DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ with four states or more there exists a word $u \in \Sigma^*$ of length 9 whose action reduces Q by al least three states: $|Q\,.\,u| \leqslant |Q|-3$; another proof of this fact was given in [57]. If Algorithm 1 starts with $w := u$ and $P := Q\,.\,u$, then the sum of the last three terms $\binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 6+3+1$ in the expression $\sum_{k=2}^{n} \binom{n-k+2}{2}$ from (6) is replaced by 9, which reduces the sum by 1. Therefore, for $n \geqslant 4$ the bound (7) can be reduced to

$$\mathfrak{C}(n) \leqslant \frac{n^3-n}{6} - 1. \tag{8}$$

The bound (8) is sharp for $n = 4$.

3.3. Upper estimates. Recent advances. For 35 years the Pin–Frankl bound remained the best upper estimate for the reset threshold of automata with a fixed number of states. In 2011 Trahtman [183] published a better upper bound, namely, $\mathfrak{C}(n) \leqslant n(7n^2+6n-16)/48$. Unfortunately, that proof contained an error. However, a novel idea was put forward in [183], which was used by Szykuła [177] to derive the estimate

$$\mathfrak{C}(n) \leqslant \frac{85059n^3 + 90024n^2 + 196504n - 10648}{511104}. \tag{9}$$

Szykuła's estimate improves the leading coefficient $1/6 = 0.166666\ldots$ in the Pin–Frankl bound to $\dfrac{85059}{511104} = \dfrac{28353}{170368} = 0.166422\ldots$. The smallest integer n for which the expression on the right-hand side of (9) is less than $\dfrac{n^3-n}{6}$ is equal to 724. Although this improvement is barely perceptible in practice, against the background of the long-lasting 'standstill', Szykuła's result is perceived as a significant advance.

At the end of the paper [177] the author conjectured that the bound (9) can be improved using more accurate calculations. This conjecture was soon confirmed by Shitov [168], who derived the estimate

$$\mathfrak{C}(n) \leqslant \left(\frac{7}{48} + \frac{15625}{798768}\right)n^3 + o(n^3) \tag{10}$$

with leading coefficient close to 0.1654.

A detailed discussion of the idea from [183] and its implementation in [168] and [177] requires some definitions and preliminaries.

Given a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$, we say that a word $w \in \Sigma^*$ *avoids* a state $q \in Q$ if $q \notin Q\,.\,w$. A state that admits an avoiding word is *avoidable*. Note that if a DFA $\mathscr{A}$ is synchronizing and a word v resets it by leaving it in state s, then v avoids all states except s. Therefore, in synchronizing automata all states except, perhaps, one are avoidable. The only possible state in a synchronizing DFA that cannot be avoided is a state z such that $z\,.\,a = z$ for each $a \in \Sigma$. Such a state is called a *sink* (*zero*) state and a DFA with a sink state is an *automaton with zero*. It is clear that a synchronizing DFA can have at most one sink state. Moreover, in considerations related to Černý's conjecture we can limit ourselves to automata without sink states. This follows from two observations: first, the problem of establishing an upper bound for the reset threshold of n-automata reduces to two particular cases: the case of strongly connected automata and the case of automata with zero; second, the case of automata with zero turns out to be easy.

The reduction mentioned in the first observation is folklore of the theory of synchronizing automata. Following [188], we put it into a form convenient for further references.

Lemma 3.3 ([188], Proposition 2.1). *Let $\mathbf{C}$ be a class of automata which is closed under taking subautomata and quotients, and let $\mathbf{C}_n$ be the class of all automata with n states in $\mathbf{C}$. Further, let $f\colon \mathbb{Z}^+ \to \mathbb{N}$ be any function such that*

$$f(n) \geqslant f(n-m+1) + f(m) \quad \textit{whenever } n \geqslant m \geqslant 1. \tag{11}$$

If each synchronizing automaton in $\mathbf{C}_n$ that is strongly connected or possesses a unique sink has reset threshold at most $f(n)$, then the same holds for all synchronizing automata in $\mathbf{C}_n$.

Proof. Let $\mathscr{A} = \langle Q, \Sigma \rangle$ be a synchronizing automaton in $\mathbf{C}_n$. Consider the set S of all states to which the automaton can be synchronized, and let $m := |S|$. Let $q \in S$, and let $w \in \Sigma^*$ be a word that resets $\mathscr{A}$ by leaving it at q. Then for any letter $a \in \Sigma$ the word wa resets $\mathscr{A}$ by leaving it in state $q\,.\,a$, so that $q\,.\,a \in S$. This means that restricting the transition function of the automaton $\mathscr{A}$ to the set $S \times \Sigma$ induces a subautomaton $\mathscr{S}$ with state set S. Obviously, the DFA $\mathscr{S}$ is synchronizing and strongly connected and, since the class $\mathbf{C}$ is closed under taking subautomata, we have $\mathscr{S} \in \mathbf{C}_m$. Hence $\mathrm{rt}(\mathscr{S}) \leqslant f(m)$. Let v be a reset word of length $f(m)$ for $\mathscr{S}$.

Now consider the partition π of the set Q into S and $n-m$ single-element classes. It is easily seen that π is a congruence of the automaton $\mathscr{A}$. The quotient $\mathscr{A}/\pi$ is synchronizing and has the class S as a unique sink. Since $\mathbf{C}$ is closed under taking quotients, we have $\mathscr{A}/\pi \in \mathbf{C}_{n-m+1}$. Hence $\mathrm{rt}(\mathscr{A}/\pi) \leqslant f(n-m+1)$. Let u be a synchronizing word of length $f(n-m+1)$ for $\mathscr{A}/\pi$. Then $Q\,.\,u \subseteq S$ and $S\,.\,v$ is a singleton. We have $Q\,.\,uv \subseteq S\,.\,v$, hence the word uv resets $\mathscr{A}$. At the same time

$$|uv| = |u| + |v| = f(n-m+1) + f(m) \leqslant f(n)$$

according to (11). Consequently, $\mathrm{rt}(\mathscr{A}) \leqslant f(n)$. □

The function $(n-1)^2$ satisfies (11). Thus, applying Lemma 3.3 to the class of all DFAs, we see that it suffices to prove Černý's conjecture for the strongly connected synchronizing DFAs and for the synchronizing DFAs with a unique sink. It has long been known ([149], Theorem 1; also see [157], Theorem 6.1) that Černý's conjecture is true for synchronizing DFAs with a unique sink; moreover, the reset threshold of a synchronizing DFA with a unique sink obeys a stronger upper bound than $(n-1)^2$.

Proposition 3.3. *The reset threshold of every synchronizing automaton with n states and a unique sink is not greater than $n(n-1)/2$ and this bound is sharp.*

Proof. Let $\mathscr{A} = \langle Q, \Sigma \rangle$ be a synchronizing automaton with n states and a unique sink z. It is clear that any reset word resets $\mathscr{A}$, just leaving it at z; therefore, any state $q \in Q$ is connected with z by a path. Take an arbitrary subset $S \subseteq Q \setminus \{z\}$ and consider a shortest path that starts at a state in S and ends at z. All states along this path except the first of them belong to $Q \setminus S$, and no state is visited

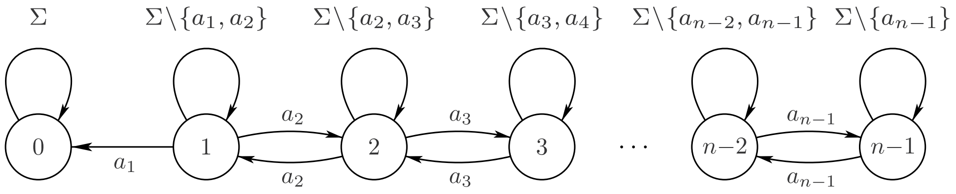

Σ
$\Sigma\backslash\{a_1, a_2\}$
$\Sigma\backslash\{a_2, a_3\}$
$\Sigma\backslash\{a_3, a_4\}$
$\Sigma\backslash\{a_{n-2}, a_{n-1}\}$
$\Sigma\backslash\{a_{n-1}\}$
0
1
2
3
$n-2$
$n-1$
a_1
a_2
a_2
a_3
a_3
a_{n-1}
a_{n-1}

Figure 18. The automaton $\mathscr{R}_n$.

twice. Therefore, the length of such a path does not exceed $|Q \setminus S| = n - |S|$. Now we construct the reset word w by means of the following algorithm:

```
initialisation: initialise $w$ by the empty word $\varepsilon$;
loop: as long as $Q . w \neq \{z\}$ find a shortest path from $Q . w \setminus \{z\}$
        to $z$ and append the word composed of the labels along this path to $w$.
```

After each execution of the main loop the number of states in $Q . w$ reduces at least by 1, and therefore the loop is executed at most $n-1$ times. As explained before, for each $k = 1, \dots, n-1$ the length of the word appended to w at the kth execution of the loop does not exceed

$$n - |Q . w \setminus \{z\}| = n + 1 - |Q . w| \leqslant n + 1 - (n - k + 1) = k.$$

This yields the inequality

$$|w| \leqslant 1 + 2 + \dots + (n-1) = \frac{n(n-1)}{2}.$$

Thus, $\operatorname{rt}(\mathscr{A}) \leqslant n(n-1)/2$.

To prove that this bound is tight consider the DFA $\mathscr{R}_n \coloneqq \langle \{0, 1, \dots, n-1\}, \Sigma \rangle$ with input symbols $\Sigma := \{a_1, \dots, a_{n-1}\}$. The action of symbols is clear from Fig. 18: the symbol a_1 fixes all states except 1, which is mapped to 0, and for $2 \leqslant i \leqslant n-1$ the symbol a_i fixes all states except $i-1$ and i, which it interchanges. This series of automata was presented by Rystsov ([149], Theorem 2; also see [157], Theorem 6.1). We see that $\mathscr{R}_n$ is an automaton with n states, where 0 is a unique sink. Since any state can be taken to 0, the automaton $\mathscr{R}_n$ is synchronizing. We show that $\operatorname{rt}(\mathscr{R}_n) \geqslant n(n-1)/2$.

Let w be a reset word for $\mathscr{R}_n$. For $S = \{s_1, \dots, s_t\} \subseteq Q$ we put $f(S) := \sum_{i=1}^{t} s_i$. Then $f(\{0\}) = 0$ and $f(Q) = n(n-1)/2$. For any S and any symbol a_j we have $f(S . a_j) \geqslant f(S) - 1$, since each symbol either interchanges two neighbouring states or maps 1 and 0 to 0. Taking the equality $Q . w = \{0\}$ into account we obtain

$$0 = f(\{0\}) = f(Q . w) \geqslant f(Q) - |w| = \frac{n(n-1)}{2} - |w|,$$

which yields the inequality $|w| \geqslant n(n-1)/2$. □

The size of the alphabet in automata with a unique sink in the series $\mathscr{R}_n$ showing the tightness of the estimate in Proposition 3.3 grows with the number of states. Can the same reset threshold be attained on series of automata with zero over a *fixed* alphabet? This question is of independent interest and, moreover, it is closely related to one important problem in the theory of formal languages (the so-called *Restivo conjecture*: see the discussion in [43], § 5). The answer to this question is not yet known; all 'slowly synchronizing' series of n-automata with zero over a fixed alphabet appearing in the literature have reset thresholds of the form $n^2/4 + O(n)$, that is, approximately twice as small as that of the series $\mathscr{R}_n$ (see [118] and [138]). The greatest advance so far in this question was made by Ananichev and Vorel [14]: for every $n \equiv 4 \pmod{12}$ starting with $n = 16$ they constructed a synchronizing n-automaton with zero and two input symbols whose reset threshold is $n^2/4 + 2n - 9$.

To sum up, Lemma 3.3 and Proposition 3.3 show that it suffices to improve the Pin–Frankl bound only for strongly connected automata. *Through the end of* § 3.3 *we assume that* $\mathscr{A} = \langle Q, \Sigma\rangle$ *is a strongly connected synchronizing automaton with* n *states*. Then every state in $\mathscr{A}$ is avoidable. Lemma 3 in [183] claims that for any state in $\mathscr{A}$ there is an avoiding word of length at most n. This is not true: we give a counterexample below. The subsequent arguments in [183] are basically correct. Their idea consists in compressing Q to half the size using avoiding words in the following way. Let $v \in \Sigma^*$ be a word such that the size of the subset $S := Q\,.\,v$ is greater than $n/2$. Then by the pigeonhole principle there exists a state $p \in S$ for which there is exactly one state q with the property $p = q\,.\,v$. If the word u avoids q, then $q \notin Q\,.\,u$ by definition, which means that $p \notin Q\,.\,uv \subset Q\,.\,v = S$. Therefore, $|Q\,.\,uv| < |S|$. (Note a distinction from Algorithm 1: avoiding words are not appended from the right, but are 'prepended' from the left.) We arrive at the following algorithm.

Algorithm 2. An algorithm for finding a word compressing a synchronizing automaton $\mathscr{A} = \langle Q, \Sigma\rangle$ to half the size

```
AVOIDING($\mathscr{A}$)
 1: $w \leftarrow \varepsilon$                                  ▷ Initialising the current word
 2: $P \leftarrow Q$                                  ▷ Initialising the current set
 3: while $|P| > |Q|/2$ do
 4:     find $p \in P$ such that there exists exactly one $q \in Q$ such that $p = q\,.\,w$
 5:     take a shortest word $u \in \Sigma^*$ avoiding $q$
 6:     $w \leftarrow uw$                              ▷ Updating the current word
 7:     $P \leftarrow Q\,.\,w$                            ▷ Updating the current set
 8: return $w$
```

Denote the word returned by Algorithm 2 by $w_{1/2}$. By construction it satisfies the inequality $|Q\,.\,w_{1/2}| \leqslant n/2$. In order to complete $w_{1/2}$ to a reset word, we start Algorithm 1 with the word $w := w_{1/2}$ and the subset $P := Q\,.\,w_{1/2}$. By Corollary 3.1 the word appended to $w_{1/2}$ by this algorithm has length at most

$$\sum_{k=2}^{\lfloor n/2\rfloor}\binom{n-k+2}{2} = \sum_{k=2}^{n}\binom{n-k+2}{2} - \sum_{k=\lfloor n/2\rfloor+1}^{n}\binom{n-k+2}{2} = \binom{n+1}{3} - \binom{\lceil n/2\rceil+2}{3}.$$

(The sum $\sum_{k=2}^{n}\binom{n-k+2}{2}$ was calculated in (6), and the sum $\sum_{k=\lfloor n/2\rfloor+1}^{n}\binom{n-k+2}{2}$ can be computed using the same method.) Direct calculations show that

$$\binom{n+1}{3}-\binom{\lceil n/2\rceil+2}{3}=\begin{cases}\dfrac{n(7n^2-6n-16)}{48} & \text{for even } n,\\ \dfrac{7n^3-9n^2-31n-15}{48} & \text{for odd } n.\end{cases}$$

It is obvious that $7n^3-6n^2-16n>7n^3-9n^2-31n-15$ for all positive integers n, hence we can take $n(7n^2-6n-16)/48$ as an upper estimate, applicable to any n, for the length of the word appended to $w_{1/2}$ by Algorithm 1, Thus, combining Algorithms 2 and 1 we can construct reset words of length at most

$$|w_{1/2}|+\frac{n(7n^2-6n-16)}{48}. \tag{12}$$

(Now the rationale behind the term 7/48 in the coefficient of n^3 in the estimate (10) should become clearer for the reader.)

It is obvious that the above construction can be used to improve the Pin–Frankl bound only if there is a nice upper estimate for the length of the word $w_{1/2}$. Since this word is composed of avoiding words, the question reduces to estimating their lengths. As already noted, the estimate claimed in [183] is not correct. Here we present the counterexample announced above; it was proposed in [81]. The automaton $\mathscr{E}_4$ shown in Fig. 19 is strongly connected, synchronizing (for example, it has the reset word ab^2abab^2a), and has four states. It can be verified, however, that the shortest word avoiding 0 is *abbaba* of length 6.

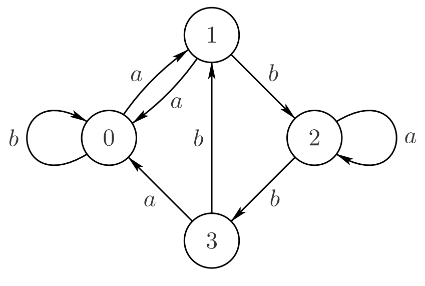


Figure 19. The automaton $\mathscr{E}_4$ (a counterexample to Lemma 3 in [183]).

What can be said about the length of avoiding words? For the convenience of reasoning we introduce the term *avoidability threshold* and the notation $\operatorname{at}(\mathscr{A})$ for the maximum length of shortest words avoiding states of a strongly connected synchronizing DFA $\mathscr{A}=\langle Q,\Sigma\rangle$:

$$\operatorname{at}(\mathscr{A}):=\max_{q\in Q}\{|w_q|\mid w_q \text{ is a shortest word avoiding } q \text{ over } \Sigma\}.$$

It is easy to see that $\operatorname{at}(\mathscr{A})\leqslant\operatorname{rt}(\mathscr{A})+1$. Indeed, if w is a shortest reset word for $\mathscr{A}$, then w avoids all states except s, at which it leaves the automaton $\mathscr{A}$. Since

$\mathscr{A}$ is strongly connected, s is not a zero state, which means that $s\,.\,a \neq s$ for some $a \in \Sigma$. Therefore, the word wa of length $\mathrm{rt}(\mathscr{A}) + 1$ avoids s.

Of course, the estimate $\mathrm{at}(\mathscr{A}) \leqslant \mathrm{rt}(\mathscr{A})+1$ is of no interest from the standpoint of the approach described above. The question of the existence of an upper bound for the avoidability threshold which is linear in n was explicitly formulated in [81]; for now it is open. Based on experiments, Kisielewicz, Kowalski, and Szykuła put forward the following conjecture.

KKS Conjecture ([104], Conjecture 5). *The avoidability threshold of any strongly connected synchronizing automaton with n states does not exceed $2n-2$.*

It is believed that the supposed upper estimate $2n-2$ is tight for automata with n states and three input letters for $n \geqslant 4$ (see [103]). Vorel ([190], Theorem 3.26) presented a series (for $n \geqslant 5$) of strongly connected synchronizing automata with n states and two input letters that have avoidability threshold $2n-3$.

Note that if a linear bound for the avoidability threshold does exist, then the length of the word $w_{1/2}$ returned by Algorithm 2 is estimated from above by a quadratic function of n. Indeed, the main loop of Algorithm 2 is executed at most $\lceil n/2 \rceil$ times, since each execution of this loop reduces the size of the current set at least by 1. Consequently, $w_{1/2}$ is composed of at most $\lceil n/2 \rceil$ avoiding words. Given a quadratic upper bound for $|w_{1/2}|$, the expression (12) provides an estimate of the form $(7/48)n^3 + O(n^2)$ for the reset threshold of strongly connected synchronizing automata with n states.

As of today, however, the best known upper bound for the avoidability threshold is $n^2 - 3n + 4$ (see [177]). This bound does not allow one to bound the length of the word returned by Algorithm 2 strongly enough so as to improve the Pin–Frankl bound using the expression (12). For this reason a word compressing an automaton to half the size was constructed in [177] in a more sophisticated way, which is based on the following key idea.

Lemma 3.4 ([177], Lemma 2). *Let $\mathscr{B} = \langle R, \Sigma \rangle$ be an arbitrary DFA, S be a non-empty subset in R, and T be a non-empty proper subset of S. If $T \nsubseteq S\,.\,u$ for some $u \in \Sigma^*$, then there exists a word $v \in \Sigma^*$ of length at most $|R| - |T|$ such that either $|S\,.\,v| < |S|$ or $T \nsubseteq S\,.\,v$.*

We explain the idea of the use of Lemma 3.4 and refer the reader to [177] for (rather cumbersome) technical details. As above, assume that $\mathscr{A} = \langle Q, \Sigma \rangle$ is a strongly connected synchronizing automaton with n states. Assume that a fragment w of the required word that 'compresses the automaton to half the size' has already been constructed and the set $S := Q\,.\,w$ contains k states, where $n/2 < k < n$. If a word u resets $\mathscr{A}$ by leaving it in some state in $Q \setminus S$, then $q \notin S\,.\,u$ for any $q \in S$. Therefore, the hypothesis of Lemma 3.4 holds for any pair of sets S, $T := \{q\}$, where $q \in S$. Applying the lemma to all such pairs of sets we see that either (case 1) there exists a word $v \in \Sigma^*$ of length at most $n-1$ such that $|S\,.\,v| < |S|$ or (case 2) for *each* state $q \in S$ there exists a word v_q of length at most $n-1$ such that $q \notin S.v_q$. In case 1 the set S is compressed by a word whose length is linear in n (rather then quadratic in n as when Corollary 3.1 is used). In case 2 the word wv_q avoids the state $q \in S$ and w avoids each state in $Q \setminus S$. Therefore, the avoidability threshold of the DFA $\mathscr{A}$ does not exceed $|w| + n - 1$.

Now it is clear how Lemma 3.4 can help one to construct a word that compresses $\mathscr{A}$ to half the size and has length less than $w_{1/2}$. If, in the process of compression, case 2 occurs sufficiently early, when the fragment w already constructed is still short, then we obtain a strong upper bound for $\mathrm{at}(\mathscr{A})$, which can be used for further compression using the method of Algorithm 2. On the other hand, if the first occurrence of case 2 is late enough, that is, when k is close to $n/2$, then the automaton is compressed to k states by a word of length at most $(n-k)(n-1)$.

In fact, the above arguments are not sufficient to improve the Pin—Frankl bound, since it can be calculated that constructing a word that compresses automata to half the size on the basis of only these arguments produces a word of length $n^3/16 + O(n^2)$, while we need a word whose length is bounded by an expression in which the coefficient of n^3 is strictly less than $1/6 - 7/48 = 1/48$. (Here $1/6$ and $7/48$ are the coefficients of n^3 in the Pin–Frankl bound and in the bound for the length of a word that compresses an automaton with $\lfloor n/2 \rfloor$ states to a single state, respectively.) Fortunately, Lemma 3.4 considered in all generality and some other tricks enable one to construct such a word.

The improvement of the bound (9) to (10) obtained in [168] relies on a certain refinement of Lemma 3.4. Note that the term $o(n^3)$ in (10) was not explicitly written out in [168]; it was only mentioned there that direct calculations reduce this term down to $O(n^2 \log n)$. Therefore, although the bound (10) is asymptotically better than (9), the size of automata on which this advantage is perceptible remains unclear.

It was shown in [177] that a word of length at most

$$\frac{85059n^3 + 90024n^2 + 196504n - 10648}{511104}$$

that resets a given synchronizing automaton with n states can be found in time polynomial in n. The same is perhaps true for the reset words constructed in [168], though this was not explicitly mentioned there.

Summarising the above, we can say that the idea of compressing the set of states with the use of avoiding words has moved the problem of finding the reset threshold for automata with n states off the dead centre and at the same time gave rise to new problems concerned with synchronizing automata. However, this idea does not go far enough to prove Černý's conjecture, nor even to obtain an estimate of the form $o(n^3)$ for the function $\mathfrak{C}(n)$.

3.4. The method of extension. Compression strategies implemented in Algorithms 1 and 2 provide upper bounds for the Černý function in the case of an arbitrary DFA which are the best ones at the moment. However, the most impressive partial results in proving Černý's conjecture for special classes of automata have been obtained using another algorithm, which constructs a reset word from bottom up. To describe this algorithm we introduce further notation.

Given a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$, a subset $P \subseteq Q$, and a word $w \in \Sigma^*$, we denote by Pw^{-1} the *full preimage* of P under the action of w:

$$Pw^{-1} := \{q \in Q \mid q \,.\, w \in P\}.$$

Note that $P(uv)^{-1} = (Pv^{-1})u^{-1}$. For a state $q \in Q$ we write qw^{-1} instead of $\{q\}w^{-1}$. The fact that the word $w \in \Sigma^*$ resets the automaton $\mathscr{A}$ by leaving it in state s is expressed in this notation by the equality $sw^{-1} = Q$.

Algorithm 3. An extension algorithm for calculating a reset word for a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$

```
GreedyExtension($\mathscr{A}$)
 1: if $|qa^{-1}| = 1$ for all $q \in Q$ and $a \in \Sigma$ then
 2:     return Failure
 3: else
 4:     $w \leftarrow a$, where $q$ and $a$ are such that $|qa^{-1}| > 1$   ▷ initialising the current word
 5:     $P \leftarrow qa^{-1}$                                           ▷ initialising the current set
 6: while $|P| < |Q|$ do
 7:     if $|Pu^{-1}| \leqslant |P|$ for all $u \in \Sigma^*$ then
 8:         return Failure
 9:     else
10:         take a shortest word $v \in \Sigma^*$ with $|Pv^{-1}| > |P|$
11:     $w \leftarrow vw$                                  ▷ updating the current word
12:     $P \leftarrow Pv^{-1}$                             ▷ updating the current set
13: return $w$
```

A proper subset $P \subset Q$ is said to be *extensible* if there exists a word $v \in \Sigma^*$ such that $|Pv^{-1}| > |P|$; such a word is called an *extending word for the subset P*. Algorithm 3 composes a reset word from extending words, in contrast to Algorithm 1, which deals with compressing words. While the search for compressing words reduces to a search for 2-element subsets in the automaton and can therefore can be implemented in time polynomial in the size of the automaton, the problem of deciding whether there exists a word that extends a given proper subset is **PSPACE**-complete even for a synchronizing DFA (see [27], Proposition 5). It was mentioned in § 3.3 that within the range of issues related to Černý's conjecture considerations can be limited to strongly connected automata. In a strongly connected synchronizing DFA any proper subset is extensible, but the problem of finding a shortest word that extends a given subset remains intractable for such automata as well (see [27], Corollary 14). For this reason it is unclear whether Algorithm 3 can be implemented in time polynomial in the size of the automaton. Some types of DFAs for which such an implementation exists were considered in [28]. An analysis of the performance of Algorithm 3 on the average seems to be an important and exciting research problem, no theoretical results on which have been obtained so far, but some experimental data are available (see [148]).

In the general case no non-trivial upper bound is known on the length of the words that the main loop of Algorithm 3 appends to the current word. However, one can isolate cases where rather strong bounds on this length exist. Given a positive constant α, an automaton with n states is said to be α-*extensible* if each proper non-singleton subset of its states is extensible by a word of length at most αn. The following folklore observation explains the importance of this property.

Proposition 3.4. *Each α-extensible DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ with $n > 2$ states is synchronizing, and its reset threshold does not exceed $1 + \alpha n(n-2)$. In particular, Černý's conjecture holds for 1-extensible automata.*

Proof. If $n > 2$, then the set Q has proper non-singleton subsets. If $S \subset Q$ is such a subset and $v \in \Sigma^*$ is a word extending S, then there exists a state $s \in S$ such that $|sv^{-1}| > 1$. If $v = a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$, then let $i \in \{1, \dots, n\}$ be the greatest index for which $|s(a_i a_{i+1} \cdots a_\ell)^{-1}| > 1$. Then we can take the letter a_i and the (unique) state in the set $s(a_{i+1} \cdots a_\ell)^{-1}$ as a pair (q, a) required for initialising the current word in line 4 of Algorithm 3.

The main loop of Algorithm 3 is executed at most $n - 2$ times, since the loop starts executing when $|P| \geqslant 2$, and each execution of the loop increases $|P|$ at least by 1. Every time the word appended to the current one has length at most αn. Hence the length of the reset word returned by the algorithm does not exceed $1 + \alpha n(n - 2)$. If $\alpha = 1$, then we obtain the bound $1 + n(n - 2) = (n - 1)^2$, which agrees with Černý's conjecture. □

The approach to Černý's conjecture via 1-extensibility traces back to Pin's paper [133]. Pin observed that every DFA in which the number of states is prime and one of the letters acts as a cyclic permutation of the state set is 1-extensible provided that the action of some other letter compresses the state set. Subsequently, Dubuc [62] generalized Pin's result by showing that every synchronizing automaton in which some letter acts as a cyclic permutation of the state set is 1-extensible. (Automata exhibiting this property are called *cyclic*.) Kari [100] proved the 1-extensibility of Eulerian synchronizing automata. (Recall that in applying graph-theoretic terms to an automaton we mean that they are applied to the underlying graph. A graph is *Eulerian* if it contains a directed cycle that uses each edge exactly once.)

Thus, even though the approach to Černý's conjecture via 1-extensibility has proved to be productive in several interesting particular cases, it cannot resolve the general case because there exist strongly connected synchronizing automata that are not 1-extensible. The first example here was the 6-state automaton $\mathscr{K}_6$, discovered by Kari [99]; it is presented in Fig. 20, which shows the action of symbols. This automaton is synchronizing, and its shortest reset word has length 25. Kari found $\mathscr{K}_6$ as a counterexample to a generalized form of Černý's conjecture proposed in Pin's thesis [132], but this automaton is remarkable in several other respects. In particular, one can verify that no subset of $\mathscr{K}_6$ containing more than 4 states is the full preimage of the set $\{2, 3, 4, 5\}$ under the action of a word of length 6 (which is required in the definition of 1-extensibility) or even 7.

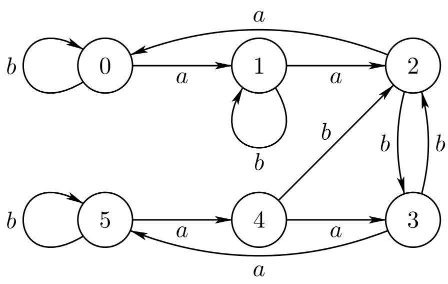


Figure 20. Kari's automaton $\mathscr{K}_6$.

In its turn, 2-extensibility (and thus — by Proposition 3.4 — a quadratic upper bound on the reset threshold) has been established for several classes of synchronizing DFAs (see [80], [154], [158], and [155]). Béal, Berlinkov, and Perrin [18], [19] established a slightly relaxed (but sufficient for deriving a quadratic upper bound on the reset threshold) version of this property for one-cluster synchronizing automata. A DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is called *one-cluster* if there exist a state $q \in Q$ and a letter $a \in \Sigma$ such that there is a path from any other state to q whose edges are labelled only by a. (For instance, the automata $\mathscr{C}_n$ and $\mathscr{W}_n$ shown in Fig. 15 are one-cluster, while Kari's automaton $\mathscr{K}_6$ is not. Examples of practically important one-cluster automata are provided by the decoders of finite maximal prefix codes discussed in § 1.4.) For the letter a in the definition of a one-cluster DFA the automaton $\mathscr{A}$ contains exactly one simple cycle whose edges are labelled by a; we call it an *a-cycle*. It is easy to see that if C is the vertex set of an a-cycle, then $Q\,.\,a^{|Q|-|C|} = C$. Algorithm 4 presented below is a modification of Algorithm 3 with regard to the above.

Algorithm 4. A modified extension algorithm for a one-cluster automaton $\mathscr{A} = \langle Q, \Sigma\rangle$ with an a-cycle C

```
RELATIVEEXTENSION($\mathscr{A}$, $C$, $a$)
 1: $w \leftarrow \varepsilon$                                  ▷ initialising the current word
 2: $P \leftarrow \{q\}$, where $q \in C$                  ▷ initialising the current set
 3: while $|P| < |C|$ do
 4:     if $|Pu^{-1} \cap C| \leqslant |P|$ for all $u \in \Sigma^*$ then
 5:         return Failure
 6:     else
 7:         take a word $v \in \Sigma^*$ of minimum length with $|Pv^{-1} \cap C| > |P|$
 8:     $w \leftarrow vw$                                   ▷ updating the current word
 9:     $P \leftarrow Pv^{-1} \cap C$                       ▷ updating the current set
10: return $a^{|Q|-|C|}w$
```

It was shown in [18] and [19] that the length of the word appended during each execution of the main loop of Algorithm 4 does not exceed $2n$, where $n := |Q|$, and this clearly implies an upper bound of the form $2n^2 + o(n^2)$ for $\mathrm{rt}(\mathscr{A})$. A similar result was obtained by Carpi and D'Alessandro [40], [41]. Steinberg [173], [174] generalized the above approach and improved slightly the explicit expression for the term $o(n^2)$ in the bounds presented in [18], [19], [40], and [41]. Namely, Steinberg proved that $\mathrm{rt}(\mathscr{A}) \leqslant 2n^2 - 9n + 14$, where, as above, $\mathscr{A}$ is a one-cluster synchronizing automaton with n states. In [174] Steinberg also verified Černý's conjecture for one-cluster automata the length of the a-cycle in which is a prime number. In [42] Carpi and D'Alessandro established an upper bound with the best asymptotic behaviour known and proved that the reset threshold of a one-cluster synchronizing automaton with n states does not exceed $2n^2 - 4n + 1 - 2(n-1)\ln(n/2)$.

The leading coefficient 2 in the original bounds in [18], [19], [40], and [41] withstood the improvements obtained with the use of the method of extension. This phenomenon is explained by Berlinkov's result [22]: he presented a series of synchronizing one-cluster DFAs $\mathscr{B}_n = \langle\{0, 1, \dots, n-1\}, \{a, b\}\rangle$, where for any $\alpha < 2$ there are automata that are not α-extensible. The automaton $\mathscr{B}_n$ is shown in Fig. 21.

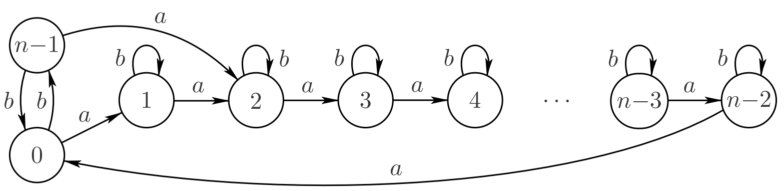


Figure 21. The automaton $\mathscr{B}_n$.

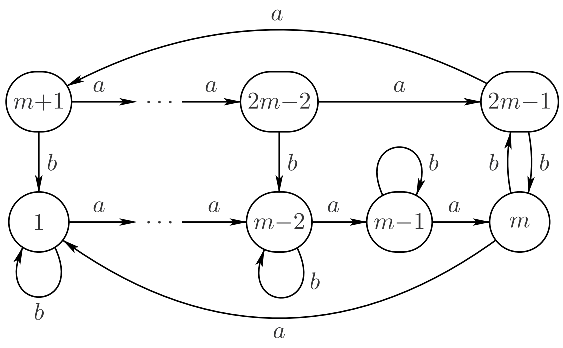


Figure 22. The automaton $\mathscr{KS}_{2m-1}$.

The action of the letters in $\mathscr{B}_n$ is defined by

$$i\,.\,a := \begin{cases} i+1 & \text{if } i < n-2, \\ 0 & \text{if } i = n-2, \\ 2 & \text{if } i = n-1; \end{cases}$$

$$i\,.\,b := \begin{cases} n-1 & \text{if } i = 0, \\ i & \text{if } 0 < i < n-1, \\ 0 & \text{if } i = n-1. \end{cases}$$

It can be verified that the shortest extending word for the subset $\{0, n-1\}$ in $\mathscr{B}_n$ is the word $a^{n-2}ba^{n-2}$ of length $2n-3$. Therefore, $\mathscr{B}_n$ is not α-extensible for any $n > 3/(2-\alpha)$.

There are some other types of synchronizing automaton whose 2-extensibility has been proved (see [42], [154], [158], and [155]). However, the hope that a quadratic (in n) upper bound for the reset threshold of arbitrary synchronizing automata with n states can be obtained by proving their α-extensibility for some α and then using Proposition 3.4 has not come true.

Kisielewicz and Szykuła [105] constructed a series of synchronizing automata $\mathscr{KS}_{2m-1} = \langle\{1, 2, \dots, 2m-1\}, \{a, b\}\rangle$ that for each α contains an automaton that is not α-extensible. The automaton $\mathscr{KS}_{2m-1}$ is presented in Fig. 22. The action of letters is defined by

$$i \,.\, a := \begin{cases} 1 & \text{if } i = m, \\ m+1 & \text{if } i = 2m-1, \\ i+1 & \text{otherwise;} \end{cases}$$
$$i \,.\, b := \begin{cases} i & \text{if } 1 \leqslant i \leqslant m-1, \\ 2m-1 & \text{if } i = m, \\ i-m & \text{otherwise.} \end{cases}$$

It can be verified that in $\mathscr{KS}_{2m-1}$ the length of a shortest word extending the subset $\{m+1, \dots, 2m-1\}$ (that is, the 'first line' in Fig. 22) is $2 + m\lceil (m-2)/2 \rceil$. This expression grows more rapidly than any linear function of $2m-1$, and therefore for any α we can choose a value of m so that the automata $\mathscr{KS}_{2m-1}$ are not α-extensible starting from this value.

The work [28] contains an algebraic analysis of the method of extension, which has led to some generalizations of this method and some improvement of the results obtained with its use. These generalizations are presented in the second paper of our cycle, in the section devoted to relationships between the theory of synchronizing DFAs and the theory of non-negative matrices and Markov chains. The paper [87] is also worth mention: it contains results that reduce the 'gap' between Algorithm 1 (compression) and Algorithm 3 (extension) in a certain sense. Namely, it was shown in [87] that for most types of automaton considered in this subsection, that is, for automata to which the method of extension is well applicable, the compression algorithm performs better than in the general case. For instance, in the case of synchronizing cyclic automata Algorithm 1 returns a reset word of length at most $n^2 \ln n$ (see [87], Theorem 13).

3.5. The summary of partial results. This subsection is provided for reference. It contains the main results that establish Černý's conjecture as restricted to various classes of DFAs. In addition, we list classes of automata for which Černý's conjecture remains open, but there exists a quadratic (in the number of states) upper bound for the reset threshold. Some results discussed in this subsection have already been mentioned above; yet we deem it appropriate to include them in this list of references to make it complete. The information about each class of automata is presented in the following form:

- the *definition* of the class;
- an *upper bound* for the reset threshold of synchronizing automata with n states in the class;
- a *method* for deriving the upper bound;
- a *lower bound* for the Černý function (the maximum of the reset thresholds of synchronizing automata with n states) in the class;
- *comments* (if necessary).

In cases where the upper bound matches the lower bound, that is, the restriction of the Černý function to the corresponding class is known, the class is labelled by the superscript †.

A. Types of DFAs with reset threshold $(n-1)^2$ or close to $(n-1)^2$.

A1†. Cyclic automata.

• *Definition*: A DFA is said to be *cyclic* if one of its letters acts as a cyclic permutation on the set of all states.

• *Upper bound*: $(n-1)^2$ (see [62]).

• *Method*: 1-extensibility (see § 3.4).

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ are cyclic.

A2. One-cluster automata with a cycle of prime length.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be *one-cluster* if there exist a state $q \in Q$ and a letter $a \in \Sigma$ such that there is a path from any other state to q whose edges are labelled only by a. For every such letter a the automaton $\mathscr{A}$ contains exactly one simple cycle whose edges are labelled by a; the class considered here consists of one-cluster automata in which for some particular letter the corresponding cycle has prime length.

• *Upper bound*: $(n-1)^2$ (see [174]).

• *Method*: 2-extensibility (see § 3.4) with some additional technical subterfuges.

• *Lower bound*: if n is a prime number, then $(n-1)^2$, since in this case the Černý automaton $\mathscr{C}_n$ belongs to this class; for composite values of n the sharp lower bound is unknown.

A3†. Orientable automata.

• *Definition*: a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be *orientable*[17] if the states in Q can be arranged in a cyclic order[18] $q_0, q_1, \dots, q_{n-1}$, where $n = |Q|$, so that the action of each letter in Σ preserves this cyclic order. To describe this condition more precisely we say that a sequence $p_0, p_1, \dots, p_{m-1}$ of (not necessarily distinct) states in Q is *properly oriented* if the removal of all but one state from the block $p_i = p_{i+1\,(\mathrm{mod}\ m)} = \dots = p_{i+k\,(\mathrm{mod}\ m)}$ of identical successive states produces a subsequence of a cyclically permuted sequence $q_0, q_1, \dots, q_{n-1}$. A letter $a \in \Sigma$ *preserves the cyclic order* if the sequence of images $q_0 \,.\, a, q_1 \,.\, a, \dots, q_{n-1} \,.\, a$ is properly oriented.

• *Upper bound*: $(n-1)^2$ (see [64]).

• *Method*: this is based on the following property of orientable automata: for any word $w \in \Sigma^*$ and any interval $I \subseteq Q$ the full preimage Iw^{-1} is an interval[19] (see [64], Lemma 1). If $w := a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$, is the reset word returned by Algorithm 3 and $q \in Q$ is the state from which Algorithm 3 starts operation, then all subsets $q(a_i a_{i+1} \cdots a_\ell)^{-1}$ are distinct and are non-singleton intervals. (A good illustration is the scheme in Fig. 11, where we see that the lift from the state

[17] This concept was originally introduced by Eppstein in [64], where such automata were called monotonic. Here we use the terminology from [10] and reserve the term 'monotonic' for a more special class of automata.

[18] Although the notion of a cyclic order on a finite set is intuitively clear, we recall here the formal definition. A *cyclic order* on a set Q is a ternary relation $\Upsilon \subset Q \times Q \times Q$ such that for any $p, q, r \in Q$ the following conditions hold:
– if $(p, q, r) \in \Upsilon$, then $(q, r, p) \in \Upsilon$;
– if $(p, q, r) \in \Upsilon$, then $(r, q, p) \notin \Upsilon$;
– if $(p, q, r) \in \Upsilon$ and $(p, r, u) \in \Upsilon$, then $(p, q, u) \in \Upsilon$;
– if p, q and r are distinct, then either $(p, q, r) \in \Upsilon$ or $(r, q, p) \in \Upsilon$.

[19] A non-empty subset I of a cyclically ordered set Q is called an *interval* if for any distinct $p, r \in I$ the subset I contains all elements $q \in Q$ such that the sequence p, q, r is properly oriented. According to this definition all singletons are intervals.

$q := 1$ to the set $Q := \{0, 1, 2, 3\}$ is accomplished only via intervals.) It can easily be calculated that a cyclically ordered n-element set has $(n-1)^2$ non-singleton intervals, and so $\ell \leqslant (n-1)^2$.

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ are orientable.

• *Comments*: In [10], Theorem 2.2, the upper bound from [64] was generalized to a wider class of weakly orientable synchronizing automata. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is *weakly orientable* if the states in Q can be arranged in a cyclic order $q_0, q_1, \dots, q_{n-1}$, where $n = |Q|$, so that for each $a \in \Sigma$ one of the sequences $q_0 \,.\, a, q_1 \,.\, a, \dots, q_{n-1} \,.\, a$ and $q_{n-1} \,.\, a, q_{n-2} \,.\, a, \dots, q_0 \,.\, a$ is properly oriented. Another generalization, which has a bulky formulation, is discussed in A4 below.

A4[†]. Automata respecting the intervals of a directed graph.

• *Definition*: Let Δ be a graph whose vertex set is the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$. (Warning: in general, the graph Δ is different from the underlying graph of the DFA $\mathscr{A}$, although they have the same vertex set.) Given $p, r \in Q$, we call a path from p to r in the graph Δ *singular* if both p and r occur only once on this path. We define $[p, r]$ as an empty set if Δ contains no path from p to r and as the set

$$\{q \in Q \mid q \text{ lies on a singular path from } p \text{ to } r\}$$

if Δ contains a path from p to r. The graph Δ is said to be *dense* if for all $p, q, r \in Q$ it follows from the condition $[p, r], [r, p], [p, q], [q, r] \neq \varnothing$ that either $q \in [p, r]$ or $q \in [r, p]$. We say that $\mathscr{A}$ *respects the intervals of the graph* Δ if the following conditions hold for all $p, r \in Q$ and every letter $a \in \Sigma$:

– if $[p, r] \neq \varnothing$, then $[p \,.\, a, r \,.\, a] \neq \varnothing$;
– if both $[p, r], [r, p] \neq \varnothing$, then $[p, r] \,.\, a \subseteq [p \,.\, a, r \,.\, a]$;
– if $p \,.\, a = r \,.\, a$, then at least one of the sets $[p, r] \,.\, a$ or $[r, p] \,.\, a$ is a singleton.

This class contains strongly connected automata respecting the intervals of some weakly connected dense graph on their state sets.

• *Upper bound*: $(n-1)^2$ (see [84], Corollary 2.1).

• *Method*: induction on the number of states and an examination of particular cases.

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ belong to this class (the cycle $0 \to 1 \to 2 \to \dots \to n-1 \to 0$ is taken as the graph Δ).

A5[†]. 2-junction automata.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is called a 2-*junction automaton* if for every letter $a \in \Sigma$ one of the following condition holds:

– all states that are not fixed by a, with the exception of at most two of them, are fixed by all letters in $\Sigma \setminus \{a\}$, and each of the exceptional states is fixed by all letters but exactly one in $\Sigma \setminus \{a\}$;
– all states that are not fixed by a, with the exception of one of them, are fixed by all letters in $\Sigma \setminus \{a\}$, and the exceptional state is fixed by all letters but exactly two in $\Sigma \setminus \{a\}$.

• *Upper bound*: $(n-1)^2$ (see [85], Theorem 2.1).

• *Method*: reduction to automata of types A1 and A6 and the examination of the remaining cases.

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ belong to this class.

A6. Eulerian automata.

A DFA is said to be *Eulerian* if the underlying graph contains a cycle which uses every edge exactly once.

• *Upper bound*: $n^2 - 3n + 3$ (see [100], Theorem 3).

• *Method*: 1-extensibility (see § 3.4) with an additional improvement: for Eulerian automata the length of the word appended at every execution of the main loop of Algorithm 3 does not exceed $n - 1$. Therefore, the algorithm returns a reset word of length at most $1 + (n - 2)(n - 1) = n^2 - 3n + 3$.

• *Lower bound*: $\lfloor (n^2 - 3)/2 \rfloor$ (see [178]).

• *Comments*: The upper bound $n^2 - 3n + 3$ was extended in [173], Theorem 3, to the class of pseudo-Eulerian automata. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be *pseudo-Eulerian* if the letters of Σ can be assigned positive weights in such a way that for every state $q \in Q$ both the total weight of the labels of all edges with tails at q and the total weight of the labels of all edges with heads at q are equal to 1. (For Eulerian automata the last condition holds true if each letter is assigned weight $1/|\Sigma|$.)

The lower bound $\lfloor (n^2 - 3)/2 \rfloor$ from [178] is attained on a series of Eulerian automata with four input letters. The automata in this series also suggest that the upper bound $n-1$ on the length of the word appended at each execution of the main loop of Algorithm 3 on Eulerian automata cannot be improved in general. In the class of Eulerian automata with two input letters Martyugin (unpublished) discovered a series of synchronizing n-state automata (where n is odd) with a conjectural reset threshold of $(n^2 - 5)/2$. This conjecture has been confirmed by an exhaustive search of all Eulerian n-state automata with two input letters for $n = 5, 7, 9, 11$; moreover, for each of these values of n Martyugin's automaton has turned out to be the only one with reset threshold $(n^2 - 5)/2$. However, in the general case the conjecture remains unproved and the greatest lower bound for the maximum reset threshold of synchronizing Eulerian n-automata with two input letters (where n is odd) that is available at the moment is $(n^2 - 3n + 4)/2$ from [86].

A7. Automata with a letter of small rank.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is called an *automaton with a letter of small rank* if there is a letter $a \in \Sigma$ such that $|Q\,.\,a| \leqslant \sqrt[3]{6|Q| - 6}$.

• *Upper bound*: $(n - 1)^2$ (see [28], Corollary 13).

• *Method*: a corollary to Theorem 6 in [28], established using an improved version of the method of extension.

• *Lower bound*: unknown.

• *Comments*: A pioneering result on the Černý problem for automata with a letter of small rank was obtained in [134], Theorem 1. In [134] a stronger constraint $|Q\,.\,a| \leqslant 1 + \log_2 |Q|$ was imposed on the 'degree of compression' of the letter a.

A8. Automata without involutions.

• *Definition*: A DFA $\mathscr{A}$ is called an *automaton without involutions* if there is no state q and word w in $\mathscr{A}$ such that $q\,.\,w \neq q = q\,.\,w^2$. In algebraic terms this means that the transition monoid of the automaton $\mathscr{A}$ contains no subgroups of even orders.

• *Upper bound*: $(n - 1)^2$ (see [181], Theorem 6).

• *Method*: a modification of the approach used in [180] (see B2).

• *Lower bound*: Only the linear lower bound $n+\lfloor n/2\rfloor-2$, which follows from [7], is known. In the case of strongly connected synchronizing automata without involutions there are no examples so far with reset threshold higher than $n-1$.

• *Comments*: The proof in [181] contains some unclear points.

B. Types of DFAs with reset threshold from n to $n(n-1)/2$.

B1†. Automata with zero.

• *Definition*: a DFA is called an *automaton with zero* if it contains a state fixed by every letter.

• *Upper bound*: $n(n-1)/2$ ([149], Theorem 1; also see [157], Theorem 6.1).

• *Method*: see the proof of Proposition 3.3.

• *Lower bound*: $n(n-1)/2$ for automata with an unlimited alphabet (see the proof of Proposition 3.3). In the case of automata with two input symbols the best known lower bound for the maximum reset threshold of synchronizing n-state automata with zero is $n^2/4+2n-9$; it is attained for $n\equiv 4 \pmod{12}$, starting with $n=16$, on a series of automata presented in [14].

B2. Aperiodic automata.

• *Definition*: a DFA $\mathscr{A}$ is said to be *aperiodic* if there is no state q in $\mathscr{A}$ and word w over the input alphabet such that $q\,.\,w\neq q=q\,.\,w^k$ for some k. In algebraic terms this means that the transition monoid of the automaton $\mathscr{A}$ contains no non-trivial subgroups.

• *Upper bound*: $n(n-1)/2$ (see [180], Theorem 10).

• *Method*: the class of aperiodic automata is closed under taking subautomata and quotients, and the function $n(n-1)/2$ satisfies inequality (11). By Lemma 3.3 it suffices to verify the upper bound $n(n-1)/2$ for strongly connected synchronizing aperiodic automata and synchronizing aperiodic automata with zero. In the second case it holds for every DFA with zero (see B1). A strongly connected aperiodic DFA $\mathscr{A}=\langle Q,\Sigma\rangle$ is always synchronizing ([180], Theorem 9) and admits a non-trivial relation of partial order $\leqslant$ that is invariant under the action of input letters: if $p\leqslant r$ for some $p,r\in Q$, then $p\,.\,a\leqslant r\,.\,a$ for every letter $a\in\Sigma$. Therefore, if $p\,.\,w=r\,.\,w$ for some word $w\in\Sigma^*$, then this word takes every state $q\in Q$ such that $p\leqslant q\leqslant r$ to state $p\,.\,w=r\,.\,w$. This makes it possible to compose a reset word of length $\leqslant n(n-1)/2$ from words of length $\leqslant n-1$ taking maximal states (with respect to the order $\leqslant$) to minimal ones or vice versa. (Since the automaton $\mathscr{A}$ is strongly connected, for any two states there is a word of length $\leqslant n-1$ that takes one of these states to the other.) It is clear that either the number of maximal or the number of minimal states does not exceed $n/2$. For an accurate implementation of the approach described above one should perform induction on the number of states.

• *Lower bound*: only a linear lower bound $n+\lfloor n/2\rfloor-2$ is currently known; it follows from [7]. In the case of strongly connected aperiodic automata no examples of automata with reset threshold higher than $n-1$ are available, and it is conjectured that the reset threshold for strongly connected aperiodic n-state automata does not exceed $n-1$. This conjecture is confirmed by an exhaustive search of strongly connected aperiodic n-state automata with two input letters for $n\leqslant 11$ and of ones with three input letters for $n\leqslant 7$ (see [176], § 7.4.4). Also see the comments to D3.

• *Comments*: In [188] the upper bound for the reset threshold of strongly connected aperiodic n-state automata was improved to $\lfloor n(n+1)/6 \rfloor$.

B3†. Automata with transition monoid in **EDS**.

• *Definition*: The class **DS** consists of the finite monoids M such that for all $x, y, z \in M$ the following implication holds[20]:

$$\text{if} \quad MxM = MyM = MzM = Mx^2M, \quad \text{then} \quad MxM = MyzM.$$

An element e of a monoid M is called an *idempotent* if $e^2 = e$. The class **EDS** consists of all finite monoids M with the following property: the submonoid generated by all idempotents in M lies in **DS**. Here we consider automata whose transition monoids belong to **EDS**.

• *Upper bound*: $n(n-1)/2$ (see [5], Corollary 4.3).

• *Method*: using information about linear representations of monoids in **EDS** which is available from the classical theory of semigroup representations (the Munn–Ponizovsky theory).

• *Lower bound*: $n(n-1)/2$ for a DFA with unlimited alphabet, since the Rystsov automata $\mathscr{R}_n$ (see the proof of Proposition 3.3) belong to this class. Almost nothing is known about lower bounds for the maximum of the reset thresholds for synchronizing DFAs over a fixed alphabet in this class or for strongly connected synchronizing DFAs in this class.

B4. Decoders of finite maximal prefix codes.

• *Definition*: A set $X \subset \Sigma^+$ is called a *prefix code over* Σ if no word in X is a prefix of another word in X. A prefix code is *maximal* if it is contained in no other prefix code over Σ. A *decoder of a finite maximal prefix code* $X \subset \Sigma^+$ is the DFA $\mathscr{A}_X = \langle Q, \Sigma \rangle$, where Q is the set of all prefixes of words in X and the transition function is defined by

$$q\,.\,a := \begin{cases} qa & \text{if } qa \text{ is a prefix of a word in } X, \\ \varepsilon & \text{if } qa \in X. \end{cases}$$

• *Upper bound*: $2 + (n+h-1)(h^3-h)/6$, where $h = \lceil \log_{|\Sigma|} n \rceil$ (see [28], Corollary 17).

• *Method*: an improved method of compression.

• *Lower bound*: $2n-5$ for even values of n and $2n-7$ for odd values of n if $|\Sigma| = 2$ (see [33]). For $|\Sigma| \geqslant 3$ a lower bound $2\lceil n/(|\Sigma|+1) \rceil$ was obtained in [28], Theorem 19.

B5. Weakly monotonic automata.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be *weakly monotonic* if the state set Q admits a linear order $\leqslant$ such that for every letter $a \in \Sigma$ either for all $p, r \in Q$ the relation $p \leqslant r$ yields the relation $p\,.\,a \leqslant r\,.\,a$ or for all $p, r \in Q$ the relation $p \leqslant r$ yields $p\,.\,a \geqslant r\,.\,a$. (In other words, the function $q \mapsto q\,.\,a$ is either monotonically non-decreasing or monotonically non-increasing.)

[20]For the reader familiar with elements of the theory of semigroups we give an equivalent and more standard definition: a finite monoid M belongs to the class **DS** if and only if every regular $\mathcal{D}$-class of M is a subsemigroup of M.

• *Upper bound*: $n(n-1)/2$ in the general case; $2n-3$ if $\mathscr{A}$ has a letter c such that the function $q \mapsto q\,.\,c$ is a monotonically decreasing one-to-one correspondence (see [10], Proposition 3.2).

• *Method*: the general case is based on the following three ideas. 1) In a weakly monotonic automaton, for every word $w \in \Sigma^*$ and every interval I of the totally ordered set $\langle Q, \leqslant \rangle$ the full preimage Iw^{-1} is an interval[21]. 2) If $w := a_1 \cdots a_\ell$, where $a_1, \dots, a_\ell \in \Sigma$ is the reset word returned by Algorithm 3 and $q \in Q$ is a state from which Algorithm 3 starts operation, then all subsets $q(a_i a_{i+1} \cdots a_\ell)^{-1}$ are distinct and are non-singleton intervals. 3) A totally ordered n-element set contains $n(n-1)/2$ non-singleton intervals. This yields the inequality $\ell \leqslant n(n-1)/2$.

If there is a letter that acts as a monotonicaly decreasing one-to-one map, then reduction to a monotone DFA is used (see C1).

• *Lower bound*: $2n-3$ for $n \equiv 3 \pmod 4$ (see [10], Proposition 3.1).

B6†. 0-monotonic automata.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ with zero 0 is said to be 0-*monotonic* if the state set $Q \setminus \{0\}$ admits a total order $\leqslant$ such that for every letter $a \in \Sigma$ and any states $p, r \in Q \setminus \{0\}$ the relations $p \leqslant r$ and $p\,.\,a, r\,.\,a \neq 0$ imply the inequality $p\,.\,a \leqslant r\,.\,a$.

• *Upper bound*: $n + \lfloor n/2 \rfloor - 2$ (see [7], Theorem 1).

• *Method*: reduction to some properties of monotonic DFAs (see C1).

• *Lower bound*: $n + \lfloor n/2 \rfloor - 2$ (see [7], Theorem 2).

B7. Finitely generated automata.

• *Definition*: A DFA $\mathscr{A}$ with an input alphabet Σ is said to be *finitely generated* if there exists a finite subset $W \subset \Sigma^*$ such that $\operatorname{Sync}\mathscr{A} = \Sigma^* W \Sigma^*$. In algebraic terms this means that the language of reset words of the automaton $\mathscr{A}$ is a finitely generated ideal of the monoid Σ^*.

• *Upper bound*: $3n-5$ (see [139], Theorem 4).

• *Method*: using the characterization of finitely generated automata from [139], Theorem 1, and the estimate from [134], Proposition 5.

• *Lower bound*: $n-1$ (see [139], the discussion of Example 1).

C. Types of DFAs with reset threshold $n-1$.

C1†. Monotonic automata.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be *monotonic* if the state set Q admits a total order $\leqslant$ such that for every letter $a \in \Sigma$ and all $p, r \in Q$ the inequality $p \leqslant r$ implies that $p\,.\,a \leqslant r\,.\,a$.

• *Upper bound*: $n-1$ (see [11]).

• *Method*: induction on the number of states.

• *Lower bound*: $n-1$; an example is provided by the series of monotonic automata $\mathscr{M}_n := \langle \{0, 1, 2, \dots, n-1\}, \{a\} \rangle$, where the action of the input letter is defined by

$$i.a := \begin{cases} i-1 & \text{if } i > 0, \\ 0 & \text{if } i = 0. \end{cases}$$

[21] A non-empty subset I of a totally ordered set $\langle Q, \leqslant \rangle$ is called an *interval* if for any $p, r \in I$ such that $p \leqslant r$ the subset I contains all elements $q \in Q$ such that $p \leqslant q \leqslant r$.

• *Comments*: As far as we can judge by the English abstract of the paper [56] (in Chinese) by Cui, He, and Sun, they established the upper bound $n-1$ for the DFAs $\mathscr{A} = \langle Q, \Sigma\rangle$ such that the state set Q admits a partial order $\leqslant$ with respect to which the set Q contains the smallest and largest elements and for every input letter $a \in \Sigma$ and all $p, r \in Q$ the inequality $p \leqslant r$ implies the relation $p \,.\, a \leqslant r \,.\, a$.

C2†. Generalized monotonic automata.

• *Definition*: a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is said to be *ρ-monotonic*, where ρ is a congruence on $\mathscr{A}$, if the state set Q admits a partial order $\leqslant$ such that two states are comparable with respect to $\leqslant$ if and only if they belong to the same ρ-class and for every letter $a \in \Sigma$ and all $p, r \in Q$ the inequality $p \leqslant r$ implies the relation $p \,.\, a \leqslant r \,.\, a$. A DFA $\mathscr{A}$ is said to be *generalized monotonic* if it admits a chain of congruences

$$\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$$

such that ρ_0 is the equality relation, ρ_ℓ is the universal relation on Q, and for every $i = 1, \dots, \ell$ the quotient $\mathscr{A}/\rho_{i-1}$ is ρ_i/ρ_{i-1}-monotonic.

• *Upper bound*: $n-1$ (see [12], Theorem 1.2).

• *Method*: induction on the number of states.

• *Lower bound*: $n-1$, since the automata $\mathscr{M}_n$ defined in C1 belong to this class.

C3†. Automata with transition monoid in **DS**.

• *Definition*: The class of monoids **DS** was defined in B3 above. Here we consider DFA whose transition monoids belong to **DS**.

• *Upper bound*: $n-1$ (see [5], Theorem 2.6).

• *Method*: employing the information about linear representations of monoids in **DS** that is available from [3].

• *Lower bound*: $n-1$, since the automata $\mathscr{M}_n$ described in C1 belong to this class.

• *Comments*: This class of automata covers a number of classes for which the upper bound $n-1$ was established earlier by Rystsov (see [153], [157], and [156]), as well as the class of the so-called weakly cyclic automata for which the same estimate was proved by Ryzhikov [160]. In terms of transition monoids the works [153], [157], and [156] are devoted to the study of automata with commutative or close to commutative transition monoid, and [160] concerns automata whose transition monoids are $\mathcal{R}$-trivial[22]. It is easily seen that both commutative and $\mathcal{R}$-trivial finite monoids belong to **DS**.

C4†. Automata with two idempotent letters.

• *Definition*: A DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is called an *automaton with idempotent letters* if $q \,.\, a = q \,.\, a^2$ for any $q \in Q$ and $a \in \Sigma$. The class under consideration contains automata of this type in which $|\Sigma| = 2$.

• *Upper bound*: $n-1$ (see [189], Proposition 5).

• *Method*: the class of automata with two idempotent letters is closed under taking subautomata and quotients, and the function $n-1$ satisfies (11). By Lemma 3.3 it suffices to prove the upper bound $n-1$ in the strongly connected case and in the case of automata with zero. It can be shown that a strongly connected synchronizing automaton with two idempotent letters has at most two states. For

[22]A monoid M is called *$\mathcal{R}$-trivial* if for any $x, y \in M$ the relation $xM = yM$ implies that $x = y$.

synchronizing automata with two idempotent letters and zero the proof is based on induction on the number of states.

• *Lower bound*: $n-1$; an example is provided by the series of automata $\mathscr{F}_n := \langle\{1,2,\dots,n\},\{a,b\}\rangle$, where the action of input letters is defined by

$$i\,.\,a := \begin{cases} i & \text{if } i \text{ is odd or } i=n,\\ i+1 & \text{if } i \text{ is even and } i<n; \end{cases}$$

$$i\,.\,b := \begin{cases} i & \text{if } i \text{ is even or } i=n,\\ i+1 & \text{if } i \text{ is odd and } i<n. \end{cases}$$

• *Comments*: If $|\Sigma|>2$, then the reset threshold of an n-state automaton with idempotent letters can be as large as $\lfloor n^2/2\rfloor - 2n + 2$ (see [189], Corollary 3, or [61], § 3).

C5. Media.

• *Definition*: Given a DFA $\mathscr{A} = \langle Q,\Sigma\rangle$, two letters $a,b\in\Sigma$ are said to be *reverses* of each other if for any distinct states $p,r\in Q$ the relation $p\,.\,a = r$ holds if and only if $r\,.\,b = p$. A word in Σ^* is called *consistent* if it contains no two letters that are reverses of each other. A word $w\in\Sigma^*$ is called *vacuous* if, for each letter that it contains, w contains equal numbers of copies of this letter and its reverse. A word $a_1\cdots a_\ell$, where $a_1,\dots,a_\ell\in\Sigma$, is called *stepwise effective for a state* $q\in Q$ if $q\,.\,a_1\neq q$ and $q\,.\,a_1\cdots a_i \neq q\,.\,a_1\cdots a_{i-1}$ for all $i=2,\dots,\ell$. A *medium* is a DFA $\mathscr{A}=\langle Q,\Sigma\rangle$ satisfying the following axioms:

- every letter from Σ has a unique reverse;
- for any two distinct states $p,r\in Q$ there exists a consistent word $w\in\Sigma^*$ such that $p\,.\,w = r$;
- if a word $w\in\Sigma^*$ is stepwise effective for a state $q\in Q$, then $q\,.\,w = q$ if and only if w is vacuous;
- if $p\,.\,u = r\,.\,v$, the word u is stepwise effective for p, the word v is stepwise effective for r, and both u and v are consistent, then uv is also consistent.

• *Upper bound*: $n-1$ (see [65], Theorem 1).

• *Method*: the problem is reduced to a search for a shortest path in an acyclic auxiliary graph on the vertex set Q.

• *Lower bound*: in [65] the upper bound $n-1$ was claimed to be tight, but an example of a medium with reset threshold $n-1$ was only presented for $n=6$.

D. Types of DFAs with a quadratic bound for the reset threshold.

D1. One-cluster automata.

• *Definition*: see A2.

• *Upper bound*: $2n^2-4n+1-2(n-1)\ln(n/2)$ (see [42], Corollary 1).

• *Method*: 2-extensibility (see § 3.4), with some additional technical subterfuges.

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ are one-cluster.

• *Comments*: Berlinkov [23] called an automaton a *quasi-one-cluster automaton with respect to a non-negative integer* d if it has a letter such that the total length of all but one simple cycles labelled by this letter does not exceed d. (One-cluster automata are exactly quasi-one-cluster automata with respect to $d=0$; thus, the parameter d measures the 'deviation' from the one-cluster structure.) It was shown in [23], Corollary 11, that the reset threshold of an n-state quasi-one-cluster automaton with respect to d does not exceed $2^d(n-d+1)(2n-d-2)$.

D2. Automata with full transition monoid.

• *Definition*: A DFA is called an *automaton with full transition monoid* if its transition monoid is equal to the full monoid of transformations of the state set.

• *Upper bound*: $2n^2-6n+5$ (see [80], Theorem 7).

• *Method*: 2-extensibility (see § 3.4).

• *Lower bound*: Theorem 4 in [80] presents a series of n-state automata with $n+1$ input letters which have full transition monoids and reset threshold $n(n-1)/2$. In [61], § 5.1, a similar series of n-state automata with n input letters and reset threshold $n(n+1)/2$ was constructed; the automata in this series can be shown to have full transition monoids too. No non-trivial lower bounds have been obtained so far for DFAs with full transition monoid over a fixed alphabet.

D3. Automata with simple idempotents.

• *Definition*: A DFA is called an *automaton with simple idempotents* if every input letter either acts as a permutation on the state set or fixes all states but one.

• *Upper bound*: $2(n-1)^2$ (see [158], Theorem 4).

• *Method*: 2-extensibility (see § 3.4).

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ belong to this class.

• *Comments*: It was shown in Rystsov's recent paper [159] that the reset threshold of an aperiodic n-state automaton with simple idempotents does not exceed $n-1$.

D4. Quasi-Eulerian automata.

• *Definition*: A DFA $\mathscr{A}=\langle Q,\Sigma\rangle$ is said to be *quasi-Eulerian with respect to a positive integer d* if it satisfies the following two conditions:

  – there is a subset $E_d\subset Q$ containing $|Q|-d$ states such that exactly one of these states (say, $s\in E_d$) can have incoming edges from the set $Q\setminus E_d$;
  – the letters in Σ can be assigned positive weights in such a way that for every state $p\in Q$ the total weight of the labels of all edges with tails at q are equal to 1, and for every state $r\in E_d\setminus\{s\}$ the total weight of the labels of all edges with heads at r is 1.

• *Upper bound*: $2^d(n-d+1)(n-1)$ (see [23], Theorem 8).

• *Method*: an improved method of extension.

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ are quasi-Eulerian with respect to $d=1$. (As the set E_1 one can take $\{1,2,\dots,n-1\}$ and as the special state s, the state 1: see Fig. 15. The weights assigned to the letters a and b can be chosen as two arbitrary positive numbers adding up to 1.)

D5. Regular automata.

• *Definition*: A DFA $\mathscr{A}=\langle Q,\Sigma\rangle$ is said to be *regular* if there is a collection of words $W\subset\Sigma^*$ containing an empty word and satisfying the following conditions:

  – the length of any word in W is less than the number of states of the automaton;
  – there is an integer $m\geqslant 1$ such that for any two states $p,r\in Q$ there are exactly m words in W that take the state p to r.

• *Upper bound*: $2(n-1)^2$ (see [155], Theorem 7) or [154], Theorem 7).

• *Method*: 2-extensibility (see § 3.4).

• *Lower bound*: $(n-1)^2$, since the Černý automata $\mathscr{C}_n$ are regular. (The role of the collection W for $m=1$ can be performed by the set of words $\{\varepsilon, b, b^2, \dots, b^{n-1}\}$.)

• *Comments*: Similar notions were introduced by Carpi and D'Alessandro in [40] and [42]. In [40] they called an n-state automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ *strongly transitive* if it is equipped with a set of n words $\{w_1, \dots, w_n\} \subset \Sigma^*$ such that for any two states $p, r \in Q$ there exists a unique word w_i for which $p \,.\, w_i = r$. If $L := \max\{|w_1|, \dots, |w_n|\}$, then (see [40], Theorem 2)

$$\mathrm{rt}(\mathscr{A}) \leqslant (n-2)(n+L-1)+1.$$

In [42] an n-state automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ was said to be *locally strongly transitive* if it is equipped with a set $\{w_1, \dots, w_k\} \subset \Sigma^*$ of k words and a set $\{q_1, \dots, q_k\} \subset Q$ of k distinct states such that

$$\{q \,.\, w_1, \dots, q \,.\, w_k\} = \{q_1, \dots, q_k\} \quad \text{for any } q \in Q.$$

If $L := \max\{|w_1|, \dots, |w_k|\}$ and $\ell := \min\{|w_1|, \dots, |w_k|\}$, then (see [42], Proposition 5)

$$\mathrm{rt}(\mathscr{A}) \leqslant (k-1)(n+L+1) - 2k \ln \frac{k+1}{2} + \ell.$$

3.6. Experimental results. The efficiency of the experimental study of automata has increased significantly in the last 20–25 years with the development of easily available and powerful computational devices. Now automata theory complies essentially with the 'method of science', where experiment is the original source of knowledge, and it is through understanding experimental results that one suggests new conjectures and searches for approaches to their proofs (or disproofs). This also applies to the range of questions concerned with Černý's conjecture. Extensive computer experiments [13], [58], [60], [103], [104], [176], [179] confirmed the conjecture for any DFA with at most 7 states and 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. To give a sense of the amount of necessary computations, recall that the number of n-state automata with m input letters is n^{nm}, as the specification of such an automaton is equivalent to a selection of m maps from the set of all n^n maps of an n-element set to itself. The function n^{nm} grows quite rapidly with n and m; for example, the number of 8-state automata with three input letters is $8^{24} = 2^{72} \approx 4.7223665 \cdot 10^{21}$. Some advanced tricks have been used to reduce the exhaustive search: a survey of this technique can be found in Szykuła's thesis [176], Chap. 6. In the case of 8-state automata with three input letters the number of automata to be studied was reduced to 20 933 723 139.

In addition to the proof of Černý's conjecture for automata of moderate size, brute force experiments revealed unexpected peculiarities in the distribution of possible reset thresholds of synchronizing automata with a fixed number of states. For instance, among the synchronizing DFAs with 6 states there are automata[23] with reset threshold 25, but no automata with reset threshold 24. The gap between

[23]Up to isomorphism and omitting inessential letters, there are two synchronizing 6-state automata with reset threshold 25: the Černý automaton $\mathscr{C}_6$ and the Kari automaton $\mathscr{K}_6$ (see Fig. 20). A letter is *inessential* if removing it from the input alphabet does not change the reset threshold.

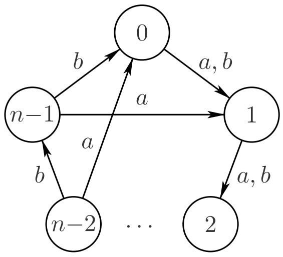


Figure 23. The automaton $\mathscr{D}_n$.

the maximum and the second largest value of the reset threshold of synchronizing automata with $n \geqslant 6$ was first observed by Trahtman [179] for $n \leqslant 10$ and hen reconfirmed by further experiments, which also covered $n = 11$ and $n = 12$. It was conjectured in [13], Conjecture 1, (d), that for $n > 6$ the second largest value of the reset threshold for synchronizing automata with n states is $n^2 - 3n + 4$ and that for $n > 7$ there exists a unique n-state DFA (up to isomorphism and omitting inessential letters) at which this value is attained, namely, the automaton $\mathscr{D}_n := \langle \{0, 1, 2, \dots, n-1\}, \{a, b\} \rangle$ with the action of input letters defined by

$$i \,.\, a := \begin{cases} i+1 & \text{if } i < n-2, \\ 0 & \text{if } i = n-2, \\ 1 & \text{if } i = n-1; \end{cases} \qquad i \,.\, b := i + 1 \pmod{n}.$$

A generic automaton in the series $\mathscr{D}_n$ is shown in Fig. 23.

The brute force experiments described in [13] showed that for $n > 8$ there is a second gap in the possible values of the reset thresholds for synchronizing automata with n states and 2 input letters. Namely, there exist no such automata with reset threshold less than $n^2 - 3n + 2$ and greater that $n^2 - 4n + 7$ if n is odd, or greater than $n^2 - 4n + 6$ if n is even. A third gap of similar type was registered for $n > 10$ in [104] and [176]. For a detailed study of the gap phenomenon, see [63].

The phenomenon of gaps exhibited by the values of reset thresholds is of independent interest; however, we dwell on it mainly because of the fact that its discovery made it possible to reveal interesting relations which had rest unobserved earlier. (It is the 'method of science' mentioned at the beginning of this subsection that had born fruit here.) In particular, it was observed in [13] that the experimentally observed behaviour of a number of synchronizing automata with a fixed number of states as a function of the reset threshold resembles strongly the behaviour of another quantity, thoroughly studied in discrete mathematics, namely, the number of primitive graphs with a fixed number of vertices as a function of the exponent[24].

[24]A strongly connected graph is *primitive* if the greatest common divisor of the lengths of all its cycles is equal to 1. A graph Γ has this property if and only if its adjacency matrix $M(\Gamma)$ is primitive in the sense of the (Perron–Frobenius) theory of non-negative matrices, that is, if $M(\Gamma)$ has a positive eigenvalue that is greater than the absolute values of all other eigenvalues. The *exponent* of $M(\Gamma)$ is the smallest integer d such that all entries of the matrix $M(\Gamma)^d$ are positive. The exponents of graphs have been studied thoroughly during the last 70 years, starting with Wielandt's classic paper [192]; a survey of the known facts in this area can be found in [37].

An analysis of the causes for this similarity has revealed a number of useful correspondences between synchronizing DFAs and primitive non-negative matrices; these relationships are discussed in detail in the second paper of our cycle.

Turning back to the discussion of experimental results concerned with Černý's conjecture, a mention should be made of the extensive experiments performed with randomly generated automata (see [52], [102], [103], and [169]). All these experiments operated with the simplest model of a random DFA with n states and m letters in which a DFA is represented by an m-tuple of maps chosen uniformly at random from all the n^n transformations of the state set. In the context of Černý's conjecture experiments with random DFAs demonstrated that, whenever a randomly generated DFA is synchronizing, its reset threshold is much smaller than Černý's bound. For instance, the maximum reset threshold observed in [102] amongst 1 000 000 synchronizing 100-state 2-letter random DFAs analyzed in that paper is equal to 41. Recall for comparison that the conjectural upper bound for the reset threshold of synchronizing automata with 100 states is $99^2 = 9801$, and the upper bound is $(100^3 - 100)/6 = 166\,650$. Thus, experiments suggest that even if Černý's conjecture does not hold in general, it holds for 'almost all' synchronizing automata. This experimental observation was theoretically justified in the important works of Nicaud [128] and Berlinkov and Szykuła [28], which we discuss in detail in the second paper of our cycle.

Conclusion

This paper is focused on the algorithmic and complexity-theoretic aspects of synchronization of finite deterministic automata and on Černý's conjecture. Other issues of the theory of synchronizing automata are addressed in the second paper of our cycle. It consists of seven sections, and the numbering goes through the whole cycle. In §§ 4 and 5 we discuss thoroughly two recent towering achievements in the theory of synchronizing automata: Trahtman's proof [182] of the conjecture put forward by Adler, Wayne Goodwyn, and Weiss [1] (and known as 'the road colouring problem') and Berlinkov's proof [26] of Cameron's conjecture [38] about the probability of synchronization of a random deterministic automaton. Section 6 is devoted to deep relationships between synchronizing DFAs and primitive non-negative matrices. In the next three sections we present the main results concerned with the synchronization of automata of other types: in § 7 we consider partial deterministic automata, in § 8 non-deterministic automata, and in § 9 other finite-state-machine models. The concluding section, § 10, contains a summary of the main currently open problems in the theory of synchronizing automata.

I am deeply grateful to an unknown referee for reading thoroughly the original version of this paper. Their constructive comments and valuable suggestions were taken into account in the final version of the text.

Bibliography

[1] R. L. Adler, L. W. Goodwyn, and B. Weiss, "Equivalence of topological Markov shifts", *Israel J. Math.* **27**:1 (1977), 49–63.

[2] P. Ageev, “Implementation of the algorithm for testing an automaton for synchronization in linear expected time”, *J. Autom. Lang. Comb.* **24**:2-4 (2019), 139–152.

[3] J. Almeida, S. Margolis, B. Steinberg, and M. Volkov, “Representation theory of finite semigroups, semigroup radicals and formal language theory”, *Trans. Amer. Math. Soc.* **361**:3 (2009), 1429–1461.

[4] J. Almeida and E. Rodaro, “Semisimple synchronizing automata and the Wedderburn–Artin theory”, *Internat. J. Found. Comput. Sci.* **27**:2 (2016), 127–145.

[5] J. Almeida and B. Steinberg, “Matrix mortality and the Černý–Pin conjecture”, *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 5583, Springer, Berlin 2009, pp. 67–80.

[6] J. Almeida and M. V. Volkov, “Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety”, *J. Algebra Appl.* **2**:2 (2003), 137–163.

[7] D. S. Ananichev, “The annulation threshold for partially monotonic automata”, *Izv. Vysh. Uchebn. Zaved. Mat.*, 2010, no. 1, 3–13; English transl. in *Russian Math.* (*Iz. VUZ*) **54**:1 (2010), 1–9.

[8] D. S. Ananichev and V. V. Gusev, “Approximation of reset thresholds with greedy algorithms”, *Fund. Inform.* **145**:3 (2016), 221–227.

[9] D. S. Ananichev, I. V. Petrov, and M. V. Volkov, “Collapsing words: a progress report”, *Internat. J. Found. Comput. Sci.* **17**:3 (2006), 507–518.

[10] D. S. Ananichev and M. V. Volkov, “Some results on Černý type problems for transformation semigroups”, *Semigroups and languages*, World Sci. Publ., River Edge, NJ 2004, pp. 23–42.

[11] D. S. Ananichev and M. V. Volkov, “Synchronizing monotonic automata”, *Theoret. Comput. Sci.* **327**:3 (2004), 225–239.

[12] D. S. Ananichev and M. V. Volkov, “Synchronizing generalized monotonic automata”, *Theoret. Comput. Sci.* **330**:1 (2005), 3–13.

[13] D. S. Ananichev, M. V. Volkov, and V. V. Gusev, “Primitive digraphs with large exponents and slowly synchronizing automata”, *Combinatorics and graph theory*. IV, Zap. Nauchn. Semin. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI), vol. 402, St. Petersburg Department of Steklov Mathematoical Institute, St. Petersburg 2012, pp. 9–39; English transl. in *J. Math. Sci.* (*N. Y.*) **192**:3 (2013), 263–278.

[14] D. Ananichev and V. Vorel, “A new lower bound for reset threshold of binary synchronizing automata with sink”, *J. Autom. Lang. Comb.* **24**:2-4 (2019), 153–164.

[15] J. Araújo, P. Cameron, and B. Steinberg, “Between primitive and 2-transitive: synchronization and its friends”, *EMS Surv. Math. Sci.* **4**:2 (2017), 101–184.

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

[17] W. R. Ashby, *An introduction to cybernetics*, Chapman and Hall, Ltd., London 1956, ix+295 pp.

[18] M.-P. Béal, M. V. Berlinkov, and D. Perrin, “A quadratic upper bound on the size of a synchronizing word in one-cluster automata”, *Internat. J. Found. Comput. Sci.* **22**:2 (2011), 277–288.

[19] M.-P. Béal and D. Perrin, “A quadratic upper bound on the size of a synchronizing word in one-cluster automata”, *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 5583, Springer, Berlin 2009, pp. 81–90.

[20] Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, and E. Shapiro, "DNA molecule provides a computing machine with both data and fuel", *Proc. Natl. Acad. Sci. USA* **100**:5 (2003), 2191–2196.

[21] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro, "Programmable and autonomous computing machine made of biomolecules", *Nature* **414**:6862 (2001), 430–434.

[22] M. V. Berlinkov, "On a conjecture by Carpi and D'Alessandro", *Internat. J. Found. Comput. Sci.* **22**:7 (2011), 1565–1576.

[23] M. V. Berlinkov, "Synchronizing quasi-Eulerian and quasi-one-cluster automata", *Internat. J. Found. Comput. Sci.* **24**:6 (2013), 729–745.

[24] M. V. Berlinkov, "Approximating the minimum length of synchronizing words is hard", *Theory Comput. Syst.* **54**:2 (2014), 211–223.

[25] M. V. Berlinkov, "On two algorithmic problems about synchronizing automata", *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 8633, Springer, Cham 2014, pp. 61–67.

[26] M. V. Berlinkov, "On the probability of being synchronizable", *Algorithms and discrete applied mathematics*, Lecture Notes in Comput. Sci., vol. 9602, Springer, Cham 2016, pp. 73–84.

[27] M. V. Berlinkov, R. Ferens, and M. Szykuła, "Preimage problems for deterministic finite automata", *J. Comput. System Sci.* **115** (2021), 214–234.

[28] M. V. Berlinkov and M. Szykuła, "Algebraic synchronization criterion and computing reset words", *Inform. Sci.* **369** (2016), 718–730.

[29] J. Berstel, D. Perrin, and C. Reutenauer, *Codes and automata*, Encyclopedia Math. Appl., vol. 129, Cambridge Univ. Press, Cambridge 2010, xiv+619 pp.

[30] M. T. Biskup, *Error resilience in compressed data – selected topics*, Ph.D. thesis, Inst. of Informatics, Univ. of Warsaw 2008, 136 pp., http://www.mimuw.edu.pl/sites/default/files/marek_biskup_mb-praca.pdf.

[31] M. T. Biskup, "Guaranteed synchronization of Huffman codes", *Data compression conference* (*DCC* 2008) (Snowbird, UT 2008), IEEE 2008, pp. 462–471.

[32] M. T. Biskup and W. Plandowski, "Guaranteed synchronization of Huffman codes with known position of decoder", *Data compression conference* (*DCC* 2009) (Snowbird, UT 2009), IEEE 2009, pp. 33–42.

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

[34] S. Bogdanović, B. Imreh, M. Ćirić, and T. Petković, "Directable automata and their generalizations: a survey", *Novi Sad J. Math.* **29**:2 (1999), 29–69.

[35] P. Bonizzoni and N. Jonoska, "Existence of constants in regular splicing languages", *Inform. and Comput.* **242** (2015), 340–353.

[36] V. Boppana, S. P. Rajan, K. Takayama, and M. Fujita, "Model checking based on sequential ATPG", *Computer aided verification*, Lecture Notes in Comput. Sci., vol. 1633, Springer-Verlag, Berlin 1999, pp. 418–430.

[37] R. A. Brualdi and H. J. Ryser, *Combinatorial matrix theory*, Encyclopedia Math. Appl., vol. 39, Cambridge Univ. Press, Cambridge 1991, x+367 pp.

[38] P. J. Cameron, "Dixon's theorem and random synchronization", *Discrete Math.* **313**:11 (2013), 1233–1236.

[39] R. M. Capocelli, L. Gargano, and U. Vaccaro, "On the characterization of statistically synchronizable variable-length codes", *IEEE Trans. Inform. Theory* **34**:4 (1988), 817–825.

[40] A. Carpi and F. D'Alessandro, "Strongly transitive automata and the Černý conjecture", *Acta Inform.* **46**:8 (2009), 591–607.

[41] A. Carpi and F. D'Alessandro, "The synchronization problem for locally strongly transitive automata", *Mathematical foundations of computer science* 2009, Lecture Notes in Comput. Sci., vol. 5734, Springer, Berlin 2009, pp. 211–222.

[42] A. Carpi and F. D'Alessandro, "Independent sets of words and the synchronization problem", *Adv. in Appl. Math.* **50**:3 (2013), 339–355.

[43] A. Carpi and F. D'Alessandro, "Locally strongly transitive automata in the Černý conjecture and related problems", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 165–184.

[44] J. Černý, "Poznámka k homogénnym eksperimentom s konečnými automatami", *Mat.-Fyz. Časopis Sloven. Akad. Vied.* **14**:3 (1964), 208–216; English transl., J. Černý, "A note on homogeneous experiments with finite automata", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 123–132.

[45] J. Černý, A. Pirická, and B. Rosenauerová, "On directable automata", *Kybernetika (Prague)* **7**:4 (1971), 289–298.

[46] Y.-B. Chen and D. J. Ierardi, "The complexity of oblivious plans for orienting and distinguishing polygonal parts", *Algorithmica* **14**:5 (1995), 367–397.

[47] A. Cherubini, "Synchronizing and collapsing words", *Milan J. Math.* **75**:1 (2007), 305–321.

[48] A. Cherubini, A. Frigeri, and Z. Liu, "Composing short 3-compressing words on a 2-letter alphabet", *Discrete Math. Theor. Comput. Sci.* **19**:1 (2017), 17, 35 pp.

[49] A. Cherubini and A. Kisielewicz, "Collapsing words, permutation conditions and coherent colorings of trees", *Theoret. Comput. Sci.* **410**:21-23 (2009), 2135–2147.

[50] A. Cherubini and A. Kisielewicz, "Recognizing 3-collapsing words over a binary alphabet", *Theoret. Comput. Sci.* **629** (2016), 64–79.

[51] A. Cherubini, A. Kisielewicz, and B. Piochi, "On the length of shortest 2-collapsing words", *Discrete Math. Theor. Comput. Sci.* **11**:1 (2009), 33–44.

[52] K. Chmiel and A. Roman, "COMPAS – a computing package for synchronization", *Implementation and application of automata*, Lecture Notes in Comput. Sci., vol. 6482, Springer, Berlin 2011, pp. 79–86.

[53] Hyunwoo Cho, S.-W. Jeong, F. Somenzi, and C. Pixley, "Multiple observation time single reference test generation using synchronizing sequences", 1993 *European conference on design automation with the European event in ASIC design* (Paris 1993), IEEE 1993, pp. 494–498.

[54] Hyunwoo Cho, S.-W. Jeong, F. Somenzi, and C. Pixley, "Synchronizing sequences and symbolic traversal techniques in test generation", *J. Electronic Testing* **4** (1993), 19–31.

[55] Th. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed., MIT Press, Cambridge, MA 2009, xx+1292 pp.

[56] Z. H. Cui, Y. He, and S. Y. Sun, "Synchronizing bounded partially ordered automata", (Chinese), *Chinese J. Comput.* **42**:3 (2019), 610–623.

[57] M. de Bondt, *A short proof of a theorem of J.-E. Pin*, 2018, 12 pp., arXiv: 1811.11660.

[58] M. de Bondt, H. Don, and H. Zantema, "DFAs and PFAs with long shortest synchronizing word length", *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 10396, Springer, Cham 2017, pp. 122–133.

[59] F. M. Dekking, "The spectrum of dynamical systems arising from substitutions of constant length", *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete* **41**:3 (1978), 221–239.

[60] H. Don and H. Zantema, "Finding DFAs with maximal shortest synchronizing word length", *Language and automata theory and applications*, Lecture Notes in Comput. Sci., vol. 10168, Springer, Cham 2017, pp. 249–260.

[61] H. Don and H. Zantema, "Counting symbol switches in synchronizing automata", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 253–286.

[62] L. Dubuc, "Sur les automates circulaires et la conjecture de Černý", *RAIRO Inform. Théor. Appl.* **32**:1-3 (1998), 21–34.

[63] M. Dżyga, R. Ferens, V. V. Gusev, and M. Szykuła, "Attainable values of reset thresholds", *42nd international symposium on mathematical foundations of computer science*, LIPIcs. Leibniz Int. Proc. Inform., vol. 83, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern 2017, 40, 14 pp.

[64] D. Eppstein, "Reset sequences for monotonic automata", *SIAM J. Comput.* **19**:3 (1990), 500–510.

[65] D. Eppstein and J.-C. Falmagne, "Algorithms for media", *Discrete Appl. Math.* **156**:8 (2008), 1308–1320.

[66] H. Fernau, V. V. Gusev, S. Hoffmann, M. Holzer, M. V. Volkov, and P. Wolf, "Computational complexity of synchronization under regular constraints", *44th international symposium on mathematical foundations of computer science*, LIPIcs. Leibniz Int. Proc. Inform., vol. 138, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern 2019, 63, 14 pp.

[67] H. Fernau, P. Heggernes, and Y. Villanger, "A multi-parameter analysis of hard problems on deterministic finite automata", *J. Comput. System Sci.* **81**:4 (2015), 747–765.

[68] H. Fernau and A. Krebs, "Problems on finite automata and the exponential time hypothesis", *Algorithms (Basel)* **10**:1 (2017), 24, 25 pp.

[69] M. A. Fischler and M. Tannenbaum, "Synchronizing and representation problems for sequential machines with masked outputs", *11th annual symposium on switching and automata theory (SWAT* 1970), IEEE 1970, pp. 97–103.

[70] P. Frankl, "An extremal problem for two families of sets", *European J. Combin.* **3**:2 (1982), 125–127.

[71] D. Frettlöh and B. Sing, "Computing modular coincidences for substitution tilings and point sets", *Discrete Comput. Geom.* **37**:3 (2007), 381–407.

[72] A. Frigeri and E. Rodaro, "Missing factors of ideals and synchronizing automata", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 309–320.

[73] P. Gawrychowski, *Complexity of shortest synchronizing word*, preprint, 2008.

[74] P. Gawrychowski and D. Straszak, "Strong inapproximability of the shortest reset word", *Mathematical foundations of computer science* 2015, Part I, Lecture Notes in Comput. Sci., vol. 9234, Springer, Heidelberg 2015, pp. 243–255.

[75] M. Gerbush and B. Heeringa, "Approximating minimum reset sequences", *Implementation and application of automata*, Lecture Notes in Comput. Sci., vol. 6482, Springer, Berlin 2011, pp. 154–162.

[76] A. Gill, "State-identification experiments in finite automata", *Information and Control* **4**:2-3 (1961), 132–154.

[77] S. Ginsburg, "On the length of the smallest uniform experiment which distinguishes the terminal states of a machine", *J. Assoc. Comput. Mach.* **5**:3 (1958), 266–280.

[78] S. Ginsburg, *An introduction to mathematical machine theory*, Addison-Wesley Publishing Co., Inc., Reading, MA–Palo Alto, CA–London 1962, ix+147 pp.

[79] K. Y. Goldberg, "Orienting polygonal parts without sensors", *Algorithmica* **10**:2-4 (1993), 201–225.

[80] F. Gonze, V. V. Gusev, R. M. Jungers, B. Gerencsér, and M. V. Volkov, "On the interplay between Černý and Babai's conjectures", *Internat. J. Found. Comput. Sci.* **30**:1 (2019), 93–114.

[81] F. Gonze, R. M. Jungers, and A. N. Trahtman, “A note on a recent attempt to improve the Pin–Frankl bound”, *Discrete Math. Theor. Comput. Sci.* **17**:1 (2015), 307–308.

[82] P. Goralčík and V. Koubek, “Rank problems for composite transformations”, *Internat. J. Algebra Comput.* **5**:3 (1995), 309–316.

[83] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete mathematics. A foundation for computer science*, 2nd ed., Addison-Wesley Publ. Co., Reading, MA 2006, xiv+657 pp.

[84] M. Grech and A. Kisielewicz, “The Černý conjecture for automata respecting intervals of a directed graph”, *Discrete Math. Theor. Comput. Sci.* **15**:3 (2013), 61–72.

[85] M. Grech and A. Kisielewicz, “Synchronizing sequences for road colored digraphs”, *Discrete Appl. Math.* **285** (2020), 128–140.

[86] V. V. Gusev, “Lower bounds for the length of reset words in Eulerian automata”, *Reachability problems*, Lecture Notes in Comput. Sci., vol. 6945, Springer, Berlin 2011, pp. 180–190.

[87] V. V. Gusev, R. M. Jungers, and D. Průša, “Dynamics of the independence number and automata synchronization”, *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 11088, Springer, Cham 2018, pp. 379–391.

[88] V. V. Gusev, M. I. Maslennikova, and E. V. Pribavkina, “Principal ideal languages and synchronizing automata”, *Fund. Inform.* **132**:1 (2014), 95–108.

[89] V. V. Gusev and E. V. Pribavkina, “On codeword lengths guaranteeing synchronization”, *Combinatorics on words*, Lecture Notes in Comput. Sci., vol. 11682, Springer, Cham 2019, pp. 207–216.

[90] J. Håstad, “Clique is hard to approximate within $n^{1-\varepsilon}$”, *Acta Math.* **182**:1 (1999), 105–142.

[91] S. Hoffmann, “On a class of constrained synchronization problems in NP”, *Proceedings of the 21st Italian conference on theoretical computer science* (Ischia 2020), CEUR Workshop Proceedings, vol. 2756, 2020, pp. 145–157, http://ceur-ws.org/Vol-2756/.

[92] S. Hoffmann, “Completely reachable automata, primitive groups and the state complexity of the set of synchronizing words”, *Language and automata theory and applications*, Lecture Notes in Comput. Sci., vol. 12638, Springer, Cham 2021, pp. 305–317.

[93] S. Hoffmann, “State complexity of the set of synchronizing words for circular automata and automata over binary alphabets”, *Language and automata theory and applications*, Lecture Notes in Comput. Sci., vol. 12638, Springer, Cham 2021, pp. 318–330.

[94] S. Hoffmann, “Constrained synchronization and subset synchronization problems for weakly acyclic automata”, *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 12811, Springer, Cham 2021, pp. 204–216.

[95] S. Hoffmann, “Computational complexity of synchronization under sparse regular constraints”, *Fundamentals of computation theory*, Lecture Notes in Comput. Sci., vol. 12867, Springer, Cham 2021, pp. 272–286.

[96] S. Hoffmann, “Ideal separation and general theorems for constrained synchronization and their application to small constraint automata”, *Computing and combinatorics*, Lecture Notes in Comput. Sci., vol. 13025, Springer, Cham 2021, pp. 176–188.

[97] S. Hoffmann, “Constrained synchronization and commutativity”, *Theoret. Comput. Sci.* **890** (2021), 147–170.

[98] H. Jürgensen, “Synchronization”, *Inform. and Comput.* **206**:9-10 (2008), 1033–1044.

[99] J. Kari, “A counter example to a conjecture concerning synchronizing words in finite automata”, *Bull. Eur. Assoc. Theor. Comput. Sci.* **73** (2001), 146.

[100] J. Kari, “Synchronizing finite automata on Eulerian digraphs”, *Theoret. Comput. Sci.* **295**:1-3 (2003), 223–232.

[101] J. Kari and M. Volkov, “Černý’s conjecture and the road colouring problem”, *Handbook of automata theory*, vol. I: *Theoretical foundations*, Ch. 15, EMS Press, Berlin 2021, pp. 525–565.

[102] A. Kisielewicz, J. Kowalski, and M. Szykuła, “Computing the shortest reset words of synchronizing automata”, *J. Comb. Optim.* **29**:1 (2015), 88–124.

[103] A. Kisielewicz, J. Kowalski, and M. Szykuła, “Experiments with synchronizing automata”, *Implementation and application of automata*, Lecture Notes in Comput. Sci., vol. 9705, Springer, Cham 2016, pp. 176–188.

[104] A. Kisielewicz and M. Szykuła, “Generating small automata and the Černý conjecture”, *Implementation and application of automata*, Lecture Notes in Comput. Sci., vol. 7982, Springer, Heidelberg 2013, pp. 340–348.

[105] A. Kisielewicz and M. Szykuła, “Synchronizing automata with extremal properties”, *Mathematical foundations of computer science* 2015, Part 1, Lecture Notes in Comput. Sci., vol. 9234, Springer, Heidelberg 2015, pp. 331–343.

[106] S. C. Kleene, “Representation of events in nerve nets and finite automata”, *Automata studies*, Ann. of Math. Stud., vol. 34, Princeton Univ. Press, Princeton, NJ 1956, pp. 3–41.

[107] A. A. Klyachko, I. K. Rystsov, and M. A. Spivak, “An extremal combinatorial problem associated with the bound of the length of a synchronizing word in an automaton”, *Kibernetika* **25**:2 (1987), 16–20; English transl. in *Cybernetics* **23**:2 (1987), 165–171.

[108] Z. Kohavi and J. Winograd, “Bounds on the length of synchronizing sequences and the order of information losslessness”, *Theory of machines and computations*, Academic Press, New York 1971, pp. 197–206.

[109] Z. Kohavi and J. Winograd, “Establishing certain bounds concerning finite automata”, *J. Comput. System Sci.* **7**:3 (1973), 288–299.

[110] D. Kozen, “Lower bounds for natural proof systems”, *18th annual symposium on foundations of computer science* (Providence, RI 1977), IEEE Comput. Sci., Long Beach, CA 1977, pp. 254–266.

[111] M. W. Krentel, “The complexity of optimization problems”, *J. Comput. System Sci.* **36**:3 (1988), 490–509.

[112] A. E. Laemmel, *A general class of discrete codes and certain of their properties*, Res. rep. R-459-55, PIB-389, Microwave Research Inst., Polytechnic Inst., Brooklyn, NY 1956.

[113] A. E. Laemmel, *Study on application of coding theory*, Tech. rep. PIBMRI-895.5-63, Dept. Electrophysics, Microwave Research Inst., Polytechnic Inst., Brooklyn, NY 1963.

[114] A. E. Laemmel and B. Rudner, *Study of the application of coding theory*, Tech. rep. PIBEP-69-034, Dept. Electrophysics, Polytechnic Inst., Brooklyn, Farmingdale, NY 1969.

[115] V. I. Levenshteĭn, “Self-adaptive automata for decoding messages”, *Dokl. Akad. Nauk SSSR* **141**:6 (1961), 1320–1323; English transl. in *Soviet Physics Dokl.* **6** (1961), 1042–1045.

[116] C. L. Liu, “Determination of the final state of an automaton whose initial state is unknown”, *IEEE Trans. Electronic Computers* **EC-12**:6 (1963), 918–920.

[117] C. L. Liu, *Some memory aspects of finite automata*, Tech. rep. 411, Research Lab. Electronics, Massachusetts Inst., MIT Res. Lab. Electronics, Cambridge, MA 1963, 71 pp., http://hdl.handle.net/1721.1/4414.

[118] P. V. Martugin, "A series of slowly synchronizing automata with a zero state over a small alphabet", *Inform. and Comput.* **206**:9-10 (2008), 1197–1203.

[119] M. Maslennikova, "Reset complexity of ideal languages over a binary alphabet", *Internat. J. Found. Comput. Sci.* **30**:6-7 (2019), 1177–1196.

[120] M. Maslennikova and E. Rodaro, "Representation of (left) ideal regular languages by synchronizing automata", *Computer science – theory and applications*, Lecture Notes in Comput. Sci., vol. 9139, Springer, Cham 2015, pp. 325–338.

[121] M. Maslennikova and E. Rodaro, "Trim strongly connected synchronizing automata and ideal languages", *Fund. Inform.* **162**:2-3 (2018), 183–203.

[122] A. Mateescu and A. Salomaa, "Many-valued truth functions, Černý's conjecture, and road coloring", *Current trends in theoretical computer science. Entering the 21th century*, World Sci. Publ., River Edge, NJ 2001, pp. 693–707.

[123] Yu. T. Medvedev, "On the class of events representable in a finite automaton", *Automata*, Inostrannaya Literatura, Moscow 1956, pp. 385–401; English transl. in E.F. Moore (ed.), *Sequential machines — Selected papers*, Addison Wesley, Boston, MA 1964.

[124] E. F. Moore, "Gedanken-experiments on sequential machines", *Automata studies*, Ann. of Math. Stud., vol. 34, Princeton Univ. Press, Princeton, NJ 1956, pp. 129–153.

[125] E. H. Moore, "On certain crinkly curves", *Trans. Amer. Math. Soc.* **1**:1 (1900), 72–90; "Errata", *Trans. Amer. Math. Soc.* **1** (1900), 507.

[126] B. K. Natarajan, "An algorithmic approach to the automated design of parts orienters", *27th annual symposium on foundations of computer science* (*SFCS* 1986) (Toronto, ON 1986), IEEE 1986, pp. 132–142.

[127] B. K. Natarajan, "Some paradigms for the automated design of parts feeders", *Int. J. Robot. Res.* **8**:6 (1989), 98–109.

[128] C. Nicaud, "The Černý conjecture holds with high probability", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 343–365.

[129] J. Olschewski and M. Ummels, "The complexity of finding reset words in finite automata", *Mathematical foundations of computer science* 2010, Lecture Notes in Comput. Sci., vol. 6281, Springer, Berlin 2010, pp. 568–579.

[130] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Co., Reading, MA 1994, xvi+523 pp.

[131] C. H. Papadimitriou and M. Yannakakis, "The complexity of facets (and some facets of complexity)", *J. Comput. System Sci.* **28**:2 (1984), 244–259.

[132] J.-E. Pin, *Le problème de la synchronisation et la conjecture de Černý*, Thèse de 3ème cycle, Univ. Paris VI 1978.

[133] J.-E. Pin, "Sur un cas particulier de la conjecture de Cerny", *Automata, languages and programming* (Udine 1978), Lecture Notes in Comput. Sci., vol. 62, Springer, Berlin–New York 1978, pp. 345–352.

[134] J.-E. Pin, "Utilisation de l'algèbre linéaire en théorie des automates", *Actes du 1er colloque AFCET-SMF de mathématiques appliquées* (Palaiseau 1978), AFCET 1978, pp. 85–92, https://hal.archives-ouvertes.fr/hal-00340773.

[135] J. E. Pin, "On two combinatorial problems arising from automata theory", *Combinatorial mathematics* (Marseille–Luminy 1981), North-Holland Math. Stud., vol. 75, Ann. Discrete Math., **17**, North-Holland, Amsterdam 1983, pp. 535–548.

[136] C. Pixley, S.-W. Jeong, and G. D. Hachtel, “Exact calculation of synchronizing sequences based on binary decision diagrams”, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **13**:8 (1994), 1024–1034.

[137] R. Pöschel, M. V. Sapir, N. V. Sauer, M. G. Stone, and M. V. Volkov, “Identities in full transformation semigroups”, *Algebra Universalis* **31**:4 (1994), 580–588.

[138] E. V. Pribavkina, “Slowly synchronizing automata with zero and noncomplete sets”, *Mat. Zametki* **90**:3 (2011), 422–430; English transl. in *Math. Notes* **90**:3 (2011), 411–417.

[139] E. V. Pribavkina and E. Rodaro, “Synchronizing automata with finitely many minimal synchronizing words”, *Inform. and Comput.* **209**:3 (2011), 568–579.

[140] N. Pytheas Fogg, *Substitutions in dynamics, arithmetics and combinatorics*, Lecture Notes in Math., vol. 1794 (V. Berthé, S. Ferenczi, C. Mauduit, A. Siegel, eds.), Springer-Verlag, Berlin 2002.

[141] M. O. Rabin and D. Scott, “Finite automata and their decision problems”, *IBM J. Res. Develop.* **3**:2 (1959), 114–125.

[142] J. L. Ramírez Alfonsín, *The Diophantine Frobenius problem*, Oxford Lecture Ser. Math. Appl., vol. 30, Oxford Univ. Press, Oxford 2005, xvi+243 pp.

[143] A. S. Rao and K. Y. Goldberg, “Manipulating algebraic parts in the plane”, *IEEE Trans. Robotics Autom.* **11**:4 (1995), 598–602.

[144] R. Reis and E. Rodaro, “Ideal regular languages and strongly connected synchronizing automata”, *Theoret. Comput. Sci.* **653** (2016), 97–107.

[145] J.-K. Rho, F. Somenzi, and C. Pixley, “Minimum length synchronizing sequences of finite state machine”, *30th ACM/IEEE design automation conference* (Dallas, TX 1993), ACM, New York, NY 1993, pp. 463–468.

[146] E. Rodaro, “Strongly connected synchronizing automata and the language of minimal reset words”, *Adv. in Appl. Math.* **99** (2018), 158–173.

[147] E. Rodaro, “A bound for the length of the shortest reset words for semisimple synchronizing automata via the packing number”, *J. Algebraic Combin.* **50**:3 (2019), 237–253.

[148] A. Roman and M. Szykuła, “Forward and backward synchronizing algorithms”, *Expert Syst. Appl.* **42**:24 (2015), 9512–9527.

[149] I. K. Rystsov, “An estimate of the length of a nuclear word for a finite automaton”, *Automata*, vol. 2, Saratov State University, Saratov 1977, pp. 43–48.

[150] I. K. Rystsov, “Minimization of the length of synchronizing words for finite automata”, *Theoretical questions of design of computer systems*, Akad. Nauk Ukr.SSR, Cybernetics Institute, Kiev 1980, pp. 75–82. (Russian)

[151] I. K. Rystsov, “Polynomial complete problems in automata theory”, *Inform. Process. Lett.* **16**:3 (1983), 147–151. (Russian)

[152] I. K. Rystsov, “Rank of a finite automaton”, *Kibernetika i Sistemnyi Anal.* **28**:3 (1992), 3–10; English transl. in *Cybernet. Systems Anal.* **28**:3 (1992), 323–328.

[153] I. K. Rystsov, “Resetting words for decidable automata”, *Kibernetika i Sistemnyi Anal.* **30**:6 (1994), 21–26; English transl, in *Cybernet. Systems Anal.* **30**:6 (1992), 807–811.

[154] I. K. Rystsov, “Almost optimal bound of reccurent word length for regular automata”, *Kiberenetika Sistemnyi Anal.* **31**:5 (1995), 40–48; English transl. in *Cybernet. Systems Anal.* **31**:5 (1995), 669–674.

[155] I. K. Rystsov, “Quasioptimal bound for the length of reset words for regular automata”, *Acta Cybernet.* **12**:2 (1995), 145–152.

[156] I. Rystsov, “Exact linear bound for the length of reset words in commutative automata”, *Publ. Math. Debrecen* **48**:3-4, suppl. (1996), 405–409.

[157] I. Rystsov, “Reset words for commutative and solvable automata”, *Theoret. Comput. Sci.* **172**:1-2 (1997), 273–279.

[158] I. K. Rystsov, “Estimation of the length of reset words for automata with simple idempotents”, *Kibernetika Sistemnyi Anal.* **36**:3 (2000), 32–39; English transl. in *Cybernet. Systems Anal.* **36**:3 (2000), 339–344.

[159] I. K. Rystsov, “Cerny’s conjecture for automata with simple idempotents”, *Kibernet. Sistem. Anal.* **58**:1 (2022), 3–10; English transl, in *Cybernet. Systems Anal.* **58**:1 (2022), 1–7.

[160] A. Ryzhikov, “Synchronization problems in automata without non-trivial cycles”, *Theoret. Comput. Sci.* **787** (2019), 77–88.

[161] A. Ryzhikov and M. Szykula, “Finding short synchronizing words for prefix codes”, 43*rd international symposium on mathematical foundations of computer science* (Liverpool, UK 2018), LIPIcs. Leibniz Int. Proc. Inform., vol. 117, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern 2018, 21, 14 pp.

[162] A. Salomaa, “Compositions over a finite domain: from completeness to synchronizable automata”, *A half-century of automata theory. Celebration and inspiration* (London, ON 2000), World Sci. Publ., River Edge, NJ 2001, pp. 131–143.

[163] A. Salomaa, “Generation of constants and synchronization of finite automata”, *J. UCS* **8**:2 (2002), 332–347.

[164] A. Salomaa, “Synchronization of finite automata: contributions to an old problem”, *The essence of computation. Complexity, analysis, transformation*, Essays dedicated to N. D. Jones [on occasion of his 60th birthday], Lecture Notes in Comput. Sci., vol. 2566, Springer, Berlin 2002, pp. 37–59.

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

[166] S. Sandberg, “Homing and synchronizing sequences”, *Model-based testing of reactive systems. Advanced lectures*, Lecture Notes in Comput. Sci., vol. 3472, Springer-Verlag, Berlin 2005, pp. 5–33.

[167] M. Schützenberger, “On an application of semi groups methods to some problems in coding”, *IRE Trans. Inform. Theory* **2**:3 (1956), 47–60.

[168] Y. Shitov, “An improvement to a recent upper bound for synchronizing words of finite automata”, *J. Autom. Lang. Comb.* **24**:2-4 (2019), 367–373.

[169] E. Skvortsov and E. Tipikin, “Experimental study of the shortest reset word of random automata”, *Implementation and application of automata*, Lecture Notes in Comput. Sci., vol. 6807, Springer, Heidelberg 2011, pp. 290–298.

[170] P. H. Starke, “Eine Bemerkung über homogene Experimente”, *Elektron. Informationsverarb. Kybernet.* **2** (1966), 257–259; English transl., P. H. Starke, “A remark about homogeneous experiments”, *J. Autom. Lang. Comb.* **24**:2-4 (2019), 133–137.

[171] P. H. Starke, *Abstrakte Automaten*, VEB Deutscher Verlag der Wissenschaften, Berlin 1969, 392 pp.; English transl., P. H. Starke, *Abstract automata*, North-Holland Publishing Co., Amsterdam–London; American Elsevier Publishing Co., Inc., New York 1972, 419 pp.

[172] B. Steinberg, “Černý’s conjecture and group representation theory”, *J. Algebraic Combin.* **31** (2010), 83–109.

[173] B. Steinberg, “The averaging trick and the Černý conjecture”, *Internat. J. Found. Comput. Sci.* **22**:7 (2011), 1697–1706.

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

[175] M. Suomalainen, A. Q. Nilles, and S. M. LaValle, "Virtual reality for robots", 2020 *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (Las Vegas, NV 2020), IEEE 2020, pp. 11458–11465.

[176] M. Szykuła, *Algorithms for synchronizing automata*, Ph.D. thesis, Inst. of Computer Science, Univ. of Wrocław 2014.

[177] M. Szykuła, "Improving the upper bound on the length of the shortest reset word", *35th symposium on theoretical aspects of computer science*, LIPIcs. Leibniz Int. Proc. Inform., vol. 96, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern 2018, 56, 13 pp.

[178] M. Szykuła and V. Vorel, "An extremal series of Eulerian synchronizing automata", *Developments in language theory*, Lecture Notes in Comput. Sci., vol. 9840, Springer, Berlin 2016, pp. 380–392.

[179] A. N. Trahtman, "An efficient algorithm finds noticeable trends and examples concerning the Černý conjecture", *Mathematical foundations of computer science* 2006, Lecture Notes in Comput. Sci., vol. 4162, Springer, Berlin 2006, pp. 789–800.

[180] A. N. Trahtman, "The Černý conjecture for aperiodic automata", *Discrete Math. Theor. Comput. Sci.* **9**:2 (2007), 3–10.

[181] A. Trahtman, "Some aspects of synchronization of DFA", *J. Comput. Sci. Tech.* **23**:5 (2008), 719–727.

[182] A. N. Trahtman, "The road coloring problem", *Israel J. Math.* **172**:1 (2009), 51–60.

[183] A. N. Trahtman, "Modifying the upper bound on the length of minimal synchronizing word", *Fundamentals of computation theory*, Lecture Notes in Comput. Sci., vol. 6914, Springer, Heidelberg 2011, pp. 173–180.

[184] A. N. Trahtman, *The Černy conjecture*, 2022 (v1 – 2012), 14 pp., arXiv: 1202.4626.

[185] A. N. Trahtman, *The length of a minimal synchronizing word and the Černy conjecture*, 2021 (v1 – 2014), 14 pp., arXiv: 1405.2435.

[186] A. N. Trahtman, *Cerny–Starke conjecture from the sixties of* XX *century*, 2021 (v1 – 2020), 18 pp., arXiv: 2003.06177.

[187] M. V. Volkov, "Synchronizing automata and the Černý conjecture", *Language and automata theory and applications*, Lecture Notes in Comput. Sci., vol. 5196, Springer, Berlin 2008, pp. 11–27.

[188] M. V. Volkov, "Synchronizing automata preserving a chain of partial orders", *Theoret. Comput. Sci.* **410**:37 (2009), 3513–3519.

[189] M. V. Volkov, "Slowly synchronizing automata with idempotent letters of low rank", *J. Autom. Lang. Comb.* **24**:2-4 (2019), 375–386.

[190] V. Vorel, *Synchronization and discontinuous input processing in transition systems*, Doctoral thesis, Charles Univ., Prague 2018, 137 pp., https://dspace.cuni.cz/handle/20.500.11956/104303.

[191] V. Vorel and A. Roman, "Parameterized complexity of synchronization and road coloring", *Discrete Math. Theor. Comput. Sci.* **17**:1 (2015), 283–305.

[192] H. Wielandt, "Unzerlegbare, nicht negative Matrizen", *Math. Z.* **52** (1950), 642–648.

[193] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number", *Theory Comput.* **3** (2007), 103–128, 6.

Mikhail V. Volkov
Ural Federal University
E-mail: `m.v.volkov@urfu.ru`

Received 31/AUG/20
Translated by A. PANKRAT'EV