Černý's conjecture and the road coloring problem

Jarkko Kari[1], *Mikhail Volkov*[2]

[1]Department of Mathematics
FI-20014 University of Turku
Turku, Finland

[2]Department of Mathematics and Mechanics
620083 Ural State University
Ekaterinburg, Russia
email: jkari@utu.fi, Mikhail.Volkov@usu.ru

March 6, 2011 21 h 36

chapterKV

2010 Mathematics Subject Classification: 68Q45 68R10

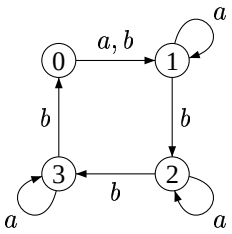
Key words: Finite automata, Synchronizing automata, Reset words, Černý's conjecture, Road Coloring Problem

Contents

1 Synchronizing automata, their origins and importance

c:motivation

A complete deterministic finite automaton (DFA) $\mathcal{A} = (Q, A)$ (here and below Q stands for the state set and A for the input alphabet) is called *synchronizing* if there exists a word $w \in A^*$ whose action resets $\mathcal{A}$, that is, w leaves the automaton in one particular state no matter at which state in Q it is applied: $q \cdot w = q' \cdot w$ for all $q, q' \in Q$. Any word w with this property is said to be a *reset* word for the automaton.



KV:fig:C4

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

KV:fig:C4

Figure 1 shows a synchronizing automaton with 4 states denoted by $\mathcal{C}_4$. The reader can easily verify that the word ab^3ab^3a resets the automaton leaving it in the state 1. With somewhat more effort one can also check that ab^3ab^3a is the shortest reset word for $\mathcal{C}_4$. The example in Figure 1 is due to KV:fig:C4 Černý, a Slovak computer scientist, in whose pioneering paper [13] the notion of a synchronizing automaton explicitly appeared for the first time. (Černý called such automata *directable*. The word *synchronizing* in this context was probably introduced by Hennie [31].) Implicitly, however, this concept has been around since the earliest days of automata theory. The very first synchronizing automaton that we were able to trace back in the literature appeared in Ashby's classic book [3, pp. 60–61], see [67, Section 1] for a discussion.

Needs double-checking!!

In [13] the notion of a synchronizing automaton arose within the classic framework of Moore's "Gedanken-experiments" [39]. For Moore and his followers finite automata served as a mathematical model of devices working in discrete mode, such as computers or relay control systems. This leads to the following natural problem: how can we restore control over such a device if we do not know its current state but can observe outputs produced by the device under various actions? Moore [39] has shown 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, that is, each next action was selected on the basis of the outputs caused by the previous actions. Ginsburg [28] considered more restricted experiments that he called *uniform*. A uniform experiment[1] is just a fixed sequence of actions, that is, a word over the input alphabet; thus, in Ginsburg's experiments outputs were only used for calculating the resulting state at the end of an experiment. From this, just one further step was needed to come to the setting in which outputs were not used at all. It should be noted that this setting is by no means artificial—there exist many practical situations when it is technically impossible to observe output signals. (Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon.)

The original "Gedanken-experiments" motivation for studying synchronizing automata is still of importance, and reset words are frequently applied in model-based testing of reactive systems. See [16, 10] as typical samples of technical contributions to the area and [61] for a recent survey.

Another strong motivation comes from the coding theory. We refer to [8, Chapters 3 and 10] for a detailed account of profound connections between codes and automata; here

[1] After [27], the name *homing sequence* has become standard for the notion.

we restrict ourselves to a special (but still very important) case of maximal prefix codes. Recall that a *prefix code* over a finite alphabet A is a set X of words in A^* such that no word of X is a prefix of another word of X. A prefix code is *maximal* if it is not contained in another prefix code over the same alphabet. A maximal prefix code X over A is *synchronized* if there is a word $x \in X^*$ such that for any word $w \in A^*$, one has $wx \in X^*$. Such a word x is called a *synchronizing word* for X. The advantage of synchronized codes is that they are able to recover after a loss of synchronization between the decoder and the coder caused by channel errors: in the case of such a loss, it suffices to transmit a synchronizing word and the following symbols will be decoded correctly. Moreover, since the probability that a word $v \in A^*$ contains a fixed factor x tends to 1 as the length of v increases, synchronized codes eventually resynchronize by themselves, after sufficiently many symbols being sent. (As shown in [11], the latter property in fact characterizes synchronized codes.) The following simple example illustrates these ideas: let $A = \{0, 1\}$ and $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X. For instance, if the code word 000 has been sent but, due to a channel error, the word 100 has been received, the decoder interprets 10 as a code word, and thus, loses synchronization. However, with a high probability this synchronization loss only propagates for a short while; in particular, the decoder definitely resynchronizes as soon as it encounters one of the segments 010, 011110, 011111110, ... in the received stream of symbols. A few samples of such streams are shown in Figure 2 in which vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

fig:decoding

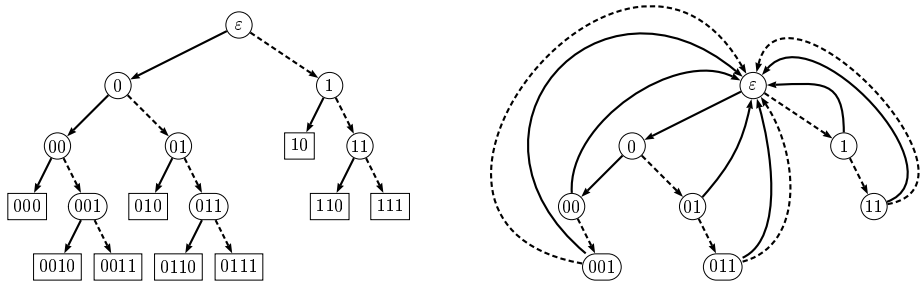
Sent	0 0 0 | 0 0 1 0 | **0 1 1 1** | ...
Received	1 0 | 0 0 0 | 1 0 | **0 1 1 1** | ...
Sent	0 0 0 | 0 1 1 1 | 1 1 0 | 0 0 1 1 | 0 0 0 | 1 0 | **1 1 0** | ...
Received	1 0 | 0 0 1 1 | 1 1 1 | 0 0 0 | 1 1 0 | 0 0 1 0 | **1 1 0** | ...
Sent	0 0 0 | 0 0 0 | 1 1 1 | **1 0** | ...
Received	1 0 | 0 0 0 | 0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

Figure 2. Restoring synchronization

If X is a finite prefix code over an alphabet A, then its decoding can be implemented by a deterministic automaton that is defined as follows. Let Q be the set of all proper prefixes of the words in X (including the empty word ε). For $q \in Q$ and $a \in A$, define

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

The resulting automaton $\mathcal{A}_X$ is complete whenever the code X is maximal and it is easy to see that $\mathcal{A}_X$ is a synchronizing automaton if and only if X is a synchronized code. Moreover, a word x is synchronizing for X if and only if x is a reset word for $\mathcal{A}_X$ and sends all states in Q to the state ε. Figure 3 illustrates this construction for the code $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$ considered above. The solid/dashed lines correspond to (the action of) 0/1.



:fig:huffman

Figure 3. A synchronized code (on the left) and its automaton (on the right)

Thus, **(to be continued and supplied by some historical references).**

An additional source of problems related to synchronizing automata has come from *robotics* or, more precisely, from part handling problems in industrial automation such as part feeding, fixturing, loading, assembly and packing. Within this framework, the concept of a synchronizing automaton was again rediscovered in the mid-1980s by Natarajan [40, 41] who showed how synchronizing automata can be used to design sensor-free orienters for polygonal parts, see [67, Section 1] for a transparent example illustrating Natarajan's approach in a nutshell. Since the 1990s synchronizing automata usage in the area of robotic manipulation has grown into a prolific research direction but it is fair to say that publications in this area deal mostly with implementation technicalities. However, amongst them there are papers of significant theoretical importance such as [20, 29, 15].

To be checked with Jarrko!!

Recently, it has been realized that a notion that arose in studying of *substitution systems* is also closely related to synchronizing automata. A *substitution* on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of *constant length* if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the *coincidence condition* if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculating the iterations of τ up to τ^4 (see Figure 4), we observe that τ satisfies the coincidence condition (with $k = 4$, $m = 7$).

kv:fig:substituion

$$\begin{array}{lllllllll} 0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 12202011\mathbf{1}12202011 \\ 1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 12202011\mathbf{2}0111212 \\ 2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 20111212\mathbf{1}2201220 \end{array}$$

:substituion

Figure 4. A substitution satisfying the coincidence condition

The importance of the coincidence condition comes from the crucial fact (established by Dekking [18]) that it is this condition that completely characterizes the constant length substitutions which give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group, see [49, Chapter 7] for a survey. For us, however, the coincidence condition is primarily interesting as yet another incarnation of synchronizability. Indeed, there is a straightforward bijection between DFAs and constant length substitutions. Each DFA $\mathcal{A} = (Q, A)$ with $A = \{a_1, \ldots, a_\ell\}$ defines a length ℓ substitu-

tion on Q that maps every $q \in Q$ to the word $(q \cdot a_1) \dots (q \cdot a_\ell) \in Q^+$. (For instance, the automaton $\mathcal{C}_4$ in Figure 1 induces the substitution $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 23,\ 3 \mapsto 30$.) Conversely, each substitution $\sigma : X \to X^+$ such that all words $\sigma(x)$, $x \in X$, have the same length ℓ gives rise to a DFA for which X serves as the state set and which has ℓ input letters $a_1, \dots, a_\ell$, say, acting on X as follows: $x \cdot a_i$ is the symbol in the i-th position of the word $\sigma(x)$. (For instance, the substitution τ considered in the previous paragraph defines the automaton shown in Figure 5.) It is clear that under the described bijection

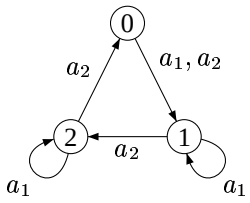


Figure 5. The automaton induced by the substitution $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$

substitutions satisfying the coincidence condition correspond precisely to synchronizing automata, and moreover, given a substitution, the number of iterations at which the coincidence first occurs is equal to the minimum length of reset word for the corresponding automaton.

We mention in passing a purely algebraic framework within which synchronizing automata also appear in a natural way. One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set. A *term* in the language of such unary algebras is an expression t of the form $x \cdot w$, where x is a variable and w is a word over an alphabet A. An *identity* is a formal equality between two terms. A DFA $\mathcal{A} = (Q, A)$ *satisfies* an identity $t_1 = t_2$, where the words involved in the terms t_1 and t_2 are over A, if t_1 and t_2 take the same value under each interpretation of their variables in the set Q. Identities of unary algebras can be of the from either $x \cdot u = x \cdot v$ (*homotypical* identities) or $x \cdot u = y \cdot v$ with $x \neq y$ (*heterotypical* identities). It is easy to realize that a DFA is synchronizing if and only if it satisfies a heterotypical identity, and thus, studying synchronizing automata may be considered as a part of the equational logic of unary algebras. In particular, synchronizing automata over a fixed alphabet form a *pseudovariety* of unary algebras. See [9] for a survey of numerous publications in this direction; it is fair to say, however, that so far this algebraic approach has not proved to be really useful for understanding the combinatorial nature of synchronizing automata.

If space permits!!

2 Algorithmic and complexity issues

It should be clear that not every DFA is synchronizing. Therefore, the very first question that we should address is the following one: *given an automaton $\mathcal{A}$, how to determine whether or not $\mathcal{A}$ is synchronizing?*

This question is in fact quite easy, and the most straightforward solution to it can be achieved via the classic subset construction by Rabin and Scott [50]. Given a DFA

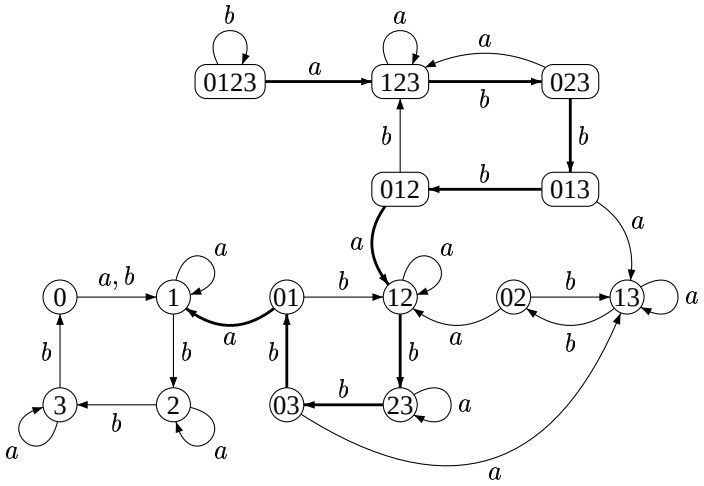


Figure 6. The power automaton $\mathcal{P}(\mathcal{C}_4)$

$\mathcal{A} = (Q, A)$, we define its *subset automaton* $\mathcal{P}(\mathcal{A})$ on the set of the non-empty subsets of Q by setting $P \cdot a = \{p \cdot a \mid p \in P\}$ for each non-empty subset P of Q and each $a \in \mathcal{A}$. (Since we start with a deterministic automaton, we do not need adding the empty set to the state set of $\mathcal{P}(\mathcal{A})$.) Figure 6 presents the subset automaton for the DFA $\mathcal{C}_4$ shown in Figure 1.

Now it is obvious that a word $w \in A^*$ is a reset word for the DFA $\mathcal{A}$ if and only if w labels a path in $\mathcal{P}(\mathcal{A})$ starting at Q and ending at a singleton. (For instance, the bold path in Figure 6 represents the shortest reset word ab^3ab^3a of the automaton $\mathcal{C}_4$.) Thus, the question of whether or not a given DFA $\mathcal{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the subset automaton $\mathcal{P}(\mathcal{A})$: is there a path from Q to a singleton? The latter question can be easily answered by breadth-first search, see, e.g., [17, Section 22.2].

The described procedure is conceptually very simple but rather inefficient because the power automaton $\mathcal{P}(\mathcal{A})$ is exponentially larger than $\mathcal{A}$. However, the following criterion of synchronizability gives rise to a polynomial algorithm.

Proposition 2.1 ([13, Theorem 2]). *A DFA $\mathcal{A} = (Q, A)$ is synchronizing if and only if for every $q, q' \in Q$ there exists a word $w \in A^*$ such that $q \cdot w = q' \cdot w$.*

Proof. Of course, only sufficiency needs a proof. For this, take two states $q, q' \in Q$ and consider a word w_1 such that $q \cdot w_1 = q' \cdot w_1$. Then $|Q \cdot w_1| < |Q|$. If $|Q \cdot w_1| = 1$, then w_1 is a reset word and $\mathcal{A}$ is synchronizing. If $|Q \cdot w_1| > 1$, take two states $p, p' \in Q \cdot w_1$ and consider a word w_2 such that $p \cdot w_2 = q' \cdot w_2$. Then $|Q \cdot w_1 w_2| < |Q \cdot w_1|$. If $|Q \cdot w_1 w_2| = 1$, then $w_1 w_2$ is a reset word; otherwise we repeat the process. Clearly, a reset word for $\mathcal{A}$ will be constructed in at most $|Q| - 1$ steps. □

One can treat Proposition 2.1 as a reduction of the synchronizability problem to a reachability problem in the subautomaton $\mathcal{P}^{[2]}(\mathcal{A})$ of $\mathcal{P}(\mathcal{A})$ whose states are *couples* (2-

element subsets) and singletons of Q. Since the subautomaton has $\frac{|Q|(|Q|+1)}{2}$ states, breadth-first search solves this problem in $O(|Q|^2 \cdot |A|)$ time. This complexity bound assumes that no reset word is explicitly calculated. If one requires that, whenever $\mathcal{A}$ turns out to be synchronizing, a reset word is produced, then the best of the known algorithms (which is basically due to Eppstein [20, Theorem 6], see also [61, Theorem 1.15]) has an implementation that consumes $O(|Q|^3 + |Q|^2 \cdot |A|)$ time and $O(|Q|^2 + |Q| \cdot |A|)$ working space, not counting the space for the output which is $O(|Q|^3)$.

For a synchronizing automaton, the subset automaton can be used to construct shortest reset words as they correspond to shortest paths from the whole state set Q to a singleton. Of course, this requires exponential (of $|Q|$) time in the worst case. Nevertheless, there were attempts to implement this approach, see, e.g., [52, 66]. One may hope that, as above, a suitable calculation in the "polynomial" subautomaton $\mathcal{P}^{[2]}(\mathcal{A})$ may yield a polynomial algorithm. However, it is not the case, and moreover, as we will see, it is very unlikely that any reasonable algorithm may exist for finding shortest reset words in general synchronizing automata. In the following discussion we assume the reader's acquaintance with some basics of computational complexity (such as the definitions of the complexity classes NP and coNP) that can be found, e.g., in [24, 43].

Consider the following decision problem:

SHORT-RESET-WORD: *Given a synchronizing automaton $\mathcal{A}$ and a positive integer ℓ, is it true that $\mathcal{A}$ has a reset word of length ℓ?*

Clearly, SHORT-RESET-WORD belongs to the complexity class NP: one can nondeterministically guess a word $w \in A^*$ of length ℓ and then check if w is a reset word for $\mathcal{A}$ in time $\ell|Q|$. Several authors [55, 20, 30, 59, 60] have proved that SHORT-RESET-WORD is NP-hard by a polynomial reduction from SAT (the satisfiability problem for a system of *clauses*, that is, disjunctions of literals). We reproduce here Eppstein's reduction from [20].

Given an arbitrary instance ψ of SAT with n variables $x_1, \dots, x_n$ and m clauses $c_1, \dots, c_m$, we construct a DFA $\mathcal{A}(\psi)$ with 2 input letters a and b as follows. The state set Q of $\mathcal{A}(\psi)$ consists of $(n+1)m$ states $q_{i,j}$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n+1$, and a special state z. The transitions are defined by

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

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

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

KV:fig:A2_example
Figure 7 shows two automata of the form $\mathcal{A}(\psi)$ build for the SAT instances

$$\psi_1 = \{x_1 \vee x_2 \vee x_3,\ \neg x_1 \vee x_2,\ \neg x_2 \vee x_3,\ \neg x_2 \vee \neg x_3\},$$
$$\psi_2 = \{x_1 \vee x_2,\ \neg x_1 \vee x_2,\ \neg x_2 \vee x_3,\ \neg x_2 \vee \neg x_3\}.$$

KV:fig:A2_example
If at some state $q \in Q$ in Figure 7 there is no outgoing edge labelled $c \in \{a, b\}$, the edge $q \xrightarrow{c} z$ is assumed (those edges are omitted to improve readability). The two instances differ only in the first clause: in ψ_1 it contains the literal x_3 while in ψ_2 it does not. Cor-

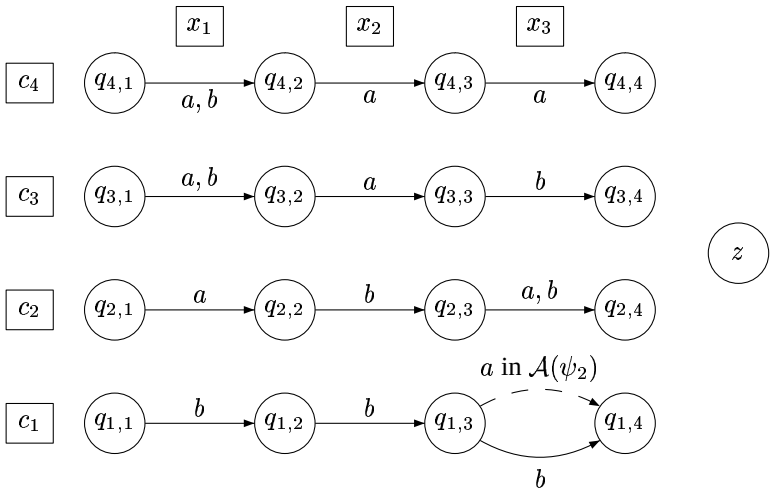


Figure 7. The automata $\mathcal{A}(\psi_1)$ and $\mathcal{A}(\psi_2)$

g:A2_example

respondingly, the automata $\mathcal{A}(\psi_1)$ and $\mathcal{A}(\psi_2)$ differ only by the outgoing edge labelled a at the state $q_{1,3}$: in $\mathcal{A}(\psi_1)$ it leads to z (and therefore, it is not shown) while in $\mathcal{A}(\psi_2)$ 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 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 reset by no word of length 3 but by every word of length 4.

In general, it is easy to see that $\mathcal{A}(\psi)$ is reset by every word of length $n+1$ and is reset by a word of length n if and only if ψ is satisfiable. Therefore assigning the instance $(\mathcal{A}(\psi), n)$ of SHORT-RESET-WORD to an arbitrary n-variable instance ψ of SAT, one obtains a polynomial reduction of the latter problem to the former. Since SAT is NP-complete and SHORT-RESET-WORD lies in NP, we obtain the following.

:complexity1

Proposition 2.2. *The problem* SHORT-RESET-WORD *is NP-complete.* □

In fact, as observed by Samotij [60], the above construction yields slightly more[2]. Consider the following decision problem:

SHORTEST-RESET-WORD: *Given a synchronizing automaton $\mathcal{A}$ and a positive integer ℓ, is it true that the minimum length of a reset word for $\mathcal{A}$ is equal to ℓ?*

Assigning the instance $(\mathcal{A}(\psi), n+1)$ of SHORTEST-RESET-WORD to an arbitrary system ψ of clauses on n variables, one sees that the answer to the instance is "Yes" if and only if ψ is not satisfiable. Thus, we have a polynomial reduction from the negation of SAT to SHORTEST-RESET-WORD whence the latter problem is coNP-hard. As a corollary, SHORTEST-RESET-WORD cannot belong to NP unless NP = coNP which is commonly considered to be very unlikely. In other words, even non-deterministic algorithms cannot decide the *reset threshold* of a given synchronizing automaton, (that is, the minimum length of its reset words) in polynomial time.

[2] Actually, the reduction in [60] is not correct but the result claimed can be easily recovered as shown below.

The exact complexity of the problem SHORTEST-RESET-WORD has been recently determined by Gawrychowski [25] and, independently, by Olschewski and Ummels [42]. It turns out that the appropriate complexity class is DP (Difference Polynomial-Time) introduced by Papadimitriou and Yannakakis [44]; this class consists of languages of the form $L_1 \cap L_2$ where L_1 is a language from NP and a L_2 is a language in coNP. A "standard" DP-complete problem is SAT-UNSAT whose instance is a pair of clause systems ψ, χ, say, and whose question is whether ψ is satisfiable and χ is unsatisfiable.

Proposition 2.3. *The problem* SHORTEST-RESET-WORD *is DP-complete.* □

Proposition 2.3 follows from mutual reductions between SHORTEST-RESET-WORD and SAT-UNSAT obtained in [25, 42].

The complexity class $\mathsf{P}^{\mathsf{NP}[\log]}$ consists of all problems solvable by a deterministic polynomial-time Turing machine that has an access to an oracle for an NP-complete problem, with the number of queries being logarithmic in the size of the input. The class DP is contained in $\mathsf{P}^{\mathsf{NP}[\log]}$ (if fact, for every problem in DP two oracle queries suffice) and the inclusion is believed to be strict. Olschewski and Ummels [42] have shown that the problem of computing the reset threshold (as opposed to deciding whether it is equal to a given integer) is complete for the functional analogue $\mathsf{FP}^{\mathsf{NP}[\log]}$ of the class $\mathsf{P}^{\mathsf{NP}[\log]}$ (see [62] for a discussion of functional complexity classes). Hence, this problem appears to be even harder than deciding the reset threshold. Recently Berlinkov [6] has shown (assuming $\mathsf{P} \neq \mathsf{NP}$) that no polynomial algorithm can approximate within a constant factor the reset threshold of a given synchronizing automaton with two input letters.

The problem of finding a reset word of minimum length (as opposed to computing only the length without writing down the word itself) may be even more difficult. From the cited result of [42] it follows that the problem is $\mathsf{FP}^{\mathsf{NP}[\log]}$-hard but its exact complexity is not known yet.

The hardness results in [6, 42] are obtained via suitable encodings of SAT in the flavor of the above proof of Proposition 2.2. Gerbush and Heeringa [26] have observed that some other well-known hard problems such as SCS (SHORTEST COMMON SUPERSEQUENCE) or SET COVER admit a transparent reduction to the problem of finding a reset word of minimum length for a given synchronizing automaton. In particular, since SCS is known to have no approximation within a constant factor unless $\mathsf{P} = \mathsf{NP}$ [32], they have deduced a similar conclusion for approximating the reset threshold but—in contrast to the cited result of [6]—without any bound on the size of the input alphabet. Moreover, using a recent result on SET COVER [1], they have concluded that the reset threshold of synchronizing automata with n states and unbounded alphabet cannot be approximated within the factor $c \log n$ for some constant $c > 0$ unless $\mathsf{P} = \mathsf{NP}$. It is a challenging problem to study approximation of the reset threshold within a logarithmic factor for synchronizing automata with a fixed alphabet size.

We mention that Pixley, Jeong and Hachtel [48] suggested an heuristic polynomial algorithm for finding short reset words in synchronizing automata that was reported to perform rather satisfactory on a number of benchmarks from [69]; further polynomial algorithms yielding short (though not necessarily shortest) reset words have been implemented by Trahtman [66] and Roman [54, 53]. Some algorithms for finding reset words will be also discussed in the next section.

3 Around the Černý conjecture

The Černý conjecture. A very natural question to ask is the following: *given a positive integer n, how long can be reset words for synchronizing automata with n states?* Černý [13] found a lower bound by constructing, for each $n > 1$, a synchronizing automaton $\mathcal{C}_n$ with n states and 2 input letters whose shortest reset word has length $(n-1)^2$. We assume that the state set of $\mathcal{C}_n$ is $Q = \{0, 1, 2, \ldots, n-1\}$ and the input letters are a and b, subject to the following action on Q:

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

Our first example of synchronizing automaton (see Figure 1) is, in fact, $\mathcal{C}_4$. A generic automaton $\mathcal{C}_n$ is shown in Figure 8 on the left.

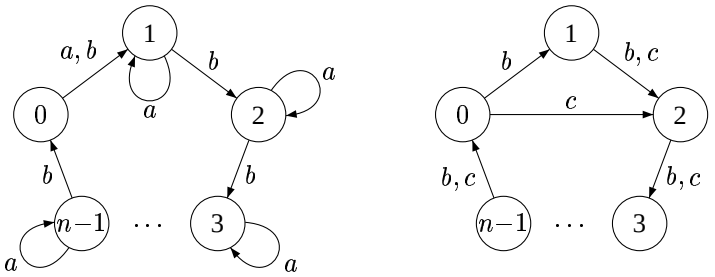


Figure 8. The DFA $\mathcal{C}_n$ and the DFA $\mathcal{W}_n$ induced by the actions of b and $c = ab$

The series $\{\mathcal{C}_n\}_{n=2,3,\ldots}$ was rediscovered many times (see, e.g., [38, 21, 20, 23]). It is easy to see that the word $(ab^{n-1})^{n-2}a$ of length $n(n-2)+1 = (n-1)^2$ resets $\mathcal{C}_n$.

Proposition 3.1 ([13, Lemma 1]). *Any reset word for $\mathcal{C}_n$ has length at least $(n-1)^2$.*

There are several nice proofs for this result. Here we present a recent proof from [2]; it is based on a transparent idea and reveals an interesting connection between Černý's automata $\mathcal{C}_n$ and an extremal series of digraphs discovered in Wielandt's classic paper [68] (see Section 4).

Proof of Proposition 3.1. Let w be a reset word of minimum length for $\mathcal{C}_n$. 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, $w = w'a$ for some $w' \in \{a, b\}^*$ such that the image of Q under the action of w' is precisely the set $\{0, 1\}$.

Since the letter a fixes each state in its image $\{1, 2, \ldots, n-1\}$, every occurrence of a in w except the last one is followed by an occurrence of b. (Otherwise a^2 occurs in w as a factor and reducing this factor to just a results in a shorter reset word.) Therefore, if we let $c = ab$, then the word w' can be rewritten into a word v over the alphabet $\{b, c\}$. The actions of b and c induce a new DFA on the state set Q; we denote this induced DFA (shown in Figure 8 on the right) by $\mathcal{W}_n$. Since w' and v act on Q in the same way, the word vc is a reset word for $\mathcal{W}_n$ and brings the automaton to the state 2.

If $u \in \{b,c\}^*$, the word uvc also is a reset word for $\mathcal{W}_n$ and it also brings the automaton to 2. Hence, for every $\ell \geqslant |vc|$, there is a path of length ℓ in $\mathcal{W}_n$ 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 $\mathcal{W}_n$. The underlying digraph of $\mathcal{W}_n$ has simple cycles only of two lengths: n and $n-1$. Each cycle of $\mathcal{W}_n$ must consist of simple cycles of these two lengths whence each number $\ell \geqslant |w|$ must be expressible as a non-negative integer combination of n and $n-1$. Here we invoke the following well-known and elementary result from arithmetics:

Lemma 3.2 ([51, Theorem 2.1.1]). *If k_1, k_2 are relatively prime positive integers, 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 3.2 implies that $|vc| > n(n-1) - n - (n-1) = n^2 - 3n + 1$. Suppose that $|vc| = n^2 - 3n + 2$. Then there should be a path of this length from the state 1 to the state 2. Every outgoing edge of 1 leads to 2, and thus, in the path it must be followed by a cycle of length $n^2 - 3n + 1$. No cycle of such length may exist by Lemma 3.2. Hence $|vc| \geqslant n^2 - 3n + 3$.

Since the action of b on any set S of states cannot change the cardinality of S and the action of c can decrease the cardinality by 1 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 contain 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, $|w| = |w'a| \geqslant n^2 - 2n + 1 = (n-1)^2$. □

If we define the *Černý function* $\mathfrak{C}(n)$ as the maximum length of shortest reset words for synchronizing automata with n states, the above property of the series $\{\mathcal{C}_n\}$, $n = 2, 3, \ldots$, yields the inequality $\mathfrak{C}(n) \geqslant (n-1)^2$. The *Černý conjecture* is the claim that the equality $\mathfrak{C}(n) = (n-1)^2$ holds true.

In the literature, one often refers to Černý's paper [13] as the source of the Černý conjecture. In fact, the conjecture was not yet formulated in that paper. There Černý only observed that $(n-1)^2 \leqslant \mathfrak{C}(n) \leqslant 2^n - n - 1$ and 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."

The conjecture in its present-day form was formulated a bit later, after the expectation in the above quotation was confirmed by [63]. (Namely, Starke improved the upper bound from [13] to $1 + \frac{n(n-1)(n-2)}{2}$, which was the first polynomial upper bound for $\mathfrak{C}(n)$.) Černý explicitly stated the conjecture $\mathfrak{C}(n) = (n-1)^2$ in his talks in the second half of the 1960s; in print the conjecture first appeared in [14].

An upper bound. The best upper bound for the Černý function achieved so far guarantees that for every synchronizing automaton with n states there exists a reset word of length $\frac{n^3-n}{6}$. Such a reset word arises as the output of the following greedy algorithm.

If $|Q| = n$, then clearly the main loop of Algorithm 1 is executed at most $n-1$ times. Finding the word v in line 7 amounts to reading the labels along a shortest path between a couple contained in P and a singleton in the automaton $\mathcal{P}^{[2]}(\mathcal{A})$ (see the discussion after

```
GREEDYCOMPRESSION(A)
1:  w ← ε                                   ▷ Initializing the current word
2:  P ← Q                                   ▷ Initializing the current set
3:  while |P| > 1 do
4:    if |P · u| = |P| for all u ∈ A* then
5:      return Failure
6:    else
7:      take a word v ∈ A* of minimum length with |P · v| < |P|
8:    w ← wv                                ▷ Updating the current word
9:    P ← P · v                             ▷ Updating the current set
10: return w
```

Algorithm 1. Compression algorithm calculating a reset word for $\mathcal{A} = (Q, A)$

Proposition 2.1). Breadth-first search does this in $O(n^2 \cdot |A|)$ time. Thus, Algorithm 1 is polynomial in the size of $\mathcal{A}$. In order to evaluate the length of the output word w, we estimate the length of each word v produced by the main loop.

Consider a generic step at which $|P| = k > 1$ and let $v = a_1 \cdots a_\ell$ with $a_i \in \Sigma$, $i = 1, \ldots, \ell$. Then each of the sets

$$P_1 = P,\ P_2 = P_1 \cdot a_1,\ \ldots,\ P_\ell = P_{\ell-1} \cdot a_{\ell-1}$$

contains exactly k states. Furthermore, since $|P_\ell \cdot a_\ell| < |P_\ell|$, there exist two distinct states $p_\ell, p'_\ell \in P_\ell$ such that $p_\ell \cdot a_\ell = p'_\ell \cdot a_\ell$. Now define couples $R_i = \{p_i, p'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $p_i \cdot a_i = p_{i+1}, p'_i \cdot a_i = p'_{i+1}$ for $i = 1, \ldots, \ell - 1$.

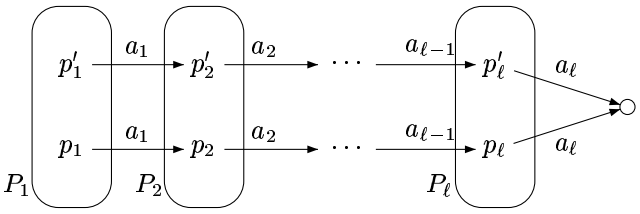


Figure 9. Combinatorial configuration at a generic step of Algorithm 1

Then the condition that v is a word of minimum length with $|P \cdot v| < |P|$ implies that $R_i \not\subseteq P_j$ for $1 \leqslant j < i \leqslant \ell$. Indeed, if $R_i \subseteq P_j$ for some $j < i$, then already the word $a_1 \cdots a_j a_i \cdots a_\ell$ of length $j + \ell - i < \ell$ would satisfy $|P \cdot a_1 \cdots a_j a_i \cdots a_\ell| < |P|$ contradicting the choice of v. Thus, we arrive at a problem from combinatorics of finite sets that can be stated as follows. Let $1 < k \leqslant n$. A sequence of k-element subsets $P_1, P_2, \ldots$ of an n-element set is called 2-*renewing* if each P_i contains a couple R_i such that $R_i \not\subseteq P_j$ for each $j < i$. What is the maximum length of a 2-renewing sequence as a function of n and k?

The problem was solved by Frankl [22] who proved the following result[3].

[3] Actually Frankl [22] considered and solved a more general problem concerning the maximum length of (analogously defined) m-renewing sequences of k-element subsets in an n-element set for any fixed $m \leqslant k$.

Proposition 3.3. *The maximum length of a 2-renewing sequence of k-element subsets in an n-element set is equal to $\binom{n-k+2}{2}$.*

Thus, if ℓ_k is the length of the word v that Algorithm 1 appends to the current word w after the iteration step that the algorithm enters while the current set P contains k states, then Proposition 3.3 guarantees that $\ell_k \leqslant \binom{n-k+2}{2}$. Summing up all these inequalities from $k = n$ to $k = 2$, one arrives at the aforementioned bound

$$\mathfrak{C}(n) \leqslant \frac{n^3 - n}{6}. \tag{3.1}$$

In the literature the bound (3.1) is usually attributed to Pin who explained the above connection between Algorithm 1 and the combinatorial problem on the maximum length of 2-renewing sequences and conjectured the estimation $\binom{n-k+2}{2}$ for this length in his talk at the Colloquium on Graph Theory and Combinatorics held in Marseille in 1981. (Frankl learned this conjecture from Pin—and proved it—during another colloquium on combinatorics held in Bielefeld in November 1981.) Accordingly, the usual reference for (3.1) is the paper [47] based on the talk. The full story is however more complicated. Actually, the bound (3.1) first appeared in [21] where it was deduced from a combinatorial conjecture equivalent to Pin's one. The conjecture however remained unproved. The bound (3.1) then reoccurred in [36, 37] but the argument justifying it in these papers was insufficient. In 1987 both (3.1) and Proposition 3.3 were independently rediscovered by Klyachko, Rystsov and Spivak [35] who were aware of [21, 36, 37] but neither [47] nor [22]. We include here a proof of Frankl's result following [35].

If space permits!!

Proof of Proposition 3.3. Let $Q = \{1, 2, \ldots, n\}$. First, we exhibit a 2-renewing sequence of k-element subsets in Q of length $\binom{n-k+2}{2}$. For this put $W = \{1, \ldots, k-2\}$, list all $\binom{n-k+2}{2}$ couples of $Q \setminus W$ in some order and let T_i be the union of W with the i-th couple in the list. Clearly, the sequence $T_1, \ldots, T_{\binom{n-k+2}{2}}$ is 2-renewing.

Now we assign to each k-element subset $S = \{s_1, \ldots, s_k\}$ of Q the following polynomial $D(S)$ in variables $x_{s_1}, \ldots, x_{s_k}$ over the field $\mathbb{R}$ of reals:

$$D(S) = \begin{vmatrix} 1 & s_1 & s_1^2 & \cdots & s_1^{k-3} & x_{s_1} & x_{s_1}^2 \\ 1 & s_2 & s_2^2 & \cdots & s_2^{k-3} & x_{s_2} & x_{s_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & s_k & s_k^2 & \cdots & s_k^{k-3} & x_{s_k} & x_{s_k}^2 \end{vmatrix}_{k \times k}.$$

Observe that for any 2-renewing sequence $S_1, \ldots, S_\ell$ of k-element subsets in Q, the polynomials $D(S_1), \ldots, D(S_\ell)$ are linearly independent. Indeed, if they were linearly dependent, then by a basic lemma of linear algebra, some polynomial $D(S_j)$ should be expressible as a linear combination of the preceding polynomials $D(S_1), \ldots, D(S_{j-1})$. By the definition of a 2-renewing sequence, S_j contains a couple $\{s, s'\}$ such that $\{s, s'\} \not\subseteq S_i$ for all $i < j$. If we substitute $x_s = s$, $x_{s'} = s'$ and $x_t = 0$ for $t \neq s, s'$ in each polynomial $D(S_1), \ldots, D(S_j)$, then the polynomials $D(S_1), \ldots, D(S_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials. The value of $D(S_j)$ however is the determinant being the product of a Vandermonde $(k-2) \times (k-2)$-determinant with the

2×2-determinant $\begin{vmatrix} s & s^2 \\ s' & (s')^2 \end{vmatrix}$, whence this value is not 0. Hence $D(S_j)$ cannot be equal to a linear combination of $D(S_1), \ldots, D(S_{j-1})$.

We see that the length of any 2-renewing sequence cannot exceed the dimension of the linear space over $\mathbb{R}$ spanned by all polynomials of the form $D(S)$. In order to prove that the dimension is at most $\binom{n-k+2}{2}$, it suffices to show that the space is spanned by the polynomials $D(T_1), \ldots, D(T_{\binom{n-k+2}{2}})$, where $T_1, \ldots, T_{\binom{n-k+2}{2}}$ is the 2-renewing sequence constructed in the first paragraph of the proof. For this, take an arbitrary k-element subset $S = \{s_1, \ldots, s_k\}$ of Q. We claim that the polynomial $D(S)$ is a linear combination of $D(T_1), \ldots, D(T_{\binom{n-k+2}{2}})$. We induct on the cardinality of the set $S \setminus W$. If $|S \setminus W| = 2$, then S is the union of W with some couple from $Q \setminus W$, whence $S = T_i$ for some $i = 1, \ldots, \binom{n-k+2}{2}$. Thus, $D(S) = D(T_i)$ and our claim holds true. If $|S \setminus W| > 2$, there is $s_0 \in W \setminus S$. Let $S' = S \cup \{s_0\}$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \cdots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(s_0) = 1$ and $p(s) = 0$ for all $s \in W \setminus \{s_0\}$. Consider the determinant

$$\Delta = \begin{vmatrix} p(s_0) & 1 & s_0 & s_0^2 & \cdots & s_0^{k-3} & x_{s_0} & x_{s_0}^2 \\ p(s_1) & 1 & s_1 & s_1^2 & \cdots & s_1^{k-3} & x_{s_1} & x_{s_1}^2 \\ p(s_2) & 1 & s_2 & s_2^2 & \cdots & s_2^{k-3} & x_{s_2} & x_{s_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(s_k) & 1 & s_k & s_k^2 & \cdots & s_k^{k-3} & x_{s_k} & x_{s_k}^2 \end{vmatrix}_{(k+1)\times(k+1)} .$$

Clearly, $\Delta = 0$ as the first column is the sum of the next $k-2$ columns with the coefficients $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-3}$. Thus, expanding Δ by the first column gives the identity

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

Since $p(s_0) = 1$ and $S' \setminus \{s_0\} = S$, the identity rewrites as

$$D(S) = \sum_{j=1}^{k} (-1)^{j+1} p(s_j) D(S' \setminus \{s_j\}), \qquad (3.2)$$

KV:eq:combination

and since $p(s) = 0$ for all $s \in W \setminus \{s_0\}$ all the non-zero summands in the right-hand side are such that $s_j \notin W$. For each such s_j, we have

$$(S' \setminus \{s_j\}) \setminus W = S' \setminus (W \cup \{s_j\}) = (S \cup \{s_0\}) \setminus (W \cup \{s_j\}) = (S \setminus W) \setminus \{s_j\},$$

whence $|(S' \setminus \{s_j\}) \setminus W| = |S \setminus W| - 1$ and by the inductive assumption, the polynomials $D(S' \setminus \{s_j\})$ are linear combinations of $D(T_1), \ldots, D(T_{\binom{n-k+2}{2}})$. From (3.2) we conclude that this holds true for the polynomial $D(S)$ as well. □

KV:eq:combination

KV:Greedy

KV:fig:power_automaton

If one executes Algorithm 1 on the Černý automaton $\mathcal{C}_4$ (Figure 6 is quite helpful here), one sees that the algorithm returns the word ab^2abab^3a of length 10 which is not the shortest reset word for $\mathcal{C}_4$. 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 also in all intermediate steps. In our

example, the optimal solution is the word ab^3ab^3a but it cannot be found by Algorithm KV:Greedy because the algorithm chooses $v = b^2a$ rather than $v = b^3a$ on the second execution of the main loop. Actually, the gap between the reset threshold of a synchronizing automaton and the length of the reset word that Algorithm KV:Greedy returns on the automaton may be arbitrarily large[4]: one can calculate that for the Černý automaton $\mathcal{C}_n$ whose reset threshold is $(n-1)^2$ Algorithm KV:Greedy produces a reset word of length $\Omega(n^2 \log n)$. The behaviour of Algorithm KV:Greedy on average is not yet understood; practically it behaves rather well.

The extension algorithm. While studying Algorithm KV:Greedy has provided the best currently known upper bound for the Černý function in the general case, the most impressive partial results proving the Černý conjecture for some special classes of automata have been obtained via analysis a different algorithm. This algorithm also operates in a greedy manner but builds a reset word in the opposite direction.

For a DFA $\mathcal{A} = (Q, A)$, a subset $P \subseteq Q$ and a word $w \in A^*$, we denote by Pw^{-1} the full pre-image of P under the action of w, that is, $Pw^{-1} = \{q \in Q \mid q \cdot w \in P\}$. In what follows, we denote the same a singleton set and its single element to lighten notation.

GREEDYEXTENSTION($\mathcal{A}$)
1: **if** $|qa^{-1}| = 1$ for all $q \in Q$ and $a \in A$ **then**
2: **return** Failure
3: **else**
4: $w \leftarrow a$ such that $|qa^{-1}| > 1$ ▷ Initializing the current word
5: $P \leftarrow qa^{-1}$ such that $|qa^{-1}| > 1$ ▷ Initializing the current set
6: **while** $|P| < |Q|$ **do**
7: **if** $|Pu^{-1}| \leqslant |P|$ for all $u \in A^*$ **then**
8: **return** Failure
9: **else**
10: take a word $v \in A^*$ of minimum length with $|Pv^{-1}| > |P|$
11: $w \leftarrow vw$ ▷ Updating the current word
12: $P \leftarrow Pv^{-1}$ ▷ Updating the current set
13: **return** w

KV:Extension

Algorithm 2. Extension algorithm calculating a reset word for $\mathcal{A} = (Q, A)$

In contrast to Algorithm KV:Greedy, it is not clear whether Algorithm KV:Extension admits a polynomial-time implementation. Moreover, in general we know no non-trivial bound on the length of the words v that the main loop of Algorithm KV:Extension appends to the current word. However, one can isolate some cases in which rather strong bounds on $|v|$ do exist. The following definition is convenient for subsequent discussion. Given a number $\alpha > 0$, a DFA $\mathcal{A} = (Q, A)$ is said to be *α-extensible* if for each proper non-singleton subset $S \subset Q$, there exists a word $u \in A^*$ of length at most $\alpha|Q|$ such that $|Su^{-1}| > |S|$. The following observation explains the importance of this property.

[4] We observe that this does not immediately follow from the non-approximation results discussed in Section KV:sec:algorithms&complexity because Algorithm KV:Greedy is not really deterministic. Indeed, in general there may be several words satisfying the conditions in line 7 of the algorithm and it has not been specified which one of the words should be taken.

xtensibility

Proposition 3.4. *If $\mathcal{A}$ is an α-extensible automaton with n states, then $\mathcal{A}$ is synchronizing and the reset threshold of $\mathcal{A}$ is at most $1+\alpha n(n-2)$. In particular, the Černý conjecture holds true for 1-extensible automata.*

KV:Extension

Proof. If we run Algorithm 2 on $\mathcal{A}$, the main loop is executed at most $n-2$ times and each word that it appends to the current word 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 get the bound $1+n(n-2)=(n-1)^2$ which complies with the Černý conjecture. □

The approach to the Černý conjecture via extensibility traces back to Pin's paper [46] of 1978. Pin observed that every DFA $\mathcal{A}=(Q,A)$ such that $|Q|$ is prime and some letter acts as a cyclic permutation of Q is 1-extensible provided some other letter acts on Q as a non-permutation. Thus, such $\mathcal{A}$ is synchronizing and its reset threshold does not exceed $(|Q|-1)^2$. 20 years later Dubuc [19] generalized Pin's result by showing that every synchronizing automata in which some letter acts as a cyclic permutation of the state set is 1-extensible. Kari [34] proved 1-extensibility of synchronizing automata whose underlying digraphs are Eulerian. In all these papers 1-extensibility is obtained via linear-algebraic arguments; we include here a proof from [34] as quite a representative example of these linearization techniques.

If space permits!!

thm:eulerian

Theorem 3.5 ([34, Theorem 2]). *If the underlying digraph of a synchronizing automata $\mathcal{A}=(Q,A)$ is Eulerian then $\mathcal{A}$ is has a reset word of length at most $(n-2)(n-1)+1$, where $n=|Q|$.*

Proof. For every vertex in an Eulerian digraph, its in-degree and its out-degree are equal. In the underlying graph of a DFA the out-degree of every vertex is equal to the cardinality of the input alphabet. Hence, if $|A|=k$, then each vertex in the underlying digraph of $\mathcal{A}$ has in-degree k and for every subset $P\subseteq Q$, the equality

$$\sum_{a\in A}|Pa^{-1}|=k|P| \tag{3.3}$$

KV:eq:eulerian

KV:eq:eulerian

KV:eq:eulerian

holds true since the left-hand side of (3.3) is the number of edges in the underlying digraph of $\mathcal{A}$ with ends in P. The equality (3.3) readily implies that for each $P\subseteq Q$, one of the following alternatives takes place: either $|Pa^{-1}|=|P|$ for all letters $a\in A$ or $|Pb^{-1}|>|P|$ for some $b\in A$. Now assume that a subset $S\subseteq Q$ and a word $u\in A^+$ are such that $|Su^{-1}|\neq|S|$ and u is a word of minimum length with this property. We write $u=aw$ for some $a\in A$ and $w\in A^*$ and let $P=Sw^{-1}$. Then $|P|=|S|$ by the choice of u and $Pa^{-1}=Su^{-1}$ whence $|Pa^{-1}|\neq|P|$. Thus, P must fall into the second of the above alternatives and so $|Pb^{-1}|>|P|$ for some $b\in A$. The word $v=bw$ has the same length as u and has the property that $|Sv^{-1}|>|S|$. Having this in mind, we now aim to prove that for every proper subset $S\subset Q$, there exists a word $u\in A^*$ of length at most $n-1$ such that $|Su^{-1}|\neq|S|$.

It is here where linear algebra comes into the play. We may assume that $Q=\{1,2,\ldots,n\}$. Assign to each subset $P\subseteq Q$ its *characteristic vector* $[P]$ in the linear space $\mathbb{R}^n$ of n-dimensional row vectors over $\mathbb{R}$ as follows: i-th entry of $[P]$ is 1 if $i\in P$, otherwise it is equal to 0. For instance, $[Q]$ is the all ones row vector and the vectors

$[1], \dots, [n]$ form the standard basis of $\mathbb{R}^n$. Observe that for any vector $x \in \mathbb{R}^n$, the inner product $\langle x, [Q]\rangle$ is equal to the sum of all entries of x. In particular, for each subset $P \subseteq Q$, we have $\langle [P], [Q]\rangle = |P|$. Further, assign to each word $w \in A^*$ the linear operator φ_w on $\mathbb{R}^n$ defined by $\varphi_w([i]) = [iw^{-1}]$ for each $i \in Q$. It is then clear that $\varphi_w([P]) = [Pw^{-1}]$ for each $P \subseteq Q$.

The inequality $|Su^{-1}| \neq |S|$ that we look for can be rewritten as $\langle \varphi_u([S]), [Q]\rangle \neq \langle [S], [Q]\rangle$ or $\langle \varphi_u([S]) - [S], [Q]\rangle \neq 0$. Let $x = [S] - \frac{|S|}{n}[Q]$. Then $x \neq 0$ as $S \neq Q$ and $\langle x, [Q]\rangle = 0$. Since $Qu^{-1} = Q$ for every word u, we have $\varphi_u([Q]) = [Q]$. Hence

$$\langle \varphi_u([S]) - [S], [Q]\rangle = \langle \varphi_u(x + \lambda[Q]) - (x + \lambda[Q]), [Q]\rangle = \langle \varphi_u(x) + \lambda[Q] - x - \lambda[Q]), [Q]\rangle = \langle \varphi_u(x) - x, [Q]\rangle = \langle \varphi_u(x), [Q]\rangle.$$

Thus, a word u satisfies $|Su^{-1}| \neq |S|$ if and only if the vector $\varphi_u(x)$ lies beyond the subspace U of all vectors orthogonal to $[Q]$. We aim to bound the minimum length of such word u but first we explain why words sending x beyond U exist. Since the automaton $\mathcal{A}$ is synchronizing and its underlying digraph is strongly connected (as the digraph is Eulerian), there exists a word $w \in A^*$ such that $Q \cdot w \subseteq S$—one can first synchronize $\mathcal{A}$ to a state q and then move q into S by applying a word that labels a path from q to a state in S. Then

There is a bug in [34] in this place

$$\varphi_w(x) = \varphi_w([S] - \frac{|S|}{n}[Q]) = \varphi_w([S]) - \frac{|S|}{n}\varphi_w([Q]) = (1 - \frac{|S|}{n})[Q] \neq 0.$$

Now consider the chain of subspaces $U_0 \subseteq U_1 \subseteq \dots$, where U_j is spanned by all vectors of form $\varphi_w(x)$ with $|w| \leqslant j$. Clearly, if $U_{j+1} = U_j$ for some ℓ then $\varphi_a(U_j) \subseteq U_j$ for all $a \in A$ whence $U_i = U_j$ for every $i \geqslant j$. Let ℓ be the least number such that $\varphi_u(x) \notin U$ for some word u of length ℓ, that is, the smallest ℓ such that $U_\ell \nsubseteq U$. Then in the chain $U_0 \subseteq U_1 \subseteq \dots \subseteq U_\ell$ all inclusions are strict whence

$$1 = \dim U_0 < \dim U_1 < \dots < \dim U_{\ell-1} < \dim U_\ell$$

and, in particular, $\dim U_{\ell-1} \geqslant \ell$. But by our choice of ℓ we have $U_{\ell-1} \subseteq U$ whence $\dim U_{\ell-1} \leqslant \dim U$. Since U is the orthogonal complement of a 1-dimensional subspace, $\dim U = n - 1$, and we conclude that $\ell \leqslant n - 1$.

As shown in the first paragraph of the proof, the above implies that for every proper subset $S \subset Q$, there exists a word $u \in A^*$ of length at most $n-1$ such that $|Su^{-1}| > |S|$. Then Algorithm 2 run on $\mathcal{A}$ returns a reset word of length at most $(n-2)(n-1)+1$. □

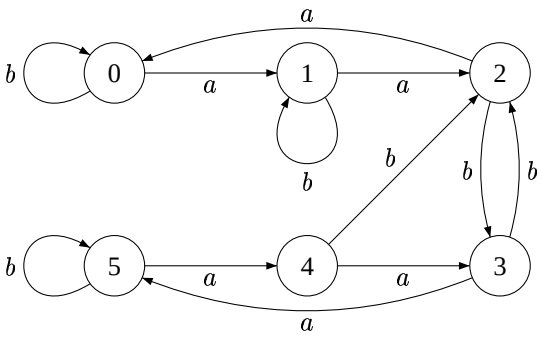
KV:Extension

We mention in passing that the upper bound provided by Theorem 3.5 is far from being tight. So far experiments have discovered no n-state synchronizing automaton $(n \geqslant 4)$ with Eulerian underlying digraph and reset threshold that would exceed $\lfloor \frac{n^2-5}{2} \rfloor$.

KV:thm:eulerian

Return to our discussion of extensibility. Even though the approach to the Černý conjecture via 1-extensibility has proved to be productive in several special cases, it cannot resolve the general case because there exist synchronizing automata that are not 1-extensible. The first example here was the 6-state automaton $\mathcal{K}_6$ discovered by Kari [33], see Figure 10. This automaton is synchronizing with reset threshold 25, the shortest reset word being $ba(ab)^3a^2b(ba)^3ab(ba^2)ab$. Kari found $\mathcal{K}_6$ as a counter example to a generalized form of the Černý conjecture proposed in Pin's thesis [45] but the automaton is remarkable in several other respects. In particular, one can verify that no word v of

KV:fig:kari



KV:fig:Kari

Figure 10. Kari's automaton $\mathcal{K}_6$

length 6 or 7 is such that the full pre-image of the set $\{2,3,4,5\}$ under the action of v has more than 4 elements.

Recently Berlinkov [7] has constructed a series of synchronizing automata that for each $\alpha < 2$ contains an automaton that is not α-extensible. The question of whether or not all synchronizing automata are 2-extensible remains open. 2-extensibility (and thus—by Proposition 3.4—a quadratic in the state number upper bound for the reset threshold) has been established for several classes of synchronizing automata by Rystsov [56, 57, 58].

Recently a slightly relaxed version of 2-extensibility has been verified by Béal, Berlinkov and Perrin [5, 4] for the important class of the so-called one-cluster automata. A DFA $\mathcal{A} = (Q, A)$ is called *one-cluster* if there exist $a \in A$ and $q \in Q$ such that each state in Q is connected to q by a path with all edges labelled by a. (For instance, the automata $\mathcal{C}_n$ and $\mathcal{W}_n$ shown in Figure 3 are one-cluster while Kari's automaton $\mathcal{K}_6$ shown in Figure 10, is not. A mass example of one-cluster automata is provided by the decoders of finite maximal prefix codes discussed in Section 1.) Let C be the set of all states q with the latter property. Clearly, a acts on C as a cyclic permutation so we refer to C as the *a-cycle*. It is easy to see that $Q \cdot a^{|Q|-|C|} = C$, and one can modify Algorithm 2 as follows.

In [5, 4] it has been shown that the length of each word v appended by the main loop of Algorithm 3 does not exceed $2|Q|$, and this clearly implies a quadratic in $|Q|$ upper bound on the reset threshold for one-cluster synchronizing automata. A similar result has been obtained by Carpi and D'Alessandro [12]. Steinberg [64, 65] has generalized the above approach and slightly improved the upper bound. Namely, Steinberg has proved that a one-cluster synchronizing automaton with n states has a reset word of length at most $2n^2 - 9n + 14$. He also has verified the Černý conjecture for one-cluster synchronizing automata with a-cycles of prime cardinality.

RELATIVEEXTENSTION$(\mathcal{A}, C, a)$

1: $w \leftarrow \varepsilon$ ▷ Initializing the current word
2: $P \leftarrow \{q\}$ where $q \in C$ ▷ Initializing the current set
3: **while** $|P| < |C|$ **do**
4: **if** $|Pu^{-1} \cap C| \leqslant |P|$ for all $u \in A^*$ **then**
5: **return** Failure
6: **else**
7: take a word $v \in A^*$ 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** $a^{|Q|-|C|}w$

:One-cluster

Algorithm 3. Modified extension algorithm for a one-cluster automaton $\mathcal{A} = (Q, A)$ with a-cycle C

4 The road coloring problem

KV:sec:rcp

5 Related work

:sec:related

References

[1] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006. 9

[2] D. Ananichev, V. Gusev, and M. Volkov. Slowly synchronizing automata and digraphs. In P. Hliněný and A. Kučera, editors, *Mathematical Foundations of Computer Science*, volume 6281 of *Lecture Notes in Comput. Sci.*, pages 55–64. Springer-Verlag, 2010. 10

[3] W. R. Ashby. *An introduction to cybernetics.* Chapman & Hall, 1956. 2

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

[5] M.-P. Béal and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. In V. Diekert and D. Nowotka, editors, *Developments in Language Theory*, Lecture Notes in Comput. Sci., pages 81–90. Springer-Verlag, 2009. 18

[6] M. Berlinkov. Approximating the minimum length of synchronizing words is hard. In F. Ablayev and E. W. Mayr, editors, *Computer Science in Russia*, volume 6072 of *Lecture Notes in Comput. Sci.*, pages 37–47. Springer-Verlag, 2010. 9

[7] M. Berlinkov. On a conjecture by Carpi and D'Alessandro. In Y. Gao, H. Lu, S. Seki, and S. Yu, editors, *Developments in Language Theory*, volume 6224 of *Lecture Notes in Comput. Sci.*, pages 66–75. Springer-Verlag, 2010. 18

[8] J. Berstel, D. Perrin, and C. Reutenauer. *Codes and automata.* Number 129 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2009. 2

[9] S. Bogdanović, B. Imreh, M. Ćirić, and T. Petković. Directable automata and their generalizations: a survey. *Novi Sad J. Math.*, 29(2):29–69, 1999. 5

[10] V. Boppana, S. Rajan, K. Takayama, and M. Fujita. Model checking based on sequential ATPG. In *Computer Aided Verification*, volume 1622 of *Lecture Notes in Comput. Sci.*, pages 418–430. Springer-Verlag, 1999. 2

[11] R. M. Capocelli, L. Gargano, and U. Vaccaro. On the characterization of statistically synchronizable variable-length codes. *IEEE Transactions on Information Theory*, 34(4):817–825, 1988. 3

[12] A. Carpi and F. D'Alessandro. The synchronization problem for locally strongly transitive automata. In R. Královic and D. Niwinski, editors, *Mathematical Foundations of Computer Science*, volume 5734 of *Lecture Notes in Computer Science*, pages 211–222. Springer-Verlag, 2009. 18

[13] J. Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fyzikalny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. (in Slovak). 2, 6, 10, 11

[14] J. Černý, A. Pirická, and B. Rosenauerová. On directable automata. *Kybernetika*, 7(4):289–298, 1971. 11

[15] Y.-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algoritmica*, 14:367–397, 1995. 4

[16] H. Cho, S.-W. Jeong, F. Somenzi, and C. Pixley. Synchronizing sequences and symbolic traversal techniques in test generation. *J. Electronic Testing*, 4:19–31, 1993. 2

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press and McGraw-Hill, 2001. 6

[18] F. M. Dekking. The spectrum of dynamical systems arising from substitutions of constant length. *Z. Wahrsch. Verw. Gebiete*, 41:221–239, 1978. 4

[19] L. Dubuc. Sur le automates circulaires et la conjecture de Černý. *RAIRO Inform. Théor. App.*, 32:21–34, 1998. (in French). 16

[20] D. Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19:500–510, 1990. 4, 7, 10

[21] M. A. Fischler and M. Tannenbaum. Synchronizing and representation problems for sequential machines with masked outputs. In *Proc. 11th Annual Symp. Foundations Comput. Sci*, pages 97–103. IEEE Press, 1970. 10, 13

[22] P. Frankl. An extremal problem for two families of sets. *European J. Combinatorics*, 3:125–127, 1982. 12, 13

[23] D. Frettlöh and B. Sing. Computing modular coincidences for substitution tilings and point sets. *Discrete Comput. Geom.*, 37:381–407, 2007. 10

[24] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, 1979. 7

[25] P. Gawrychowski. Complexity of shortest synchronizing word. Private communiction, 2008. 9

[26] M. Gerbush and B. Heeringa. Approximating minimum reset sequences. In M. Domaratzki and K. Salomaa, editors, *Implementation and Application of Automata*, volume 6482 of *Lecture Notes in Comput. Sci.*, pages 154–162. Springer-Verlag, 2011. 9

601 [27] A. Gill. State-identification experiments in finite automata. *Inform. Control*, 4(2-3):132–154,
602 1961. 2

603 [28] S. Ginsburg. On the length of the smallest uniform experiment which distinguishes the termi-
604 nal states of a machine. *J. Assoc. Comput. Mach.*, 5:266–280, 1958. 2

605 [29] K. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993. 4

606 [30] P. Goralčik and V. Koubek. Rank problems for composite transformations. *Internat. J. Algebra*
607 *Comput.*, 5:309–316, 1995. 7

608 [31] F. C. Hennie. Fault detecting experiments for sequential circuits. In *Switching Circuit Theory*
609 *and Logical Design*, pages 95–110. IEEE Press, 1964. 2

610 [32] T. Jiang and M. Li. On the approximation of shortest common supersequences and longest
611 common subsequences. *SIAM J. Comput.*, 24(5):1122–1139, 1995. 9

612 [33] J. Kari. A counter example to a conjecture concerning synchronizing words in finite automata.
613 *Bull. European Assoc. Theor. Comput. Sci.*, 73:146, 2001. 17

614 [34] J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theoret. Comput. Sci.*, 295:223–
615 232, 2003. 16, 17

616 [35] A. A. Klyachko, I. K. Rystsov, and M. A. Spivak. An extremal combinatorial problem associ-
617 ated with the bound of the length of a synchronizing word in an automaton. *Cybernetics and*
618 *System Analysis*, 23(2):165–171, 1987. translated from Kibernetika, No. 2, 1987, pp. 16–20,
619 25. 13

620 [36] Z. Kohavi and J. Winograd. Bounds on the length of synchronizing sequences and the or-
621 der of information losslessness. In Z. Kohavi and A. Paz, editors, *Theory of Machines and*
622 *Computations*, pages 197–206. Academic Press, 1971. 13

623 [37] Z. Kohavi and J. Winograd. Establishing certain bounds concerning finite automata. *J. Com-*
624 *put. System Sci.*, 7(3):288–299, 1973. 13

625 [38] A. E. Laemmel and B. Rudner. Study of the application of coding theory. Technical Report
626 PIBEP-69-034, Dept. Electrophysics, Polytechnic Inst. Brooklyn, Farmingdale, N.Y., 1969.
627 10

628 [39] E. F. Moore. Gedanken experiments on sequential machines. In C. E. Shannon and J. Mc-
629 Carthy, editors, *Automata Studies*, pages 129–153. Princeton Universty Press, 1956. 2

630 [40] B. K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *Proc.*
631 *27th Annual Symp. Foundations Comput. Sci.*, pages 132–142. IEEE Press, 1986. 4

632 [41] B. K. Natarajan. Some paradigms for the automated design of parts feeders. *Internat. J.*
633 *Robotics Research*, 8(6):89–109, 1989. 4

634 [42] J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In
635 P. Hliněný and A. Kučera, editors, *Mathematical Foundations of Computer Science*, number
636 6281 in Lecture Notes in Comput. Sci., pages 568–579. Springer-Verlag, 2010. 9

637 [43] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. 7

638 [44] C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of com-
639 plexity). *J. Comput. System Sci.*, 28(2):244–259, 1984. 9

640 [45] J.-E. Pin. *Le problème de la synchronisation et la conjecture de Černý*. Thèse de 3ème cycle,
641 Université Paris VI, 1978. 17

642 [46] J.-E. Pin. Sur un cas particulier de la conjecture de Černý. In *Proc. 5th Colloq. on Automata,*
643 *Languages, and Programming (ICALP)*, volume 62 of *Lecture Notes in Comput. Sci.*, pages
644 345–352. Springer-Verlag, 1978. (in French). 16

[47] J.-E. Pin. On two combinatorial problems arising from automata theory. *Ann. Disc. Math.*, 17:535–548, 1983. 13

[48] C. Pixley, S.-W. Jeong, and G. D. Hachtel. Exact calculation of synchronization sequences based on binary decision diagrams. In *Proc. 29th Design Automation Conf.*, pages 620–623. IEEE Press, 1992. 9

[49] N. Pytheas Fogg. *Substitutions in dynamics, arithmetics and combinatorics*, volume 1794 of *Lecture Notes in Mathematics*. Springer-Verlag, 2002. Edited by V. Berthé, S. Ferenczi, C. Mauduit and A. Siegel. 4

[50] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3(2):114–125, 1959. 5

[51] J. L. Ramírez Alfonsín. *The diophantine Frobenius problem.* Oxford University Press, 2005. 11

[52] J.-K. Rho, F. Somenzi, and C. Pixley. Minimum length synchronizing sequences of finite state machine. In *Proc. 30th Design Automation Conf.*, pages 463–468. ACM, 1993. 7

[53] A. Roman. Genetic algorithm for synchronization. In A. Dediu, A. Ionescu, and C. Martín-Vide, editors, *Language and Automata Theory and Applications*, volume 5457 of *Lecture Notes in Comput. Sci.*, pages 684–695. Springer-Verlag, 2009. 9

[54] A. Roman. Synchronizing finite automata with short reset words. *Applied Mathematics and Computation*, 209(1):125–136, 2009. 9

[55] I. K. Rystsov. On minimizing length of synchronizing words for finite automata. In *Theory of Designing of Computing Systems*, pages 75–82. Institute of Cybernetics of Ukrainian Acad. Sci., 1980. (in Russian). 7

[56] I. K. Rystsov. Almost optimal bound of reccurent word length for regular automata. *Cybernetics and System Analysis*, 31:669–674, 1995. translated from Kibernetika i Sistemnyj Analiz, No. 5, 1995, pp. 40–48. 18

[57] I. K. Rystsov. Quasioptimal bound for the length of reset words for regular automata. *Acta Cybernetica*, 12:145–152, 1995. 18

[58] I. K. Rystsov. Reset words for automata with simple idempotents. *Cybernetics and System Analysis*, 36:339–344, 2000. translated from Kibernetika i Sistemnyj Analiz, No. 3, 2000, pp. 32–39. 18

[59] A. Salomaa. Composition sequences for functions over a finite domain. *Theoret. Comput. Sci.*, 292:263–281, 2003. 7

[60] W. Samotij. A note on the complexity of the problem of finding shortest synchronizing words. In *Proc. AutoMathA 2007, Automata: from Mathematics to Applications.* Univ. Palermo, 2007. (CD). 7, 8

[61] S. Sandberg. Homing and synchronizing sequences. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems*, volume 3472 of *Lecture Notes in Comput. Sci.*, pages 5–33. Springer-Verlag, 2005. 2, 7

[62] A. L. Selman. A taxonomy of complexity classes of functions. *J. Comput. System Sci.*, 42(1):357–381, 1994. 9

[63] P. H. Starke. Eine Bemerkung über homogene Experimente. *Elektronische Informationverarbeitung und Kybernetik*, 2:257–259, 1966. (in German). 11

[64] B. Steinberg. The averaging trick and the Černý conjecture. In Y. Gao, H. Lu, S. Seki, and S. Yu, editors, *Developments in Language Theory*, volume 6224 of *Lecture Notes in Comput. Sci.*, pages 423–431. Springer-Verlag, 2010. 18

[65] B. Steinberg. The Černý conjecture for one-cluster automata with prime length cycle. *CoRR*, abs/1005.1835, 2010. 18

[66] A. Trahtman. An efficient algorithm finds noticeable trends and examples concerning the Černý conjecture. In R. Královič and P. Urzyczyn, editors, *31st Int. Symp. Math. Foundations of Comput. Sci.*, volume 4162 of *Lecture Notes in Comput. Sci.*, pages 789–800. Springer-Verlag, 2006. 7, 9

[67] M. Volkov. Synchronizing automata and the Černý conjecture. In C. Martín-Vide, F. Otto, and H. Fernau, editors, *Language and Automata Theory and Applications*, volume 5196 of *Lecture Notes in Comput. Sci.*, pages 11–27. Springer-Verlag, 2008. 2, 4

[68] H. Wielandt. Unzerlegbare, nicht negative Matrizen. *Math. Z.*, 52:642–648, 1950. (in German). 10

[69] S. Yang. Logic synthesis and optimization benchmarks. Technical Report User Guide Version 3.0, Microelectronics Center of North Carolina, Research Triangle Park, NC, 1991. 9

Index