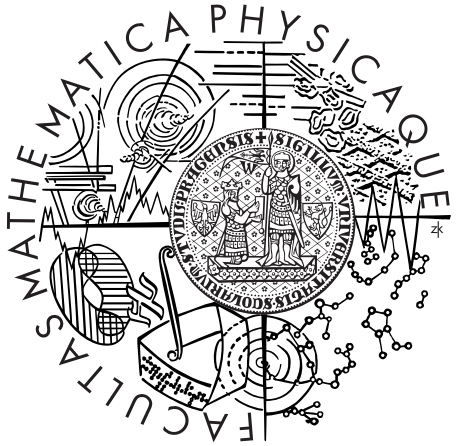
Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS



Vojtěch Vorel

Synchronization, Road Coloring, and Jumps in Finite Automata

Department of Theoretical Computer Science and Mathematical Logic

Supervisor: prof. RNDr. Václav Koubek, DrSc.

Study programme: Computer science

Specialization: Theoretical computer science

Prague 2015

I am grateful to professors Václav Koubek and Tomáš Dvořák for their longstanding support and professional advice. My thanks also go to Adam Roman, Marek Szykuła, Pavel Panteleev, François Gonze, Szabolsc Iván, Pavel Martyugin, František Mráz, Peter Černo, Petr Zemek and anonymous reviewers for sharing their ideas and knowledge.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, May 7, 2015

Název práce: Synchronizace, barvení cesty a skoky v konečných automatech

Autor: Vojtěch Vorel

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: prof. RNDr. Václav Koubek, DrSc.

Abstrakt: Práce shrnuje několik původních výsledků v teorii automatů a formálních jazyků. Studuje kombinatorické otázky a výpočetních úlohy z oblasti synchronizačních slov a barvení cesty. Kromě toho se zabývá skokovými konečnými automaty a souvisejícími typy přepisovacích systémů.

Klíčová slvoa: Synchronizace, Synchronizační slova, Barvení cesty, Skokové konečné automaty, Restartovací automaty

Title: Synchronization, Road Coloring, and Jumps in Finite Automata

Author: Vojtěch Vorel

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: prof. RNDr. Václav Koubek, DrSc.

Abstract: Multiple original results in the theory of automata and formal languages are presented, dealing mainly with combinatorial problems and complexity questions related to reset words and road coloring. The other results concern jumping finite automata and related types of rewriting systems.

Keywords: Synchronization, Reset Word, Road Coloring, Jumping Finite Automata, Restarting Automata

Contents

Preface

The thesis collects multiple original results in the theory of automata and formal languages. Most of the results deal with synchronization and road coloring - research fields that are introduced and surveyed in Chapter 1. Besides of that we present contributions to the study of languages accepted by jumping finite automata and clearing restarting automata. The main text is organized as follows:

- Chapter 2 deals with lower bounds of synchronization thresholds in multiple types of finite automata:
  - First, we give new lower bounds of the synchronization threshold in partial finite automata and the subset synchronization threshold in deterministic finite automata, both restricted to binary alphabets. Besides of the main proof we provide general formulations of auxiliary facts and discuss consequences and reformulations of the result. The result, presented [103] at the conference *AFL 2014* (Szeged, Hungary), answers a question asked by Martyugin, 2013 [62]. An extended paper was submitted to a journal.
  - Second, we present attempts to raise the known lower bound of the synchronization threshold in deterministic automata with sink states. We give new isolated examples of automata that exceed the currently best lower bound by Martyugin, 2008 [57] and formulate a hypothesis that involves an infinite series of automata based on the examples.
- Chapter 3 consists of two results about a basic computational task (SYN) concerning synchronization of deterministic finite automata:
  - First, we prove that SYN does not have a polynomial kernel if parameterized by the number of states unless polynomial hierarchy collapses. This fills the only remaining gap in a research of Fernau, Heggernes, and Villanger, 2013 [34] (in the latest version of this article [35], our result is already cited). A paper [105] containing the proof was published in *Discrete Mathematics and Theoretical Computer Science*.
  - Second, we prove NP-completeness of SYN restricted to Eulerian automata with binary alphabets, as it was conjectured by Martyugin, 2011 [60]. The proof was presented [102] at the conference *LATA 2014* (Madrid, Spain). An extended paper was submitted to a journal.
- Chapter 4 studies two related questions about road coloring. These results come partially from a collaboration with Adam Roman (Jagiellonian University in Krakow, Poland):
  - First, we give a multi-parameter analysis of the basic computational problem called SRCP. This consists mainly of studying a scale of various restrictions and finishes a work that was started by Roman and Drewienkowski [78, 79]. The results are contained in the above-mentioned journal paper [105].
  - Second, we give a similar analysis with respect to slightly different computational problem called SRCW. The results are not complete - they leave much space for a further research.

They were presented [104] at the conference *LATA 2015* (Nice, France). An extended version was submitted to a journal.

- Chapter 5 quits the field of synchronization and studies the expressive power of two simple models for discontinuous analysis of strings: jumping finite automata and clearing restarting automata.
  - First, we complete the initial study of jumping finite automata, which was started in a former article of Meduna and Zemek [64, 65]. The open questions about basic closure properties are solved. Besides of that, we correct erroneous results presented in [64, 65]. Finally, we point out important relations between jumping finite automata and other models studied in the literature. An article presenting these results was submitted to a journal.
  - Second, we deal with clearing restarting automata, which is a class of contextual rewriting systems. We construct a clearing restarting automaton with two-letter contexts that accepts a language over a two-letter alphabet lying outside the class CFL, thus closing the study raised by Černo and Mráz, 2010 [25].
- In Chapter 6 we discuss further research in all the studied directions and formulate open problems that are closely related to the results of the thesis.

In the following table we give a concise listing of our contributions to the state of art. Diamonds mark sections that give closing answers to questions formulated in the literature:

	Topic	Ref.	Publ.
Sec. 2.1	Careful Synchronization threshold of binary PFA ♦	[62]	[103]
Sec. 2.2	Synchronization thresholds of DFA with sinks	[57]	-
Sec. 3.1	Polynomial Kernel of SYN ♦	[34]	[105]
Sec. 3.2	NP-Completess of SYN restricted to Eulerian automata ♦	[60]	[102]
Sec. 4.1	Complexity of SRCP with fixed parameters ♦	[78]	[105]
Sec. 4.2	Complexity of SRCW with singleton fixed set of words	[78]	[104]
Sec. 5.2	Closure properties of the class GJFA ♦	[65]	-
Sec. 5.3	A non-CFL binary language accepted by 2-cl-RA ♦	[25]	-

Some of the results were presented at the workshop *Černý's Conjecture and Optimization Problems* held in Opava, Czech Republic, 2014 and at a meeting of the *Group of Computational Complexity and Analysis of Algorithms* at University of Wrocław.

The research was supported by the Czech Science Foundation grant GA14-10799S and the GAUK grant No. 52215. The collaborative work with Adam Roman was supported also by the Polish Ministry of Science and Higher Education grant IP2012 052272.

Chapter 1

A Survey of Synchronization and Road Coloring

1.1 Motivation and History

The central idea of *synchronization* is very natural: For a given machine, we want to find an input sequence that gets the machine to a unique state, no matter in which state the machine initially was. Considering deterministic finite automata (DFA), we get a clear notion of *reset words*. A machine that admits a reset word is said to be *synchronizing*. Reset words for DFA have been studied since the early times of automata theory. First related studies considered input sequences that do not necessarily leave the machine in a unique state, but it need to be possible to infer the resulting state from the observed output [39, 66]. Reset words, as they are informally defined above, were introduced in 1964 by Černý in [26]. In the 1960's, the subject was discussed also by Starke [87] and others, using various systems of terminology.

Most of the research of synchronization of DFA has concerned minimal length of reset words. For a given synchronizing automaton, one is interested in its shortest reset words. Černý [26] presented a series of synchronizing n-state automata that required reset words of length at least $(n-1)^2$. In the same year the *Černý conjecture* (that appeared in a subsequent article [27]) was formulated, saying that each synchronizing n-state automaton has a reset word of length at most $(n-1)^2$. Since that, the research of the maximum length of shortest reset words, taken over n-state DFA, has become a classical topic of automata theory and the Černý conjecture has become a notorious open problem. So far, the best upper bound of the threshold is $\frac{n^3-n}{6}$ [73].

However, the upper bound $(n-1)^2$ or lower has been confirmed for various special classes of DFA, see Section 1.3.2. Some of the results consist of non-trivial application of advanced tools from linear algebra and the theory of semigroups. Several generalizations of the Černý conjecture have been studied, both for DFA and for more general settings, see Section 1.4. For instance, in *subset synchronization* one looks for a word that puts a DFA to one particular state, assuming that initially the DFA was in some state from a given subset of states. Here we are interested in the worst cases taken over all DFA and all their subsets of states. Most synchronization problems can be formulated as essential problems about transformation monoids and matrix monoids.

Several fields of mathematics and engineering deal with synchronization:

- Classical applications include model-based testing [28] and equivalence checking [74] of sequential circuits. Reset words are used to generate effective test patterns. It is characteristic for these applications that the state space of the arising finite automata are exponential in the number of components that form the hardware under test. Thus, basic

computational tasks become hardly feasible and the way of representing the transition function plays a key role (e.g., binary decision diagrams are used).

- In automated assembling, so-called *part orienters* are used. Such device consists typically of a conveyor belt carrying parts that may lie in various positions. One needs to construct a device that gets a particle to a well-defined position. However, the device should be as simple and cheap as possible, preferably suitable stable obstacles should be used to push the particle while it is carried by the conveyor belt. When such obstacles are designed, one can see its effect on various positions as transitions. In fact, the resulting scheme corresponds to a DFA and a reliable part orienter corresponds to a reset word. For details, see e.g. [68].

- In design of control systems modeled by finite automata, one may take into account the possibility of passing to a wrong state during reading a correct input. For synchronizing automata, the *noise-resistance*, i.e. probability of getting back to the right state, is essentially greater than in general automata and on random inputs it tends to 1 as the input length goes to infinity [29].

- In information theory, a *finite-state information source* [96] corresponds formally to a partial finite automaton, but the transition labels correspond to the output. Reset words, if appropriately generalized to partial automata, are output sequences that uniquely determine the current state of the information source, if observed.

- Reset words (also known as *constants*) play a role in studying non-classical representations of formal languages. See [21] for such a study, motivated by biomolecular processes

Each DFA has a unique underlying graph, which is a directed multigraph where each vertex has exactly $|\Sigma|$ outgoing edges, Σ denoting the alphabet of the DFA. On the other other hand, directed multigraphs with constant out-degrees can be turned into DFA by coloring the edges with letters. The research of such colorings was originally motivated by symbolic dynamic - Adler, Goodwyn, and Weiss [1] pointed out in 1977 that certain properties of Markov shifts depend on whether the edges of a corresponding directed multigraph can be colored such that a synchronizing DFA arises (see also [14] for a modern survey). They raised the *Road coloring problem*, a hypothesis claiming that a trivial necessary condition is also sufficient for a directed multigraph to admit such a coloring. The hypothesis was confirmed in 2008 by Trahtman [91] and is known as the *Road coloring theorem*, see Section 1.3.4.

In the reminder of Chapter 1 we give key definitions and present a survey of former results in the field. For a more concise (but not fully up-to-date) overview, see [82] or [99].

1.2 Key Definitions

The symbol $\mathbb{N}$ denotes the set $\{0, 1, 2, \dots\}$ of nonnegative integers. For sets A, B, by $A \times B$ we denote the Cartesian product of A and B. By 2^A we denote the set of subsets of A. If x is a real number, $\lfloor x \rfloor$ and $\lceil x \rceil$ denote the largest integer not greater than x and the smallest integer not less than x, respectively. If m, n are positive integers, $n \bmod m$ denotes the remainder after division of n by m.

1.2.1 Finite Automata, Graphs, and Finite Functions

Definition 1.1. A triple $A = (Q, \Sigma, \delta)$, where Q is a finite set of *states* and Σ is a finite *alphabet*, is:

1. a *deterministic finite automaton* (*DFA*) if δ is a total function from $Q \times \Sigma$ to Q,

2. a *partial finite automaton* (*PFA*) if δ is a partial function from $Q \times \Sigma$ to Q,

3. a *non-deterministic finite automaton* (*NFA*) if δ is a total function from $Q \times \Sigma$ to 2^Q (i.e. to the set of subsets of Q).

Clearly, each deterministic automaton is a partial automaton. Partial automata can be equivalently seen as non-deterministic automata with $|\delta(s,x)| \leq 1$ for each $s \in Q$ and $x \in X$. Note that we consider machines without any form of output and without initial or final states.
We extend transition functions such that they operate also on sets of states and whole words over the alphabet. We set

$$\delta(S,x) = \{\delta(s,x) \mid s \in S \text{ and } \delta(s,x) \text{ is defined}\}$$

in PFA and

$$\delta(S,x) = \bigcup_{s \in S} \delta(s,x)$$

in NFA for each $S \subseteq Q, x \in \Sigma$. In both PFA and NFA we set

$$\begin{aligned} \delta(S,wx) &= \delta(\delta(S,w),x), \\ \delta(s,wx) &= \delta(\delta(s,w),x), \end{aligned}$$

for each $S \subseteq Q, s \in Q, w \in \Sigma^+, x \in \Sigma$. Moreover, having a PFA $A = (Q,\Sigma,\delta)$, for each $S \subseteq Q, s \in Q$ and $w \in \Sigma^\star$ we denote

$$\begin{aligned} \delta^{-1}(S,w) &= \{r \in Q \mid \delta(r,w) \in S\}, \\ \delta^{-1}(s,w) &= \{r \in Q \mid \delta(r,w) = s\}. \end{aligned}$$

For a fixed automaton, a word $w \in \Sigma^\star$ is informally identified with the function $\delta(\_,w) : Q \to Q$. Thus, we speak e.g. about words that are permutations or constants, about their ranges and so on. We define some key notions about graphs, finite transformations, and automata:

Definition 1.2.

- A *directed graph* is a pair $G = (V,E)$ where V, E are finite sets and $E \subseteq V \times V$.
- A *directed multigraph* is a tuple $G = (V,E,\mathbf{s},\mathbf{t})$ where V, E are finite sets and $\mathbf{s},\mathbf{t}$ are functions from E to V.
- In a directed multigraph $G = (V,E,\mathbf{s},\mathbf{t})$, $\mathbf{s}(e)$ and $\mathbf{t}(e)$ are the *start* and the *target* of $e \in E$, respectively.
- In a directed multigraph $G = (V,E,s,t)$, the *out-degree* and the *in-degree* of $v \in V$ is $|\{e \in E \mid \mathbf{s}(e) = v\}|$ and $|\{e \in E \mid \mathbf{t}(e) = v\}|$ respectively.
- For a directed multigraph $G = (V,E,s,t)$ and $r, s \in V$, the $\mathrm{termd}_G(r,s)$ denotes the length of shortest directed paths from r to s in G.

Definition 1.3. The *underlying graph* of a DFA $A = (Q,\Sigma,\delta)$ is the directed multigraph $G_A = (Q, Q \times \Sigma, \mathbf{s}, \mathbf{t})$, where

$$\begin{aligned} \mathbf{s}(r,x) &= r, \\ \mathbf{t}(r,x) &= \delta(r,x) \end{aligned}$$

for each $(r,x) \in Q \times \Sigma$.

Definition 1.4. Let Q be an n-element set and let $f : Q \to Q$ be a partial function.

- The *domain* of f is $\mathrm{dom}(f) = \{s \in Q \mid f(s) \text{ is defined}\}$.
- The *range* of f is $\mathrm{rng}(f) = \{f(s) \mid s \in \mathrm{dom}(f)\}$.
- The *rank* of f is the size of $\mathrm{rng}(f)$.
- We say that f is *idempotent* if $f(s) = s$ for each $s \in \mathrm{rng}(f)$.
- The *graph* of f is the directed graph $G_f = (Q, E)$ with $E = \{(s, f(s)) \mid s \in \mathrm{dom}(f)\}$.
- A *cycle* of f is the set of vertices of a strongly connected component in G_f.
- A *cluster* of f is the set of vertices of a weakly connected component in G_f.
- We say that f *preserves* a binary relation $\preceq$ on Q if $s_1 \preceq s_2$ implies $f(s_1) \preceq f(s_2)$ for each $s_1, s_2 \in Q$.

Definition 1.5. Let us introduce some special classes of PFA and DFA together with corresponding notation:

- A PFA $A = (Q, \Sigma, \delta)$ is *strongly connected* (*s.c.*)[1] if its underlying graph is strongly connected. The class of strongly connected PFA is denoted by $\mathcal{SC}$.
- A PFA $A = (Q, \Sigma, \delta)$ is *incomplete* if δ is not total.
- A PFA $A = (Q, \Sigma, \delta)$ is *circular* if some $x \in \Sigma$ is a total cyclic permutation. The class of circular PFA is denoted by $\mathcal{CY}$.
- A DFA $A = (Q, \Sigma, \delta)$ is *one-cluster* if some $x \in \Sigma$ has exactly one cluster. The class of one-cluster DFA is denoted by $\mathcal{OC}$.
- A DFA $A = (Q, \Sigma, \delta)$ is *aperiodic*[2] if for each $w \in \Sigma^\star$ there is $k \geq 0$ such that $\delta\big(s, w^k\big) = \delta\big(s, w^{k+1}\big)$ for each $s \in Q$. The class of aperiodic DFA is denoted by $\mathcal{AP}$.
- A DFA $A = (Q, \Sigma, \delta)$ is *monotonic* if there is a linear order $\preceq$ on Q that is preserved by each $x \in \Sigma$. The class of monotonic DFA is denoted by $\mathcal{MO}$.
- A DFA $A = (Q, \Sigma, \delta)$ has a *sink state* $q_0 \in Q$ (also known as a *zero state*) if $\delta(q_0, x) = q_0$ for each $x \in \Sigma$. The class of DFA having a sink state is denoted by $\mathcal{Z}$.
- A state $s \in Q$ in a PFA $A = (Q, \Sigma, \delta)$ is a *merging state* if $\big|\delta^{-1}(s, x)\big| \geq 2$ for some $x \in \Sigma$.

For each integer $k \geq 1$, by $\mathcal{AL}_k$ we denote the class of PFA with alphabets of size k. A PFA is called *unary* or *binary* if it lies in $\mathcal{AL}_1$ or $\mathcal{AL}_2$ respectively.

1.2.2 Synchronization

There are several different interpretations of the concept of synchronization. In the case of DFA and PFA, the classical variant says that we are given an automaton with known transition function but unknown initial state. However, it may be known that the initial state lies in certain subset of states, i.e. the *initial uncertainty* is specified. Then we should find an input word that makes the automaton switch to a target state that is common for all possible initial states - no uncertainty is left.
Here we introduce a system of notation, which comes from former literature and is compatible with all the variants of synchronization we study. Thus the most classical notions are presented as special cases of generalized notions.

[1] Also known as *transitive*.
[2] Also known as *counter-free*.

Definition 1.6. A word $w \in \Sigma^\star$ is a *careful reset word* of a PFA $A = (Q, \Sigma, \delta)$ if there is $r \in Q$ such that

$$\delta(s, w) = r$$

for each $s \in Q$. If there is such $w \in \Sigma^\star$, we say that A is *carefully synchronizing*. If A is a DFA, we just say that such w is a *reset word* and A is *synchronizing*.

Definition 1.7. A word $w \in \Sigma^\star$ is a *careful reset word of a subset* $S \subseteq Q$ in a PFA $A = (Q, \Sigma, \delta)$ if there is $r \in Q$ such that

$$\delta(s, w) = r$$

for each $s \in S$. If there is such $w \in \Sigma^\star$, we say that the subset S is *carefully synchronizable*. If A is a DFA, we just say that such w is a *reset word* of S and S is *synchronizable*.

Definition 1.8. For each PFA $A = (Q, \Sigma, \delta)$ and each carefully synchronizable $S \subseteq Q$ we denote:

$$\begin{aligned} \mathrm{car}(A) &= \min\{|w| \mid w \text{ is a careful reset word of } A\}, \\ \mathrm{csub}(A, S) &= \min\{|w| \mid w \text{ is a careful reset word of } S \text{ in } A\}. \end{aligned}$$

If A is a DFA, we write $\mathrm{C}(A)$ and $\mathrm{sub}(A, S)$ instead of $\mathrm{car}(A)$ and $\mathrm{csub}(A, S)$.

Let us define several functions that describe the *worst cases* of minimum lengths of reset words depending on the number of states of an automaton. Such functions are informally called *synchronization thresholds*. As we describe later, there is a rich research that aims to obtain upper and lower bounds of these functions.

Definition 1.9. For each $n \geq 1$ we denote:

$$\begin{aligned} \mathrm{C}_n &= \max\{\mathrm{C}(A) \mid A \text{ is a DFA with at most } n \text{ states}\}, \\ \mathrm{sub}_n &= \max\{\mathrm{sub}(A, S) \mid A = (Q, \Sigma, \delta) \text{ is a DFA with at most } n \text{ states and } S \subseteq Q\}, \\ \mathrm{car}_n &= \max\{\mathrm{car}(A) \mid A \text{ is a PFA with at most } n \text{ states}\}, \\ \mathrm{csub}_n &= \max\{\mathrm{csub}(A, S) \mid A = (Q, \Sigma, \delta) \text{ is a PFA with at most } n \text{ states and } S \subseteq Q\}. \end{aligned}$$

Thus, the values C_n, sub_n, car_n, and csub_n express the worst cases among all n-state automata and subsets of their states. The values satisfy the following trivial inequalities, where $\to$ stands for *less or equal*:

$$\begin{array}{ccc} \mathrm{C}_n & \to & \mathrm{car}_n \\ \downarrow & & \downarrow \\ \mathrm{sub}_n & \to & \mathrm{csub}_n \end{array}$$

The following definition formalizes the notation of synchronization thresholds with respect to special classes of automata.

Definition 1.10. If $\mathcal{M}$ is a class of PFA and $n \geq 1$, we denote

$$\begin{aligned} \mathrm{C}_n^{\mathcal{M}} &= \max\{\mathrm{C}(A) \mid A \in \mathcal{M} \text{ is a DFA with at most } n \text{ states}\}, \\ \mathrm{sub}_n^{\mathcal{M}} &= \max\{\mathrm{sub}(A, S) \mid A = (Q, \Sigma, \delta) \in \mathcal{M} \text{ is a DFA with at most } n \text{ states}, S \subseteq Q\}, \\ \mathrm{car}_n^{\mathcal{M}} &= \max\{\mathrm{car}(A) \mid A \in \mathcal{M} \text{ is a PFA with at most } n \text{ states}\}, \\ \mathrm{csub}_n^{\mathcal{M}} &= \max\{\mathrm{csub}(A, S) \mid A = (Q, \Sigma, \delta) \in \mathcal{M} \text{ is a PFA with at most } n \text{ states}, S \subseteq Q\}. \end{aligned}$$

In addition, if $\mathcal{M}$ is a class of pairs $\langle A, S\rangle$ where A is a PFA and S is a subset of its states, we

denote

$$\begin{aligned} \mathrm{sub}_n^{\mathcal{M}} &= \max\{\mathrm{sub}(A,S) \mid A \text{ is a DFA with at most } n \text{ states and } \langle A,S\rangle \in \mathcal{M}\}, \\ \mathrm{csub}_n^{\mathcal{M}} &= \max\{\mathrm{csub}(A,S) \mid A \text{ is a PFA with at most } n \text{ states and } \langle A,S\rangle \in \mathcal{M}\}. \end{aligned}$$

1.2.3 Road Coloring

Road coloring refers to coloring edges of directed multigraph in order to obtain representations of DFA. The study is motivated mainly by symbolic dynamics. It was initially pointed out in [1] that a suitable coloring of a directed multigraph corresponds to an almost injective mapping from a subshift of finite type to a full shift. See also [14] for a modern survey.

Definition 1.11. A DFA $A = (Q, \Sigma, \delta)$ is a *coloring* of a directed multigraph G if G isomorphic to the underlying graph of A. We also say that the function δ is a coloring of G.

Definition 1.12. A directed multigraph is:

1. an *aperiodic graph* if the lengths of its cycles do not have any nontrivial common divisor,
2. an *admissible graph* if it is aperiodic and all its out-degrees are equal,
3. a *road colorable graph* if it has a synchronizing coloring.

In [1] it was pointed out as an open problem (the *Road Coloring Problem*) whether each strongly connected admissible graph is road colorable. It was solved positively by Trahtman [91, 95] in 2008, see Theorem 1.26.

1.3 Former Results

1.3.1 A Lower Bound for DFA

In the seminal paper [26], Černý presented the infinite series C_n of automata defined as $C_n = (\{0,\ldots,n-1\},\{a,b\},\delta)$, where

$$\begin{aligned} \delta(s,a) &= (s+1) \bmod n, \\ \delta(s,b) &= \begin{cases} 1 & \text{if } s = 0, \\ s & \text{otherwise,} \end{cases} \end{aligned}$$

for each $s \in Q$, see Figures 1.1 and 1.2. This series is used as a witness in the proof of the following theorem, which is the higher known lower bound of both C_n and C_n

Theorem 1.13 ([26])**.** *For each $n \geq 1$ it holds that $\mathrm{C}_n \geq (n-1)^2$.*

Except for the series given by Černý, only isolated examples reaching the bound $\mathrm{C}(A) = (n-1)^2$ are known, see [77]. However, several infinite series that reach very high values were obtained from matrices with large exponents, see e.g. [9].
In the paper [27], Theorem 1.13 was turned into the following well known hypothesis:

Conjecture 1.14 (The Černý conjecture)**.** *For each $n \geq 1$ it holds that $\mathrm{C}_n = (n-1)^2$.*

1.3.2 Upper Bounds for DFA

The paper [26] presents only a trivial upper bound $\mathrm{C}_n \leq 2^n - n - 1$, but it turned out soon that it is not hard to establish cubic bounds (in [27], this is attributed to [86]). Since that, there were several improvements of the cubic bound. The following theorem expresses the current

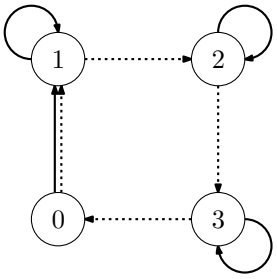


Figure 1.1: The Černý automaton C_4

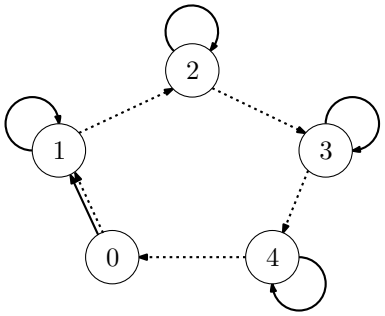


Figure 1.2: The Černý automaton C_5

best one (an improved upper bound published by Trahtman [93] in 2011 has turned out to be proved incorrectly [40]). The proof of the following theorem is based on a combinatorial result of Frankl [37] (see [99] for a history of this result):

Theorem 1.15 (Pin, 1983 [73])**.** *Each n-state synchronizing automaton has a reset word of length at most $\frac{n^3-n}{6}$.*

One of the key remarks dealing with synchronization of DFA is the following:

Lemma 1.16 (Černý, 1964 [26])**.** *A DFA $A = (Q, \Sigma, \delta)$ is synchronizing if and only if for each $r, s \in Q$ there is $w \in \Sigma^\star$ such that $\delta(r, w) = \delta(s, w)$.*

It turns out to be advantageous to think about synchronization also in a reversed way. We call a set $S \subseteq Q$ *m-extendable* if there is $w \in \Sigma^\star$ of length at most m such that $\left|\delta^{-1}(S, w)\right| > |S|$. Many upper bounds for special classes of automata (see below) are based on the following remark:

Lemma 1.17 (folklore)**.** *Let $A = (Q, \Sigma, \delta)$ be a PFA with $|Q| = n$. If each $S \subseteq Q$ is m-extendable, then $\mathrm{C}(A) \leq (n-2) \cdot m + 1$.*

Corollary 1.18 (folklore)**.** *Let $A = (Q, \Sigma, \delta)$ be a PFA with $|Q| = n$. If each $S \subseteq Q$ is n-extendable, then $\mathrm{C}(A) \leq (n-1)^2$.*

Studying inverse images of subsets is called *the extension method*. It turns out that Corollary 1.18 cannot be straightforwardly used to prove the Černý conjecture: for each n there is an n-state synchronizing automaton having a subset that is not $(2n-4)$-extandable [16].
Let us mention several groups of former results. In some of their proofs (mainly of the first three groups) the approach of Lemma 1.17 and Corollary 1.18 is widely used:

- The classes of circular automata has been intensively studied since it turned out that the Černý's series of circular automata (see [26]) provides the worst known case of n-state DFA for each n. Using a clever linear-algebraic approach, it was proven by Pin [72] that $\mathrm{C}_n^{\mathcal{CY}} \leq (n-1)^2$ whenever n is prime, Savický and Vaněček [83] then shown that $\mathrm{C}_n^{\mathcal{CY}} \leq (n-1)^2 + (n-2)^2$ for each n, and finally Dubuc [31] proved the upper bound $(n-1)^2$ for the whole class $\mathcal{CY}$. Several directions of further generalization of this result arose. First, Béal, Perrin, and Berlinkov [13] provided a quadratic upper bound for $\mathrm{C}_n^{\mathcal{OC}}$. Second, Steinberg [89] reached the upper bound $(n-1)^2$ of $\mathrm{C}_n^{\mathcal{OC}}$ for prime n. Third, Berlinkov [18] proved a quadratic upper bound for *quasi-one-cluster* automata, a strong generalization of one-cluster automata. Let us mention also the papers [16] and [88], which both present generalizing ideas concerning various results including most of the the above-mentioned ones.

- Kari [51] (see also [10]) verified Černý conjecture for Eulerian automata, using linear-algebraic methods as well. After that, the result was generalized to *pseudo-Eulerian* [88] and *quasi-Eulerian* [17] automata by combining former methods with a kind of probabilistic approach. As for lower bounds, there is a series [42] witnessing that $\mathrm{C}_n^{\mathcal{EU}} \geq \frac{n^2-3n+4}{2}$ for $n \geq 5$.

- There is a spectrum of DFA classes that are defined by preserving relations on states. For the (already defined) class of monotonic automata there is a tight upper bound $\mathrm{C}_n^{\mathcal{MO}} \leq n-1$, which is a non-trivial result from [6]. In [100], a generalization to *weakly monotonic* automata is introduced and the upper bound $\mathrm{C}(A) \leq \left\lfloor \frac{n(n+1)}{6} \right\rfloor$ is proved for each n-state weakly monotonic automaton. For the class of *oriented* automata[3], Eppstein [33] gives a tight bound $(n-1)^2$, whose tightness is witnessed by the above-mentioned series of Černý. Oriented automata are defined by preserving a cyclic order of states. See also [101] for results about generalizations of oriented automata.

- Aperiodicity is a well-known and widely studied property of deterministic finite automata. Let us note two classical publications: First, in 1965 Schützenberger [85] proves an equivalence of two notions, both of them turns out to coincide with the notion of aperiodic DFA. Second, a book of McNaughton and Papert [63] reveals that the notion is equivalent to even more definitions coming from different branches of computer science and logic. Trakhtman [94] proved that $\mathrm{C}_n^{\mathcal{AP}} \leq \frac{n(n-1)}{2}$ for each $n \geq 1$. In [100] it is pointed out that each aperiodic DFA is weakly monotonic and thus $\mathrm{C}_n^{\mathcal{AP}} \leq \left\lfloor \frac{n(n+1)}{6} \right\rfloor$. It is not hard to show that each monotonic DFA is aperiodic. Lower bounds for aperiodic automata are still linear, the best one is $n + \left\lfloor \frac{n}{2} \right\rfloor - 1$, which follows from [5]

- Special attention is paid to the class of DFA that have a sink state, see Definition 1.5. In Section 2.2 we inspect related techniques and former results, and provide new isolated examples exceeding the best general lower bound concerning binary DFA.

Finally, the papers [8, 43] study classes of DFA based on the ranks of letters, while in [3, 4] and other articles, the authors study synchronization thresholds with respect to certain classes of DFA defined in terms of semigroup theory. All the results listed above may be seen as attempts to obtain general upper bounds of C_n or even to prove the Černý conjecture.

1.3.3 Lower Bounds for PFA and Subsets in DFA

First results about subset synchronization appeared in 1970's. The following is actually easy to prove:

Theorem 1.19 ([23])**.** *For each $n \geq 2$ and $k \leq n-2$ there is $\binom{n-2}{k-1}$-letter DFA with a k-state subset S such that* $\mathrm{sub}(A, S) \geq \binom{n-2}{k-1}$.

Corollary 1.20. $\mathrm{sub}(n) = \Omega\left(\frac{2^n}{\sqrt{n}}\right)$.

Proof. Set $k = \frac{n}{2}$ and use the Stirling's approximation to check that

$$\lim_{n \longrightarrow \infty} \binom{n}{\frac{n}{2}} = \sqrt{\frac{2}{\pi}} \cdot \frac{2^n}{\sqrt{n}}.$$

□

Subset synchronization was discussed also in [87], but an incorrect result is presented in this book. The threshold $\mathrm{car}(n)$ has been initially studied in 1982 by Goralčík et al., together with

[3] Also known as *monotonic* [33]

several related problems. The authors show that for infinitely many n there is a permutation of n states having order at least $(\sqrt[3]{n})!$ and they use it to prove the following:

Theorem 1.21 ([41])**.** $\mathrm{car}(n) \geq (\sqrt[3]{n})!$.

The construction can be easily (e.g. using our Lemma 2.1) modified to establish $\mathrm{sub}(n) \geq (\sqrt[3]{n})!$ as well, as it was later re-discovered in the paper [54]. Though exceeded by $\Omega\Big(\frac{2^n}{\sqrt{n}}\Big)$, the later lower bound of $\mathrm{sub}(n)$ remains interesting since the proof uses binary alphabets only.
In 2004, Ito and Shikishima-Tsuji revisited the topic and prove:

Theorem 1.22 ([48])**.** $\mathrm{car}(n) \geq 2^{\frac{n}{2}}$.

This was subsequently improved by Martyugin:

Theorem 1.23 ([59])**.** $\mathrm{car}(n) \geq 3^{\frac{n}{3}}$.

Again, the construction can be applied to subsets, so we get $\mathrm{sub}(n) \geq 3^{\frac{n}{3}}$.
For subset synchronization there is only an easy upper bound:

Lemma 1.24. $\mathrm{sub}(n) \leq \mathrm{csub}(n) \leq 2^n - n - 1$.

Proof. The first inequality is trivial. The second one follows from the fact that for a shortest careful reset word $x_1 \dots x_d \in \Sigma^\star$ of $A = (Q, \Sigma, \delta)$, the d sets

$$Q, \delta(Q, x_1), \delta(Q, x_1 x_2), \dots, \delta(Q, x_1 \dots x_{d-1})$$

are pairwise distinct non-empty and non-singleton subsets of Q. □

Known upper bounds of careful synchronization come from bounds of the function $\mathrm{d}_3(n)$, see Section 1.4.2. Namely, Theorem 1.31 implies that

$$\mathrm{car} = \mathcal{O}\big(n^2 \cdot 4^{\frac{n}{3}}\big).$$

Chapter 2.1 presents new results concerning lower bounds of subset synchronization and careful synchronization with respect to the alphabet size.

1.3.4 Road Coloring

Definitions related to road coloring were given in Section 1.2.3.

Lemma 1.25. *Let G be a directed multigraph. If G is road colorable, then it is admissible.*

Proof. The constant out-degree is clear. As for the cycles of G, let w be a reset word of a coloring $A = (Q, \Sigma, \delta)$ of G, such that $\delta(Q, w) = \{r\}$. Choose some $x \in \Sigma$ and consider the cycles $r \xrightarrow{v} r$ and $r \xrightarrow{x} \delta(r, x) \xrightarrow{v} r$. Their lengths are $|v|$ and $|v| + 1$, thus their greatest common divisor is 1. □

Theorem 1.26 (Road Coloring Theorem [91, 95])**.** *Let G be a strongly connected directed multigraph. Then G is road colorable if and only if it is admissible.*

What about directed multigraphs that are not strongly connected? We say that a strongly connected component of a directed multigraph is the *sink component* if it is reachable from any other component. Observe that:

1. If a sink component exists, then it is unique.
2. If G has constant out-degree k, the sink component also has constant out-degree k.

Corollary 1.27. *Let G be a directed multigraph. Then G is road colorable if and only if it has constant out-degree and its sink component is admissible.*

Proof. For the forward implication, the constant out-degree is trivial. Then, fix a synchronizing coloring of G and observe that the sink component corresponds to a subautomaton. Trivially, a subautomaton of a synchronizing DFA is synchronizing, so the sink component is road colorable and thus admissible due to Lemma 1.25. As for the backward implication - if the sink component is admissible, we use Theorem 1.26 to find a synchronizing coloring. Then we extend it arbitrarily to a coloring of G. Any such extension is synchronizing using a word that maps all the states into the subautomaton, followed by a reset word of the subautomaton. □

Once the Road coloring problem was solved, generalizations appeared, see [22].

1.4 Modifications of the Concepts

1.4.1 Reducing Range Size

Say that $k \in \mathbb{N}$ is the *rank* of a DFA $A = (Q, \Sigma, \delta)$ if k is the minimum of ranks of the letters $x \in \Sigma$ (see Def. 1.4). Pin [71] formulated a generalization of the Černý conjecture, saying that each DFA of rank at most k has a word of rank at most k and length at most $(n-k)^2$. This conjecture was disproved by Kari [50]. In [56] another formulation was introduced: each DFA of rank exactly k has a word of rank k and length at most $(n-k)^2$. This claim remains open since the counter-example from [50] has rank 1 (i.e. is synchronizing). In [7] the authors introduce the class of *generalized monotonic* DFA and prove that each generalized monotonic automaton of rank k has a word of rank k and length at most $n-k$. They also point out that each generalized monotonic automaton is aperiodic.

1.4.2 Synchronization of NFA

In 1999, Imreh and Steinby [46] introduced three different synchronization thresholds concerning general non-deterministic finite automata (NFA). The key definitions are the following

Definition 1.28. For an NFA $A = (Q, X, \delta)$, a word $w \in X^\star$ is:

- *D1-directing* if there is $r \in Q$ such that $\delta(s, w) = \{r\}$ for each $s \in Q$,
- *D2-directing* if $\delta(s_1, w) = \delta(s_2, w)$ for each $s_1, s_2 \in Q$,
- *D3-directing* if there is $r \in Q$ such that $r \in \delta(s, w)$ for each $s \in Q$.

Definition 1.29.

1. By $\mathrm{d}_1(A), \mathrm{d}_2(A), \mathrm{d}_3(A)$ we denote the length of shortest D1-, D2-, and D3-directing words for A, or 0 if there is no such word.
2. By $\mathrm{d}_1(n), \mathrm{d}_2(n), \mathrm{d}_3(n)$ we denote the maximum values of $\mathrm{d}_1(A), \mathrm{d}_2(A), \mathrm{d}_3(A)$ taken over all NFA A with at most n states.

Possible restrictions are marked by superscripts as usual.

It is clear that PFA are a special kind of NFA. Any careful reset word of a PFA A is D1-, D2-, and D3-directing. On the other hand, any D1- or D3-directing word of a PFA is a careful reset word. Thus, we get

$$\mathrm{d}_2^{\mathrm{PFA}}(n) \leq \mathrm{d}_1^{\mathrm{PFA}}(n) = \mathrm{d}_3^{\mathrm{PFA}}(n) = \mathrm{car}(n)$$

and

$$\mathrm{d}_1(n) \geq \mathrm{car}(n), \quad \mathrm{d}_3(n) \geq \mathrm{car}(n). \tag{1.1}$$

Note that each D2-directing word w of a PFA A is either a careful reset word of A or satisfies that $\delta(\{s\}, w) = \emptyset$ for each $s \in Q$. PFA are of a special importance for the threshold $\mathrm{d}_3(n)$ due to the following key lemma:

Lemma 1.30 ([48]). *For any n-state NFA A, there is a n-state PFA B such that $\mathrm{d}_3(B) \geq \mathrm{d}_3(A)$. Thus*

$$\mathrm{d}_3(n) = \mathrm{d}_3^{\mathrm{PFA}}(n) = \mathrm{car}(n)$$

for each n.

It is known that $\mathrm{d}_1(n) = \Omega(2^n)$ [48] and $\mathrm{d}_3(n) = \Omega\big(3^{\frac{n}{3}}\big)$ [59] (for upper bounds and further details see [38]).

Theorem 1.31 ([38]). *It holds that:*

1. $\mathrm{d}_1(n) = \Theta(2^n)$,
2. $\mathrm{d}_2(n) = \Theta(2^n)$,
3. $\mathrm{d}_3(n) = \mathcal{O}\big(n^2 \cdot 4^{\frac{n}{3}}\big)$.

See [11] and its references for results concerning D2-directing words of *unambiguous* and *local* automata. Certain variant of synchronization of PFA different from the careful synchronization has been studied in [15, 96].

1.4.3 Composing Functions

It has been pointed out by Arto Salomaa [81] in 2001 that very little was known about the minimum length of a composition needed to generate a function by a given set of generators. To be more precise, let us adopt and slightly extend the notation. We denote by $\mathcal{T}_n$ the semigroup of all functions from $\{1, \ldots, n\}$ to itself. Given $\mathbf{G} \subseteq \mathcal{T}_n$, we denote by $\langle \mathbf{G} \rangle$ the subsemigroup generated by $\mathbf{G}$. Given $\mathbf{F} \subseteq \mathcal{T}_n$, we denote by $\mathrm{D}(\mathbf{G}, \mathbf{F})$ the length k of a shortest sequence $g_1, \ldots, g_k$ of functions from $\mathbf{G}$ such that $g_1 \circ \cdots \circ g_k \in \mathbf{F}$. Finally, denote

$$\mathrm{D}_n = \max_{\overline{n} \leq n} \max_{\substack{\mathbf{F}, \mathbf{G} \subseteq \mathcal{T}_{\overline{n}} \\ \mathbf{F} \cap \langle \mathbf{G} \rangle \neq \emptyset}} \mathrm{D}(\mathbf{G}, \mathbf{F}). \tag{1.2}$$

From basic connections between automata and transformation semigroups it follows that various synchronization thresholds can be defined alternatively by putting additional restrictions to the space of considered sets $\mathbf{G}$ and $\mathbf{F}$ in the definition (1.2) of the threshold D_n:

1. For the basic synchronization threshold of DFA (may be denoted by $\mathrm{car}^{\mathrm{DFA}}(n)$), we restrict $\mathbf{F}$ to be exactly the set of $\overline{n}$-ary constant functions. Recall that a set $\mathbf{G} \subseteq \mathcal{T}_{\overline{n}}$ corresponds to a DFA $A = (\{1, \ldots \overline{n}\}, \Sigma, \delta)$: Each $g \in \mathbf{G}$ just encodes the action of certain $x \in \Sigma$. Finding a reset word of A then equals composing transitions from $\mathbf{G}$ in order to get a constant.

2. For the threshold $\mathrm{sub}(n)$, we restrict $\mathbf{F}$ to be some of the sets

   $$\mathbf{F}_S = \{f \in \mathcal{T}_{\overline{n}} \mid (\forall r, s \in S)\, f(r) = f(s)\}$$

   for $S \subseteq \{1, \ldots, \overline{n}\}$. Therefore it holds that $\mathrm{D}_n \geq \mathrm{sub}(n)$.

3. For $\mathrm{car}(n)$, we should consider an alternative formalism for PFA, where the „undefined" transitions lead to a special error sink state. Let the largest number stand for the error

state. A careful reset word should map all the states except for the error state to one particular non-error state. So, here we restrict

$$\begin{aligned}\mathbf{F} &= \{f \in \mathcal{T}_{\overline{n}} \mid (\forall r, s \in \{1, \ldots, \overline{n}-1\})\, f(r) = f(s) \neq \overline{n}\}\,,\\ \mathbf{G} &\subseteq \{g \in \mathcal{T}_n \mid g(n) = n\}\,.\end{aligned}$$

However, in the canonical formalism such $\mathbf{G} \subseteq \mathcal{T}_n$ corresponds to a $(n-1)$-state PFA, so we get $\mathrm{D}_n \geq \mathrm{car}(n-1)$. Allowing suitable sets $\mathbf{F}_S$ for $S \subseteq \{1, \ldots, \overline{n}-1\}$, we get $\mathrm{D}_n \geq \mathrm{csub}(n-1)$ as well.

Arto Salomaa refers to a single nontrivial bound of D_n, namely $\mathrm{D}_n \geq (\sqrt[3]{n})!$. In fact, he omits a construction of Kozen [53, Theorem 3.2.7] from 1977, which deals with lengths of *proofs* rather than compositions but witnesses easily that $\mathrm{D}_n = 2^{\Omega\left(\frac{n}{\log n}\right)}$. However, the lower bound of $\mathrm{car}(n)$ from [49] revealed soon that $\mathrm{D}_n = 2^{\Omega(n)}$. Finally, in [70] the following tight bound was derived from properties of finite groups (see also [45]):

$$\mathrm{D}_n = 2^n \cdot e^{(1+o(1))\sqrt{\frac{n}{2}\ln n}}.$$

In the above-mentioned article [41] the authors study an even more general concept: composing binary relations. Such questions can be equivalently formulated as questions about NFA.

1.5 Computational Problems

1.5.1 Synchronization of DFA

Various basic computational problems arise from the study of DFA synchronization:

- *Given an automaton, decide if it is synchronizing.* A relatively simple algorithm, which could be traced back to [26], works in polynomial time.
- *Given a synchronizing automaton and a number d, decide if d is the length of shortest reset words.* This has been shown to be both NP-hard [33] and coNP-hard. More precisely, it is DP-complete [69].
- *Given a synchronizing automaton and a number d, decide if there exists a reset word of length d.* This is proven to be NP-complete [33]. It is also NP-complete to decide, whether d is within a constant [19] or even logarithmic [15] factor from the length of shortest reset words. Following the notation of [60], we call it Syn.

Assuming that $\mathcal{M}$ is a class of automata and membership in $\mathcal{M}$ is polynomially decidable, we define the restricted problem:

Syn($\mathcal{M}$)	
Input:	Synchronizing automaton $A = (Q, \Sigma, \delta) \in \mathcal{M}$, $d \in \mathbb{N}$
Output:	Does A have a reset word of length d?

Among the above-mentioned facts, the following are the most important for the present thesis. Corollary 1.33 follows from Theorem 1.32 and Theorem 1.15.

Theorem 1.32 ([26])**.** *There is a polynomial-time algorithm that decides whether a given automaton is synchronizing.*

Corollary 1.33. Syn*, if restricted to the instances with $d \geq \frac{|Q|^3-|Q|}{6}$, is solvable in polynomial time.*

Theorem 1.34 ([33]). SYN *is NP-complete, even if restricted to automata with two-letter alphabets.*

In Section 3.2 we study the complexity of SYN with restrictions to certain special classes of automata.

1.5.2 Synchronization of PFA and Subset Synchronization

The first natural problems in these directions are:

SUBSET SYNCHRONIZABILITY	
Input:	n-state DFA $A = (Q, \Sigma, \delta)$, $S \subseteq Q$
Output:	Is there some $w \in \Sigma^\star$ such that $\lvert\delta(S, w)\rvert = 1$?

CAREFUL SYNCHRONIZABILITY	
Input:	n-state PFA $A = (Q, \Sigma, \delta)$,
Output:	Is there some $w \in \Sigma^\star$ such that $(\exists r \in Q)\,(\forall s \in Q)\,\delta(s, w) = r$?

Both these problems, in contrast to the synchronizability of DFA, are known to be PSPACE-complete, which is further discussed in Section :

Theorem 1.35 ([68, 82]). SUBSET SYNCHRONIZABILITY *is a PSPACE-complete problem.*

Theorem 1.36 ([61]). CAREFUL SYNCHRONIZABILITY *is a PSPACE-complete problem.*

There are also interesting results about an alternative (*non-careful*) variant of PFA synchronization. In this case, it has been shown that the basic synchronizability problem is solvable in polynomial time for strongly connected automata [96] but becomes PSPACE-complete if we only require all the states to be reachable from one particular state [15].

1.5.3 Road Coloring

As the aperiodicity of a given directed multigraph can be tested in polynomial time, Theorem 1.26 and Corollary 1.27 imply that in polynomial time we can decide whether a graph is road colorable. For finding an exact synchronizing coloring of a given graph, a cubic time algorithm was developed by Trahtman [92] and improved to quadratic time by Béal and Perrin [12].
In the present thesis (Chapter 4) we study the following two problems, which deal with more subtle properties of synchronizing colorings:

SRCP	
Input:	Alphabet Σ, admissible graph $G = (Q, E)$ with out-degrees $\lvert\Sigma\rvert$, $d \in \mathbb{N}$
Output:	Is there a coloring δ such that $\lvert\delta(Q, w)\rvert = 1$ for some $w \in \Sigma^\star$ of length at most d?

SRCW	
Input:	Alphabet Σ, graph $G = (Q, E)$ with out-degrees $\lvert\Sigma\rvert$, $W \subseteq \Sigma^\star$
Output:	Is there a coloring δ such that $\lvert\delta(Q, w)\rvert = 1$ for some $w \in W$?

Restrictions of these two problems are denoted by subscripts and superscripts. A term of the form $\mathrm{SRCP}^{\mathcal{M}}_{|\Sigma|,d}$ denotes the restriction of SRCP to instances with out-degrees $|\Sigma|$, number d, and graphs from a class $\mathcal{M}$. Similarly, a term of the form $\mathrm{SRCW}^{\mathcal{M}}_{|\Sigma|,W}$ denotes the restriction of SRCW to instances with out-degrees $|\Sigma|$, set W of prescribed words, and graphs from a class $\mathcal{M}$. If a subscript or superscript is omitted, the corresponding parameter is not restricted.

1.5.4 Key Notions of Parameterized Complexity

In most of the paper, we do not need to work with any formal definition of a *parameterized problem*. We see it as a classical decision problem where we consider some special numerical property (*parameter*) of each input. Parameterized complexity is the study of the way in which the hardness of an NP-complete problem relies on the parameter. A problem may remain NP-hard even if restricted to instances with a particular value of the parameter or there may be a polynomial-time algorithm for each such value. In the second case, if the algorithm is the same for all the values (a *uniform* algorithm), the problem is said to lie in the class *XP*. Moreover, if the time-bounding polynomials for different values are all of the same degree, we get into the class *FPT*:
A parameterized problem *is fixed-parameter tractable* (*FPT*) if there is an algorithm that decides it in time

$$f(P) \cdot r(|x|)$$

where x is the input string, $P \in \mathbb{N}$ its parameter, r is an appropriate polynomial, and f is any computable function. If there is more than one possible parameter for a problem, one may consider *combinations* of the parameters. A problem is FPT with respect to parameters P, Q if it is decidable in time

$$f(P,Q) \cdot r(|x|)\,.$$

This is typically much less restrictive condition than the previous one, where f depends on P only.
There is a hierarchy of problems (the *W-hierarchy*) lying in XP but possibly outside FPT. It consists of the classes $\mathrm{W}[1]\,, \mathrm{W}[2]\,, \ldots$:

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subset \mathrm{XP}. \tag{1.3}$$

Since it has been conjectured that all the inclusions are proper, it is common to use $\mathrm{W}[k]$-hardness (with respect to an appropriate type of reduction) as an evidence of lying outside FPT. However, we do not need to define the W-hierarchy here since it is used only to formulate the preceding results (see Table 3.1), not for the new ones. See the textbook [30] for the definitions and many other great ideas of parameterized complexity.
A *kernel* of a parameterized problem is a polynomial-time procedure that transforms any input x of the problem to another input y such that the length and the parameter of y are bounded by some function f of the parameter associated with x. Having a kernel is equivalent to lying in FPT. If the function f is a polynomial, we get a *polynomial kernel*.
For functions $f, g : \mathbb{N} \to \mathbb{N}$, the term $g = \mathcal{O}^\star(f)$ means that for a suitable polynomial p and each $x \in \mathbb{N}$ we have $g(x) \leq f(x) \cdot p(x)$.

Chapter 2

Lower Bounds of Synchronization Thresholds

In this chapter we deal mostly with a new lower bound for subset synchronization in binary DFA and careful synchronization in binary PFA. Specifically, we show that both the thresholds are of the form $2^{\Omega(n)}$, even if only strongly connected automata are considered. The results were presented at the conference *AFL 2014* [103] and submitted in an extended paper to a journal.
Besides of that, we include Section 2.2, which deals with the classical synchronization threshold of DFA, restricted to automata having a *sink state* - a state with no outgoing transitions except for loops (also known as a *zero state*). We inspect former lower bounds and propose certain generalizations. The proposed methods turn out to be useful - they lead to slightly improved lower bounds for several small values of the number of states, using computationally verified examples. Moreover, the examples belong to a simple infinite series of automata, so it seems very likely that the entire series has the desired properties, which is a topic for further research.

2.1 Careful Synchronization and Subset Synchronization

In Section 1.2.2 we have defined the thresholds car_n and sub_n, while in Section 1.3.3 we have pointed out that $\mathrm{car}(n) \geq 3^{\frac{n}{3}}$ (Theorem 1.23). In fact, $\mathrm{sub}(n) \geq 3^{\frac{n}{3}}$ holds as well, since the proof of Theorem 1.23 can be easily modified in order to operate with subsets of states in DFA (see Lemma 2.1). However, the proof uses very artificial series of automata to witness this lower bound:

- In the series, the alphabet size grows linearly with the growing number of states - the result relies on the convention of measuring the *size* of an automaton only by the number of states, ignoring the alphabet size. The result says nothing about the thresholds $\mathrm{sub}^{\mathcal{AL}_k}(n)$ or $\mathrm{car}^{\mathcal{AL}_k}(n)$ for any fixed $k \geq 2$. In 2013, Martyugin [62] proves that

$$\begin{aligned} \mathrm{car}^{\mathcal{AL}_2}(n) &> 3^{\frac{n}{6 \cdot \log_2 n}}, \\ \mathrm{car}^{\mathcal{AL}_k}(n) &> 3^{\frac{n}{3 \cdot \log_{m-1} n}} \end{aligned}$$

  for each $k \geq 3$, which applies in a similar form also to subset synchronization. However, it remained unclear whether $\mathrm{car}^{\mathcal{AL}_k}(n) = 2^{\Omega(n)}$ or $\mathrm{sub}^{\mathcal{AL}_k}(n) = 2^{\Omega(n)}$ for some $k \geq 2$. Here we confirm this for $k = 2$, so for any greater k the claim follows easily.

- In the subset-synchronization form, the series consists of DFA with sink states, two of them in each automaton. Use of sink states is a very strong tool for designing automata having given properties, but in practice such automata seem very special. They represent

unstable systems balancing between different deadlocks. The very opposite are strongly connected automata. Does the threshold remain so high if we consider only strongly connected DFA? Unfortunately, we show below that it does, even if we restrict the alphabet size to a constant. We introduce *swap congruences* as an alternative to sink states.
Note that in the case of careful synchronization, any lower bound of $\mathrm{car}(n)$ applies easily to $\mathrm{car}^{\mathcal{SC}}(n)$ using a simple trick from Lemma 2.2. Moreover, for suitable series the alphabet size is increased only by a constant.

In this chapter we prove that

$$\begin{aligned} \mathrm{sub}^{\mathcal{AL}_2\cap\mathcal{SC}}(n) &= 2^{\Omega(n)}, \\ \mathrm{car}^{\mathcal{AL}_2\cap\mathcal{SC}}(n) &= 2^{\Omega(n)}, \end{aligned}$$

which shows that the bounds remain high even if restricted to binary, strongly connected automata. The new bounds are tight in the sense of $\mathrm{car}(n) = 2^{\Theta(n)}$ and $\mathrm{sub}(n) = 2^{\Theta(n)}$. This result has the following consequences:

- The naturally related computational problems are Subset synchronizability and Careful Synchronizability (see Section 1.5.2). Both these problems, in contrast to the synchronizability of DFA, are known to be PSPACE-complete. Note that such hardness is not a consequence of any lower bound of synchronization thresholds, because an algorithm does not need to produce an explicit reset word. The proofs of both the theorems above make use of a result of Kozen [53], which establishes that it is PSPACE-complete to decide if given finite acceptors with a common alphabet accept a common word. This problem is polynomially reduced to our problems using the idea of two sink states. Is it possible to avoid the non-connectivity here? In Theorem 2.10 we give a positive answer: for Careful synchronizability it is in fact a trivial task, and in the nontrivial case of Subset synchronizability the method of swap congruences is general enough to perform a suitable reduction.

- It is known that $\mathrm{d}_1(n) = \Omega(2^n)$ [49] and $\mathrm{d}_3(n) = \Omega\big(3^{\frac{n}{3}}\big)$ [59] (for upper bounds and further details see [38]). Due to the easy relationship similar to (1.1) in Section 1.4.2, our strongly exponential lower bounds apply directly to the thresholds $\mathrm{d}_1(n)$ and $\mathrm{d}_3(n)$ with the restriction to binary strongly connected NFA:

$$\mathrm{d}_1^{\mathcal{AL}_2\cap\mathcal{SC}}(n) = 2^{\Omega(n)}, \quad \mathrm{d}_3^{\mathcal{AL}_2\cap\mathcal{SC}}(n) = 2^{\Omega(n)}.$$

- Like in the case of $\mathrm{car}(n)$ and $\mathrm{sub}(n)$, the notion of D_n does not concern the size of $\mathbf{G}$, thus providing a ground for artificial series of bad cases based on growing alphabets. Our results show that actually the growing size of $\mathbf{G}$ is not necessary: a strongly exponential lower bound of D_n holds even if we restrict $\mathbf{G}$ to any nontrivial fixed size.

First, in Sections 2.1.1 to 2.1.4 we prepare the ground for the proof of the results formulated above by introducing basic principles and relationships concerning the studied thresholds. These principles are not innovative, except for the method using *swap congruences* described in Section 2.1.2, dealing with strong connectivity in subset synchronization. The main proof is presented in Sections 2.1.5 and 2.1.6.
As noted before, many of the lower bounds of $\mathrm{car}(n)$ and $\mathrm{sub}(n)$ found in the literature were formulated for only one of the notions but used ideas applicable to the other as well. The key method used in the present paper for the binary case is of this kind again. However, we are not able to calculate any of the thresholds from the other exactly.

2.1.1 Determinization by adding sink states

The following inequality is not a key tool of the present paper, we prove it in order to illustrate that even careful subset synchronization is not much harder than subset synchronization itself. Recall that trivially $\mathrm{car}(n) \leq \mathrm{csub}(n)$ and $\mathrm{sub}(n) \leq \mathrm{csub}(n)$ for each n.

Lemma 2.1. *For each $n \geq 1$ it holds that*

$$\mathrm{csub}(n) \leq \mathrm{sub}(n+2) - 1.$$

Proof. Take any PFA $A = (Q_A, \Sigma_A, \delta_A)$ with a carefully synchronizable subset $S_A \subseteq Q_A$ and choose a shortest careful reset word $w \in \Sigma^\star$ of S_A with $\delta_A(s, w) = r_0$ for each $s \in S_A$. We construct a DFA $B = (Q_B, \Sigma_B, \delta_B)$ and a synchronizable subset $S_B \subseteq Q_B$ such that $\mathrm{sub}(B, S_B) \geq |w| + 1$. Let us set

$$\begin{aligned} Q_B &= Q_A \cup \left\{\mathrm{D}, \overline{\mathrm{D}}\right\}, \quad & \delta_B(\mathrm{D}, x) &= \mathrm{D}, \\ X_B &= X_A \cup \{\omega\}, & \delta_B\left(\overline{\mathrm{D}}, x\right) &= \overline{\mathrm{D}} \end{aligned}$$

for each $x \in \Sigma_B$, and

$$\delta_B(s,x) = \begin{cases} \delta_A(s,x) & \text{if defined} \\ \overline{\mathrm{D}} & \text{otherwise} \end{cases} \qquad \delta_B(s,\omega) = \begin{cases} \mathrm{D} & \text{if } s = r_0 \\ \overline{\mathrm{D}} & \text{otherwise} \end{cases}$$

for each $s \in Q_A, x \in \Sigma_A$. Denote $S_B = S_A \cup \{\mathrm{D}\}$. The word $w\omega$ witnesses that the subset S_B is synchronizable. On the other hand, let v be any reset word of S_B. Since D is a sink state and $\mathrm{D} \in S_B$, we have $\delta_B(v, s) = \mathrm{D}$ for each $s \in S_B$. Thus:

- The state $\overline{\mathrm{D}}$ is not active during the application of v.
- There need to be an occurrence of ω in v.

Denote $v = v_0 \omega v_1$, where $v_0 \in \Sigma_A^\star$ and $v_1 \in \Sigma_B^\star$. If $|\delta_B(S_B, v_0) \cap Q_A| = 1$, we are done since v_0 maps all the states of S_A to a unique state using only the transitions defined in A, so $|v| \geq |w| + 1$. Otherwise, there is some $s \in \delta_B(Q_B, v_0) \cap Q_A$ such that $s \neq r_0$, but then $\delta_B(\omega, s) = \overline{\mathrm{D}}$, which is a contradiction.

□

2.1.2 Strong connectivity

First we show an easy reduction concerning careful synchronization of strongly connected PFA. We use a simple trick: A letter that is defined only on a single state cannot appear in a shortest careful reset word, so one can make a PFA strongly connected by adding such letters. The number of new letters needed may be reduced by adding special states, but the simple variant described by Lemma 2.2 is illustrative and strong enough for our purpose.
For each $j \geq 0$ we define the class $\mathcal{C}_j$ of PFA as follows. A PFA $A = (Q, \Sigma, \delta)$ belongs to $\mathcal{C}_j$ if there are j pairs $(r_1, q_1), \ldots, (r_j, q_j) \in Q \times Q$ such that adding transitions of the form $r_i \longrightarrow q_i$ for each $i = 1, \ldots, j$ makes the automaton strongly connected. Note that $\mathcal{C}_0 = \mathcal{SC}$.

Lemma 2.2. *For each $n, k, j \geq 1$ it holds that*

$$\mathrm{car}^{\mathcal{AL}_k \cap \mathcal{C}_j}(n) \leq \mathrm{car}^{\mathcal{AL}_{k+j} \cap \mathcal{SC}}(n).$$

Proof. Take any PFA $A = (Q, \Sigma_A, \delta_A) \in \mathcal{AL}_k \cap \mathcal{C}_j$ together with the pairs $(r_1, q_1), \ldots, (r_j, q_j) \in Q \times Q$ from the definition of $\mathcal{C}_j$. We construct a PFA $B = (Q, \Sigma_B, \delta_B)$ where $\Sigma_B = \Sigma_A \cup$

$\{\psi_1, \ldots, \psi_j\}$, $\delta_B(s, x) = \delta_A(s, x)$ for $x \in \Sigma_A$ and $s \in Q$, and

$$\delta_B(s, \psi_i) = \begin{cases} q_i & \text{if } s = r_i \\ \text{undefined} & \text{otherwise} \end{cases}$$

for $i = 1 \ldots, , j'$ and $s \in Q$. Now it is easy to check that B is strongly connected and that $\mathrm{car}(B) = \mathrm{car}(A)$. □

Second, we present an original method concerning subset synchronization of strongly connected DFA. All the lower bounds applicable to $\mathrm{sub}(n)$ that we have found in the literature used two sink states (deadlocks) to force application of particular letters during a synchronization process. A common step in such proof looks like *„The letter x cannot be applied since that would make the sink state $\overline{\mathrm{D}}$ active, while another sink state* D *is active all the time"*. In order to prove a lower bound of $\mathrm{sub}^{\mathcal{SC}}(n)$, we have to develop an alternative mechanism. Our mechanism relies on *swap congruences*:
Recall that, given a DFA $A = (Q, \Sigma, \delta)$, an equivalence relation $\rho \subseteq Q \times Q$ is a *congruence* if

$$r \rho s \Rightarrow \delta(r, x) \, \rho \, \delta(s, x)$$

for each $x \in \Sigma$. We say that a congruence ρ is a *swap congruence* of a DFA if, for each equivalence class C of ρ and each letter $x \in \Sigma$, the restricted function $\delta : C \times \{x\} \to Q$ is injective. The key property of swap congruences is the following:

Lemma 2.3. *Let $A = (Q, \Sigma, \delta)$ be a DFA, let $\rho \subseteq Q^2$ be a swap congruence and take any $S \subseteq Q$. If there are any $r, s \in S$ with $r \neq s$ and $r \rho s$, the set S is blind.*

Proof. Because r and s lie in a common equivalence class of ρ, by the definition of a swap congruence we have $\delta(r, w) \neq \delta(s, w)$ for any $w \in \Sigma^\star$. □

Thus, the alternative mechanism relies on arguments of the form *„The letter x cannot be applied since that would make both the states r, s active, while it holds that $r \rho s$"*. It turns out that our results based on the method can be derived from more transparent but not strongly connected constructions by the following reduction principle:

Lemma 2.4. *For each $n \geq 1$ it holds that*

$$\mathrm{sub}(n) \leq \mathrm{sub}^{\mathcal{SC}}(2n + 2) - 1.$$

Moreover, for each $n, k \geq 1$ and $j \geq 2$ it holds that

$$\mathrm{sub}^{\mathcal{AL}_k \cap \mathcal{C}_j}(n) \leq \mathrm{sub}^{\mathcal{AL}_{k+j} \cap \mathcal{SC}}(2n + 2) - 1.$$

Proof. The first claim follows easily from the second one. So, take any DFA $A = (Q_A, \Sigma_A, \delta_A) \in \mathcal{AL}_k \cap \mathcal{C}_j$ together with the pairs $(r_1, q_1), \ldots, (r_j, q_j) \in Q_A \times Q_A$ from the definition of $\mathcal{C}_j$ and let $S \subseteq Q_A$ be synchronizable. We construct a strongly connected DFA $B = (Q_B, \Sigma_B, \delta_B)$ and a subset $S_B \subseteq Q_B$ such that $\mathrm{sub}\,(B, S_B) \geq \mathrm{sub}\,(A, S_A) + 1$. Let us set

$$\begin{aligned} Q_B &= \{s, \overline{s} \mid s \in Q_A\} \cup \left\{\mathrm{E}, \overline{\mathrm{E}}\right\}, \\ X_B &= X_A \cup \{\psi_1, \ldots, \psi_j\}. \end{aligned}$$

We want the relation

$$\rho = \langle (s, \overline{s}) \mid s \in Q_A \cup \{\mathrm{E}\} \rangle,$$

where $\langle \ldots \rangle$ denotes an equivalence closure, to be a swap congruence. Regarding this requirement, it is enough to define δ_B on $Q_A \cup \{\mathrm{E}\}$. The remaining transitions are forced by the

injectivity on the equivalence classes. We set

$$\delta_B(s,x) = \delta_A(s,x)\,, \quad \delta_B(\mathrm{E},x) = \mathrm{E}$$

for any $s \in Q_A, x \in \Sigma_A$, while the letters $\psi_1,\ldots,\psi_j$ act as follows:

$$\delta_B(s,\psi_I) = \begin{cases} q_I & \text{if } s = r_i \\ \overline{q_I} & \text{otherwise} \end{cases}, \quad \delta_B(\mathrm{E},\psi_I) = q_I,$$
$$\delta_B(s,\psi_i) = \begin{cases} q_i & \text{if } s = r_i \\ \overline{\mathrm{E}} & \text{otherwise} \end{cases}, \quad \delta_B(\mathrm{E},\psi_i) = \mathrm{E}$$

for $s \in Q_A$ and $i \neq I$, where I is chosen such that for a reset word w of S_A in A with $\delta_A(s,w) = r_0$, the state r_I is reachable from r_0. It is easy to see that such I exists for any $r_0 \in S_A$. We set $S_B = S_A \cup \{\mathrm{E}\}$.

- First, note that the set S_B is synchronizable in B by the word $wu\psi_I$ where $u \in \Sigma_A^\star$ such that $\delta_A(r_0,u) = r_I$.
- On the other hand, let v be a reset word of S_B in B. The word v necessarily contains some ψ_i for $i \in \{1,\ldots,j\}$, so we can write $v = v_0\psi_i v_1$, where $v_0 \in \Sigma_A^\star, v_1 \in \Sigma_B^\star$. If v_0 is a reset word of S_A in A, $|v| \geq \mathrm{sub}(A,S_A)+1$ and we are done. Otherwise there is a state $s \neq r_i$ in $\delta_B(S,v_0)$ and we see that both q_i and $\overline{q_i}$ (if $i = I$) or both E and $\overline{\mathrm{E}}$ (if $i \neq I$) lie in $\delta_B(S,v_0\psi_1)$, which is a contradiction with properties of the swap congruence ρ.

The automaton B is strongly connected since the transitions $r_i \xrightarrow{\psi_i} q_i$ and $\overline{r_i} \xrightarrow{\psi_i} \overline{q_i}$ for each $i = 1,\ldots,j$ make both the copies of A strongly connected and there are transitions $\mathrm{E} \xrightarrow{\psi_I} q_I$, $s \xrightarrow{\psi_i} \overline{\mathrm{E}}$, $\overline{\mathrm{E}} \xrightarrow{\psi_I} \overline{q_2}$, and $\overline{s} \xrightarrow{\psi_i} \mathrm{E}$ for some $i \neq I$ and $s \neq r_i$.

□

2.1.3 A special case of subset synchronization

We are not aware of any general bad-case reduction from subset synchronization to careful synchronization. Here we suggest a special class (denoted by $\mathcal{M}_\mathrm{P}$) of pairs automaton-subset such that the instances from the class are in certain sense reducible to careful synchronization. The main construction of the present paper (i.e. the proof of Lemma 2.7) yields instances of subset synchronization that fit to this class. We use the following definitions:

- Given a PFA $A = (Q,\Sigma,\delta)$ and a carefully synchronizable subset $S \subseteq Q$, the *S-relevant part* of A is

  $$Q_{A,S} \;=\; \bigcup_{w \in W_S} \delta(S,w)\,,$$

  where W_S is the set of prefixes of careful reset words of S in A. The *S-relevant automaton* of A is $R_{A,S} = (Q_{A,S},\Sigma,\delta_{A,S})$, where

  $$\delta_{A,S}(s,x) = \begin{cases} \delta(s,x) & \text{if } \delta(s,x) \in Q_{A,S} \\ \text{undefined} & \text{otherwise} \end{cases}$$

  for each $s \in Q_{A,S}$ and $x \in \Sigma$.

- The class $\mathcal{M}_\mathrm{P}$ is defined as follows. For any PFA $A = (Q,\Sigma,\delta)$ and any carefully synchronizable $S \subseteq Q$, the pair $\langle A,S\rangle$ lies in $\mathcal{M}_\mathrm{P}$ if there are subsets $P_1,\ldots,P_{|S|} \subseteq Q$ such that:

– The sets $P_1, \ldots, P_{|S|}$ are disjoint and $\bigcup_{i=1}^{|S|} P_i = Q_{A,S}$.

– For each $v \in \Sigma^\star$ such that $\delta(s,u) \in Q_{A,S}$ for any prefix u of v and any $s \in S$, it holds that v is a careful reset word of S, or

$$|\delta(S,v) \cap P_i| = 1$$

for each $i = 1, \ldots, |S|$. In particular, the choice of empty v implies that

$$|S \cap P_i| = 1$$

must hold for each $i = 1, \ldots, |S|$.

- The class $\mathcal{C}_j^{\mathrm{R}}$ for $j \geq 0$ is defined as follows. For any PFA $A = (Q, \Sigma, \delta)$ and any carefully synchronizable $S \subseteq Q$, the pair $\langle A, S\rangle$ lies in $\mathcal{C}_j^{\mathrm{R}}$ if $R_{A,S} \in \mathcal{C}_j$.

Lemma 2.5. *For each $n \geq 1$ it holds that*

$$\mathrm{csub}^{\mathcal{M}_{\mathrm{P}}}(n) \leq \mathrm{car}(n)\,.$$

Moreover, for each $n, k, j \geq 1$ it holds that

$$\mathrm{csub}^{\mathcal{AL}_k \cap \mathcal{C}_j^{\mathrm{R}} \cap \mathcal{M}_{\mathrm{P}}}(n) \leq \mathrm{car}^{\mathcal{AL}_{k+1} \cap \mathcal{C}_j}(n)\,.$$

Proof. The first claim follows easily from the second. So, take any $\langle A, S\rangle \in \mathcal{M}_{\mathrm{P}}$ with $A = (Q, \Sigma_A, \delta_A)$ and $S \subseteq Q$, together with the sets $P_1, \ldots, P_{|S|}$ from the definition of $\mathcal{M}_{\mathrm{P}}$. By adding a letter α to the automaton $R_{A,S}$, we construct a carefully synchronizing PFA $B = (Q_{A,S}, \Sigma_B, \delta_B)$ with $\mathrm{car}(B) \geq \mathrm{csub}(A,S)$. Let $\Sigma_B = \Sigma_A \cup \{\alpha\}$. For each $s \in Q_{A,S}$ we find the i such that $s \in P_i$ and define

$$\delta_B(s, \alpha) = q_i,$$

where q_i is the only state lying in $S \cap P_i$, as guaranteed by the membership in $\mathcal{M}_{\mathrm{P}}$. The letters of Σ_A act in B as they do in $R_{A,S}$.

- It is easy to check that the automaton B is carefully synchronizing by αw for any $w \in \Sigma_A^\star$ that is a careful reset word of S in A.

- On the other hand, take a shortest careful reset word v of B. If α does not occur in v, then v is a careful reset word of S in A, so $|v| \geq \mathrm{csub}\,(A,S)$. Otherwise, denote $v = v_0 \alpha v_1$ where $v_0 \in \Sigma_B^\star$ and $v_1 \in \Sigma_A^\star$. By the membership in $\mathcal{M}_{\mathrm{P}}$ we have $|\delta(S, v_0) \cap P_i| = 1$ for each $i = 1, \ldots, |S|$ and thus $\delta_B(S, v_0\alpha) = S$. It follows that v_1 is a careful reset word of S in A, so $|v| \geq \mathrm{csub}\,(A,S)$.

□

2.1.4 Decreasing the alphabet size

The following method is quite simple and has been already used in the literature [19]. It modifies an automaton in order to decrease the alphabet size while preserving high synchronization thresholds.

Lemma 2.6. *For each $n, k \geq 1$ it holds that*

1. $\mathrm{sub}^{\mathcal{AL}_k}(n) \leq \mathrm{sub}^{\mathcal{AL}_2}(k \cdot n)$ *and* $\mathrm{sub}^{\mathcal{AL}_k \cap \mathcal{SC}}(n) \leq \mathrm{sub}^{\mathcal{AL}_2 \cap \mathcal{SC}}(k \cdot n)\,,$
2. $\mathrm{car}^{\mathcal{AL}_k}(n) \leq \mathrm{car}^{\mathcal{AL}_2}(k \cdot n)$ *and* $\mathrm{car}^{\mathcal{AL}_k \cap \mathcal{SC}}(n) \leq \mathrm{car}^{\mathcal{AL}_2 \cap \mathcal{SC}}(k \cdot n)\,.$

Proof. Take a PFA $A = (Q_A, \Sigma_A, \delta_A)$ with $\Sigma_A = \{a_0, \dots, a_m\}$. We define a PFA $B = (Q_B, \Sigma_B, \delta_B)$ as follows: $Q_B = Q_A \times \Sigma_A$, $\Sigma_B = \{\alpha, \beta\}$, and

$$\begin{aligned}
\delta_B((s, a_i), \alpha) &= \begin{cases} (\delta_A(s, a_i), a_0) & \text{if } \delta_A(s, a_i) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases} \\
\delta_B((s, a_i), \beta) &= \begin{cases} (s, a_{i+1}) & \text{if } i < m \\ (s, a_m) & \text{if } i = m \end{cases}
\end{aligned}$$

for each $i = 0, \dots, m$. The construction of B applies to both the claims:

1. Let A be a DFA. We choose a synchronizable $S_A \subseteq Q_A$ and denote $S_B = S_A \times \{a_0\}$. It is not hard to see that reset words of S_B in B are in a one-to-one correspondence with reset words of S_A in A. A word $a_{i_1} \dots a_{i_d} \in \Sigma_A^\star$ corresponds to $\left(\beta^{i_1}\alpha\right) \dots \left(\beta^{i_d}\alpha\right) \in \Sigma_B^\star$.

2. Let A be carefully synchronizing. We can suppose that $\delta_A(s, a_m)$ is defined on each $s \in Q_A$ since for a carefully synchronizing PFA there always exists such letter. For any careful reset word $a_{i_1} \dots a_{i_d}$ of A, the word $\beta^m\alpha\left(\beta^{i_1}\alpha\right) \dots \left(\beta^{i_d}\alpha\right)$ is a careful reset word of B. On the other hand, any careful reset word of B is also a careful reset word of the subset $Q_A \times \{a_0\} \subseteq Q_B$, whose careful reset words are in a one-to-one correspondence with careful reset words of A, like in the previous case.

Since $\delta_B((s, a_m), \alpha)$ is defined for each $s \in S_A$, it is not hard to check that if A is strongly connected, so is B.

□

2.1.5 The key construction

Let us present the central construction of this chapter. We build a series of DFA with a constant-size alphabet and a constant structure of strongly connected components, together with subsets that require strongly exponential reset words. Moreover, the pairs automaton-subset are of the special kind represented by $\mathcal{M}_{\mathrm{P}}$, so a reduction to careful synchronization of PFA, as introduced in Lemma 2.5, is possible.

Lemma 2.7. *For infinitely many $m \geq 1$ it holds that*

$$\mathrm{sub}^{\mathcal{AL}_4 \cap \mathcal{C}_2 \cap \mathcal{C}_2^{\mathrm{R}} \cap \mathcal{M}_{\mathrm{P}}}(5m + \log m + 3) \geq 2^m \cdot (\log m + 1) + 1.$$

Proof. Suppose $m = 2^k$. For each $t \in 0, \dots, m-1$ we denote by $\tau = \mathrm{bin}(t)$ the standard k-digit binary representation of t, i.e. a word from $\{\mathbf{0}, \mathbf{1}\}^k$. By a classical result proved in [36] there is a *De Bruijn sequence* $\xi = \xi_0 \dots \xi_{m-1}$ consisting of letters $\xi_i \in \{\mathbf{0}, \mathbf{1}\}$ such that each word $\tau \in \{\mathbf{0}, \mathbf{1}\}^k$ appears exactly once as a cyclic factor of ξ (i.e. it is a factor or begins by a suffix of ξ and continues by a prefix of ξ). Let us fix such ξ. By $\pi(i)$ we denote the number t, whose binary representation $\mathrm{bin}(t)$ starts by ξ_i in ξ. Note that π is a permutation of $\{0, \dots, m-1\}$. Set

$$\begin{aligned}
Q &= \left(\{0, \dots, m-1\} \times \left\{\mathbf{0}, \mathbf{0}^{\downarrow}, \mathbf{1}, \mathbf{1}^{\downarrow}, \mathbf{1}^{\uparrow}\right\}\right) \cup \left\{\mathrm{C}_0, \dots, \mathrm{C}_k, \mathrm{D}, \overline{\mathrm{D}}\right\}, \\
X &= \{\mathbf{0}, \mathbf{1}, \kappa, \omega\}, \\
S &= (\{0, \dots, m-1\} \times \{\mathbf{0}\}) \cup \{\mathrm{C}_0, \mathrm{D}\}.
\end{aligned}$$

Figure 2.1 visually distinguishes main parts of A. The states D and $\overline{\mathrm{D}}$ are sink states. Together with $\mathrm{D} \in S$ it implies that any reset word of S takes the states of S to D and that the state $\overline{\mathrm{D}}$ must not become active during its application. The states $\mathrm{C}_0, \dots, \mathrm{C}_k$ guarantee that any reset

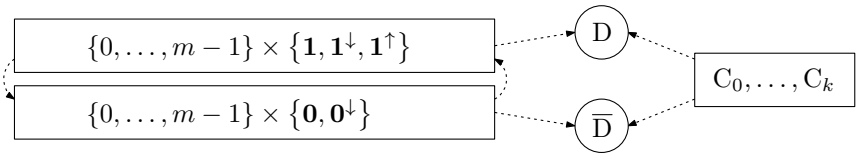


Figure 2.1: A connectivity pattern of the automaton A.

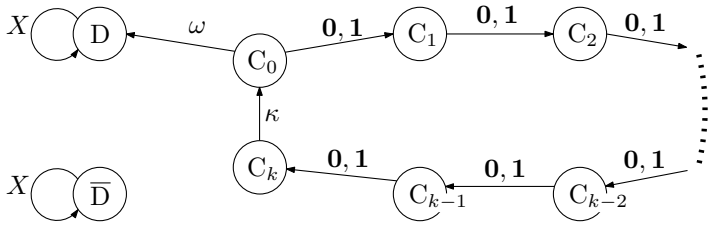


Figure 2.2: A part of A. All the outgoing transitions that are not depicted lead to $\overline{\mathrm{D}}$.

word of S lies in

$$\left(\{\mathbf{0},\mathbf{1}\}^k\kappa\right)^\star\omega X^\star. \tag{2.1}$$

Indeed, as defined by Figure 2.2, no other word takes C_0 to D. Let the letter ω act as follows:

$$\begin{aligned}\{0,\dots,m-1\}\times\{\mathbf{1}\},\mathrm{C}_0,\mathrm{D} &\xrightarrow{\omega} \mathrm{D},\\ \{0,\dots,m-1\}\times\{\mathbf{0},\mathbf{0}^{\downarrow},\mathbf{1}^{\downarrow},\mathbf{1}^{\uparrow}\},\mathrm{C}_1,\dots,\mathrm{C}_{\log m},\overline{\mathrm{D}} &\xrightarrow{\omega} \overline{\mathrm{D}}.\end{aligned}$$

We see that ω maps each state to D or $\overline{\mathrm{D}}$. This implies that once ω occurs in a reset word of S, it must complete the synchronization. In order to map C_0 to D, the letter ω *must* occur, so any shortest reset word of S is exactly of the form

$$w=(\tau_1\kappa)\dots(\tau_d\kappa)\,\omega, \tag{2.2}$$

where $\tau_j\in\{\mathbf{0},\mathbf{1}\}^k$ for each j.

The two biggest parts depicted by Figure 2.1 are very similar to each other. The letters $\mathbf{0}$ and $\mathbf{1}$ act on them as follows:

$$(i,\mathbf{0})\xrightarrow{\mathbf{0}}\begin{cases}(i+1,\mathbf{0}) & \text{if } \xi_i=\mathbf{0}\\ (i+1,\mathbf{0}^{\downarrow}) & \text{if } \xi_i=\mathbf{1}\end{cases}\qquad (i,\mathbf{1})\xrightarrow{\mathbf{0}}\begin{cases}(i+1,\mathbf{1}) & \text{if } \xi_i=\mathbf{0}\\ (i+1,\mathbf{1}^{\downarrow}) & \text{if } \xi_i=\mathbf{1}\end{cases}$$

$$(i,\mathbf{0})\xrightarrow{\mathbf{1}}\begin{cases}\overline{\mathrm{D}} & \text{if } \xi_i=\mathbf{0}\\ (i+1,\mathbf{0}) & \text{if } \xi_i=\mathbf{1}\end{cases}\qquad (i,\mathbf{1})\xrightarrow{\mathbf{1}}\begin{cases}(i+1,\mathbf{1}^{\uparrow}) & \text{if } \xi_i=\mathbf{0}\\ (i+1,\mathbf{1}) & \text{if } \xi_i=\mathbf{1}\end{cases}$$

and $(i,\mathbf{b})\xrightarrow{\mathbf{0},\mathbf{1}}(i+1,\mathbf{b})$ for each $\mathbf{b}=\mathbf{0}^{\downarrow},\mathbf{1}^{\downarrow},\mathbf{1}^{\uparrow}$, using the addition modulo m everywhere. For example, Figure 2.3 depicts a part of A for $m=8$ and for a particular De Bruijn sequence ξ. Figure 2.4 defines the action of κ on the states $\{i\}\times\{\mathbf{0},\mathbf{0}^{\downarrow},\mathbf{1},\mathbf{1}^{\downarrow},\mathbf{1}^{\uparrow}\}$ for any i, so the automaton A is completely defined.

Let w be a shortest reset word of S in A. It is necessarily of the form (2.2), so it makes sense to denote $v_t=\mathrm{bin}(t)\,\kappa$ and treat w as

$$w=v_{t_1}\dots v_{t_d}\omega\in\{v_0,\dots,v_{m-1},\omega\}^\star. \tag{2.3}$$

The action of each v_t is depicted by Figure 2.5. It is a key step of the proof to confirm that Figure 2.5 is correct. Indeed:

- Starting from a state $(i,\mathbf{1})$, a word $\mathrm{bin}(t)$ takes us through a kind of decision tree to one

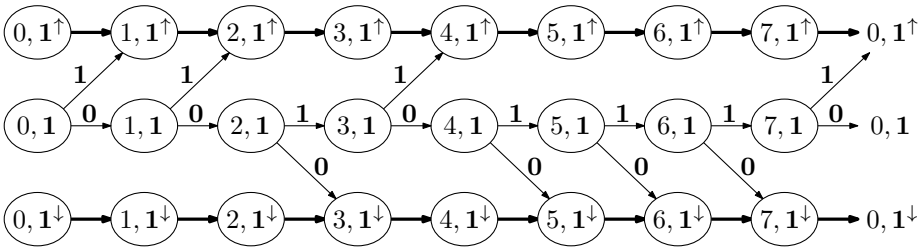


Figure 2.3: A part of A assuming $m = 8$ and $\xi = \mathbf{00101110}$. Bold arrows represent both $\mathbf{0}, \mathbf{1}$.

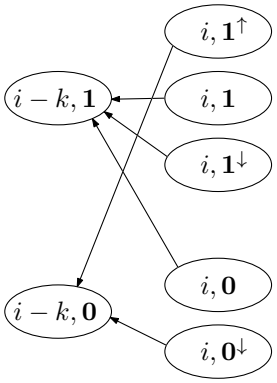


Figure 2.4: The action of the letter κ, with subtraction modulo m.

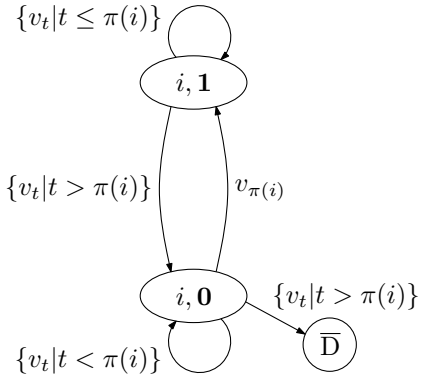


Figure 2.5: The action of $v_0, \ldots, v_{m-1}$ on the i-th switch.

of the states $\left(i+k, \mathbf{1}^{\downarrow}\right), (i+k, \mathbf{1}), \left(i+k, \mathbf{1}^{\uparrow}\right)$, depending on whether t is lesser, equal, or greater than $\pi(i)$, respectively. This is guaranteed by wiring the sequence ξ into the transition function, see Figure 2.3. The letter κ then take us back to $\{i\} \times \{\dots\}$, namely to $(i, \mathbf{0})$ or $(i, \mathbf{1})$.

- Starting from a state $(i, \mathbf{0})$, we proceed similarly, but in the case of $t > \pi(i)$ we fall into $\overline{\mathrm{D}}$ during the application of $\mathrm{bin}(t)$.

It follows that after applying any prefix $v_{t_1} \dots v_{t_j}$ of w, exactly one of the states $(i, \mathbf{0}), (i, \mathbf{1})$ is active for each i. We say that *the i-th switch is set to* $\mathbf{0}$ *or* $\mathbf{1}$ *at time* j. Note that $Q_A \setminus \left\{\overline{\mathrm{D}}\right\}$ is the S-relevant part of A and that the sets $\{i\} \times \left\{\mathbf{0}, \mathbf{0}^{\downarrow}, \mathbf{1}, \mathbf{1}^{\downarrow}, \mathbf{1}^{\uparrow}\right\}$ for $i = 0, \dots, m-1$, together with the sets $\{\mathrm{D}\}$ and $\{\mathrm{C}_0, \dots, \mathrm{C}_k\}$, can play the role of $P_1, \dots, P_{m+2}$ in the definition of $\mathcal{M}_{\mathrm{P}}$. Observe that at time d all the switches are necessarily set to $\mathbf{1}$ because otherwise the state $\overline{\mathrm{D}}$ would become active by the application of ω. On the other hand, at time 0 all the switches are set to $\mathbf{0}$. We are going to show that in fact during the synchronization of S the switches together perform a binary counting from 0 (all the switches set to $\mathbf{0}$) to $2^m - 1$ (all the switches set to $\mathbf{1}$). For each i the *significance* of the i-th switch is given by the value $\pi(i)$. So the $\pi^{-1}(m-1)$-th switch carries the most significant digit, the $\pi^{-1}(0)$-th switch carries the least significant digit and so on. The number represented in this manner by the switches at time j is denoted by $\mathfrak{b}_j \in \{0, \dots, 2^m - 1\}$. We claim that $\mathfrak{b}_j = j$ for each j. Indeed:

- At time 0, all the switches are set to $\mathbf{0}$, we have $\mathfrak{b}_0 = 0$.
- Suppose that $\mathfrak{b}_{j'} = j'$ for each $j' \leq j - 1$. We denote

$$\overline{t_j} = \min \{\pi(i) \mid i\text{-th switch is set to } \mathbf{0} \text{ at time } j-1\} \tag{2.4}$$

and claim that $t_j = \overline{t_j}$. Note that $\overline{t_j}$ is defined to be the least significance level at which there occurs a $\mathbf{0}$ in the binary representation of $\mathfrak{b}_{j-1}$. Suppose for a contradiction that $t_j > \overline{t_j}$. By the definition of $\overline{t_j}$ the state $\left(\pi^{-1}\left(\overline{t_j}\right), \mathbf{0}\right)$ lies in $\delta\left(S, v_{t_1} \dots v_{t_{j-1}}\right)$. But v_{t_j}

takes this state to $\overline{\mathrm{D}}$, which is a contradiction. Now suppose that $t_j < \overline{t_j}$. In such case the application of v_{t_j} does not turn any switch from **0** to **1**, so $\mathfrak{b}_j \leq \mathfrak{b}_{j-1}$ and thus at time j the configuration of switches is the same at it was at time $\mathfrak{b}_j$. This contradicts the assumption that w is a shortest reset word. We have proved that $t_j = \overline{t_j}$ and it remains only to show that the application of v_{t_j} performs the addition of 1 and so makes the switches represent the value $\mathfrak{b}_{j-1} + 1$.

- Consider an i-th switch with $\pi(i) < t_j$. By the definition of $\overline{t_j}$, it is set to **1** at time $j-1$ and the word v_{t_j} sets it to **0** at time j. This is what we need because such switches represent a continuous leading segment of **1**s in the binary representation of $\mathfrak{b}_{j-1}$.
- The $\pi^{-1}(t_j)$-th switch is set from **0** to **1** by the word v_{t_j}.
- Consider an i-th switch with $\pi(i) > t_j$. The switch represents a digit of $\mathfrak{b}_{j-1}$ which is more significant than the $\overline{t_j}$-th digit. As we expect, the word v_{t_j} leaves such switch unchanged.

Because $\mathfrak{b}_d = 2^m$, we deduce that $d = 2^m$ and thus $|w| = 2^m \cdot (k+1) + 1$, assuming that a shortest reset word w exists. But in fact we have also shown that there is only one possibility for such w and that it is a true reset word for S. The unique w is of the form (2.3), where t_j is the position of the least significant **0** in the binary representation of $j-1$.
The automaton A lies in $\mathcal{C}_2 \cap \mathcal{C}_2^{\mathrm{R}}$ since the addition of $\mathrm{D} \longrightarrow \mathrm{C}_0$ and $\overline{\mathrm{D}} \longrightarrow (0, \mathbf{0})$ makes A strongly connected, while the addition of $\mathrm{D} \longrightarrow \mathrm{C}_0$ and $\mathrm{C}_0 \longrightarrow (0, \mathbf{0})$ makes $R_{A,S}$ strongly connected. □

2.1.6 The results

The following theorem presents the main results of the present paper:

Theorem 2.8. *For infinitely many $n \geq 1$ it holds that*

1. $\mathrm{sub}^{\mathcal{AL}_2 \cap SC}(n) \geq 2^{\frac{n}{61}}$,
2. $\mathrm{car}^{\mathcal{AL}_2 \cap \mathcal{SC}}(n) \geq 2^{\frac{n}{36}}$.

Proof. Lemma 2.7 says that

$$2^m \cdot (\log m + 1) + 1 \quad \leq \quad \mathrm{sub}^{\mathcal{AL}_4 \cap \mathcal{C}_2 \cap \mathcal{C}_2^{\mathrm{R}} \cap \mathcal{M}_{\mathrm{P}}}(5m + \log m + 3) \tag{2.5}$$

for infinitely many $m \geq 1$. Now we apply some of the lemmas from the above sections:

1. Lemma 2.4 extends (2.5) with

   $$\mathrm{sub}^{\mathcal{AL}_4 \cap \mathcal{C}_2}(5m + \log m + 3) \leq \mathrm{sub}^{\mathcal{AL}_6 \cap \mathcal{SC}}(10m + 2 \cdot \log m + 8) - 1$$

   and Lemma 2.6 adds

   $$\mathrm{sub}^{\mathcal{AL}_6 \cap \mathcal{SC}}(10m + 2 \cdot \log m + 8) - 1 \leq \mathrm{sub}^{\mathcal{AL}_2 \cap SC}(60m + 12 \cdot \log m + 48) - 1.$$

   We chain the three inequalities and deduce

   $$\begin{aligned} \mathrm{sub}^{\mathcal{AL}_2 \cap SC}(60m + 12 \cdot \log m + 48) &\geq 2^m \cdot (\log m + 1) + 2, \\ \mathrm{sub}^{\mathcal{AL}_2 \cap SC}(61m) &\geq 2^m, \\ \mathrm{sub}^{\mathcal{AL}_2 \cap SC}(n) &\geq 2^{\frac{n}{61}}. \end{aligned}$$

2. Lemma 2.5 extends (2.5) with

$$\mathrm{csub}^{\mathcal{AL}_4\cap\mathcal{C}_2^{\mathrm{R}}\cap\mathcal{M}_{\mathrm{P}}}(5m+\log m+3)\leq \mathrm{car}^{\mathcal{AL}_5\cap\mathcal{C}_2}(5m+\log m+3)\,,$$

while Lemma 2.2 adds

$$\mathrm{car}^{\mathcal{AL}_5\cap\mathcal{C}_2}(5m+\log m+3)\leq \mathrm{car}^{\mathcal{AL}_7\cap\mathcal{SC}}(5m+\log m+3)$$

and Lemma 2.6 adds

$$\mathrm{car}^{\mathcal{AL}_7\cap\mathcal{SC}}(5m+\log m+3)\leq \mathrm{car}^{\mathcal{AL}_2\cap SC}(35m+7\cdot\log m+21)\,.$$

We chain the four inequalities and deduce:

$$\begin{aligned}\mathrm{car}^{\mathcal{AL}_2\cap SC}(35m+7\cdot\log m+21) &\geq 2^m\cdot(\log m+1)+1,\\ \mathrm{car}^{\mathcal{AL}_2\cap SC}(36m) &\geq 2^m,\\ \mathrm{car}^{\mathcal{AL}_2\cap SC}(n) &\geq 2^{\frac{n}{36}}.\end{aligned}$$

□

Note that there are more subtle results for less restricted classes of automata:

Proposition 2.9. *It holds that* $\mathrm{sub}^{\mathcal{AL}_2}(n)\geq 2^{\frac{n}{21}}$, $\mathrm{car}^{\mathcal{AL}_2}(n)\geq 2^{\frac{n}{26}}$, $\mathrm{sub}^{SC}(n)\geq 3^{\frac{n}{6}}$, *and* $\mathrm{car}^{\mathcal{SC}}(n)\geq 3^{\frac{n}{3}}$ *for infinitely many* $n\geq 1$.

Proof. The first claim follows easily from Lemmas 2.7 and 2.6, the second one requires also using Lemma 2.5 first. The third and the last claim follow from applying Lemmas 2.1 and 2.4 (or Lemma 2.2 respectively) to the construction from [59]. □

2.1.7 An Application to Computational Complexity

Theorem 2.10. *The following problems are PSPACE-complete:*

1. Subset synchronizability restricted to binary strongly connected DFA
2. Careful synchronizability restricted to binary strongly connected PFA

Proof. There are polynomial reductions from the general problems Subset synchronizability and Careful synchronizability: Perform the construction from Lemma 2.4 (or Lemma 2.2 respectively) and then the one from Lemma 2.6. □

2.2 Synchronization Thresholds of Automata with Sink States

In this short section we leave the field of subset synchronization and careful synchronization and we inspect a more basic field - synchronization of DFA with a sink state[1], see Definition 1.5. First, we formulate the following easy observations:

Lemma 2.11. *Let* $A=(Q,\Sigma,\delta)\in\mathcal{Z}$ *be a DFA with a sink state* $q_0\in Q$.

1. *If* A *is synchronizing, then* q_0 *is the only sink state and* $\delta(Q,w)=\{q_0\}$ *for each reset word* w.
2. A *is synchronizing if and only if* q_0 *is reachable from each* $r\in Q$.

[1]The questions can be equivalently expressed in terms of *annulation* of PFA, see e.g. [5].

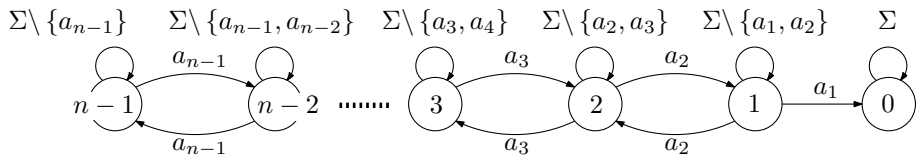


Figure 2.6: The automaton A_n from Theorem 2.12

Automata with sink states seem much easier to synchronize. Informally, once a state is mapped to q_0, it cannot be re-mapped anywhere else. However, it turns out that some questions about lengths of shortest reset words remain hard, especially concerning alphabet size. For the whole class $\mathcal{Z}$ the exact synchronization threshold has been already found by Rystsov:

Theorem 2.12 ([80])**.** *For each $n \geq 1$ it holds that $\mathrm{C}_n = \frac{n(n-1)}{2}$.*

The proof uses a series of DFA with growing alphabet size. Specifically, for each $n \geq 1$ there is the following automaton $A_n = (Q, \Sigma, \delta)$ with $\mathrm{C}(A) = \frac{n(n-1)}{2}$:

$$\begin{aligned} Q &= \{0, \ldots, n-1\}, \\ \Sigma &= \{a_1, \ldots, a_{n-1}\}, \\ \delta(s,x) &= \begin{cases} s-1 & \text{if } s > 0 \text{ and } x = a_s, \\ s+1 & \text{if } s < n-1,\ s \neq 0, \text{ and } x = a_{s+1}, \\ s & \text{otherwise,} \end{cases} \end{aligned}$$

see also Figure 2.6.

On the other hand, in [57], Martyugin presents the lower bound $\left\lceil \frac{n^2+6n-16}{4} \right\rceil$ that holds also with the restriction to binary DFA. Besides of that, he provided a single isolated example of a 10-state automaton that requires resets words of length 37, thus exceeding the bound above.

In the following we introduce an infinite series of automata admitting the Martyugin's example as an initial case. This consists of a series $B_1, B_2, \ldots$ of DFA with sink states, together with a general construction of $A\langle k, r\rangle$, which informally append a *tail* of length k to a sink state of a DFA $A \in \mathcal{Z}$, depending on a given non-sink state r of A. Using such notation, the Martyugin's 10-state example is equal to $B_1\langle 3, q_1\rangle$, where q_1 is a particular state of B_1.

The following table lists the first six automata in our series. It was computationally verified that their synchronization threshold exceeds $\left\lceil \frac{n^2+6n-16}{4} \right\rceil$, including the Martyugin's one. As our automata have even number of states, we write $\frac{1}{4}n^2 + \frac{3}{2}n - 4$ instead of $\left\lceil \frac{n^2+6n-16}{4} \right\rceil$. Software, hardware, and assistance for the computations were kindly provided by Marek Szykuła.

automaton A	number n of states	$\frac{1}{4}n^2 + \frac{3}{2}n - 4$	synchronization threshold of A
$B_1\langle 3, q_1\rangle$	10	36	37
$B_2\langle 9, q_1\rangle$	22	150	151
$B_3\langle 15, q_1\rangle$	34	336	337
$B_4\langle 21, q_1\rangle$	46	594	595
$B_5\langle 27, q_1\rangle$	58	924	925
$B_6\langle 33, q_1\rangle$	70	1326	1327

The table suggests the following conjecture:

Conjecture 2.13. *For each $j \geq 1$ it holds that $\mathrm{C}(B_j\langle 6j-3, q_1\rangle) = \frac{1}{4}n^2 + \frac{3}{2}n - 3$, where $n = 12j - 2$.*

Let us define all the notation mentioned above. First, we describe a general concept of adding a *tail* of length $k \geq 1$ to an automaton with a sink state. Let $A = (Q, \{a, b\}, \delta)$ be a binary DFA with a unique sink state $q_0 \in Q$. Then for each $k \geq 0$ and each $r \in Q$ the term $A\langle k, r\rangle$ stands for the following automaton $(Q', \{a, b\}, \delta')$ with k additional states:

$$\begin{aligned} Q' &= Q \cup \{t_0, t_1, \ldots, t_{k-1}\}, \\ \delta'(s,a) &= \begin{cases} \delta(s,a) & \text{if } s \in Q \setminus \{q_0\} \\ t_{k-1} & \text{if } s = q_0 \\ t_0 & \text{if } s = t_0, \\ t_{i-1} & \text{if } s \in \{t_1, \ldots, t_{k-1}\}, \end{cases} \\ \delta'(s,b) &= \begin{cases} \delta(s,b) & \text{if } s \in Q \setminus \{q_0\}, \\ r & \text{otherwise,} \end{cases} \end{aligned}$$

for each $s \in Q'$. Observe that t_0 is the new unique sink state. The key property of such transformation is the following:

Lemma 2.14. *Let $A = (Q, \{a, b\}, \delta)$ be a synchronizing binary DFA with a sink state $q_0 \in Q$ and $k \in \mathbb{N}$. Suppose that:*

1. *a is a permutation,*
2. *$k \geq 1$ is a common multiple of the lengths of cycles of a,*
3. *$r \in Q$ is the only non-sink state with $\delta(r, b) = q_0$,*
4. *b is a permutation of $Q \setminus \{r\}$.*

Then $\mathrm{C}(A\langle k, r\rangle) = \mathrm{C}(A) + nk$, where $n = |Q|$.

Proof. First, we show that $\mathrm{C}(A\langle k, r\rangle) \leq \mathrm{C}(A) + nk$. Let w be a reset word of A. For each $d \geq 1$ we denote by u_d the shortest prefix u of w satisfying that $\left|\delta^{-1}(\{q_0\}, u)\right| \geq d$, which means that u maps at least d states to q_0. Clearly, $u_1 = \epsilon$ and $u_n = w$. Denote

$$w = v_2 \ldots v_n,$$

where $u_d = v_2 \ldots v_d$ for each $2 \leq d \leq n$. Let $w' = a^k v_2 a^k v_3 a^k \ldots v_n a^k$. It is enough to show that w' synchronizes $A\langle k, r\rangle$, i.e. $\delta'(s, w') = t_0$ for each $s \in Q'$. If $s \in \{t_0, t_1, \ldots, t_{k-1}\}$, we just observe that $\delta'(s, a^k) = t_0$. If $s \in Q$, let d be the least integer such that $\delta(s, u_d) = q_0$. Since $\delta(s, u') \neq q_0$ for each proper prefix u' of u_d, the definition of δ' implies that

$$\delta'(s, u_d) = \delta(s, u_d) = q_0.$$

As $\delta'(q_0, a^k) = t_0$, we are done.
Second, we show that $\mathrm{C}(A) + nk \leq \mathrm{C}(A\langle k, r\rangle)$. Let w' be a reset word of $A\langle k, r\rangle$. For each $s \in Q'$, denote by u'_s the shortest prefix u' of w' with $\delta'(s, u') = t_0$.

1. Observe that whenever $s \in Q$, the word u'_s must be of the form $u'_s = v'_s a^k$ for some v'_s with $\delta'(s, v'_s) = q_0$. Moreover, as $q_0 \notin \mathrm{rng}(a)$, each v'_s ends by b or is empty.

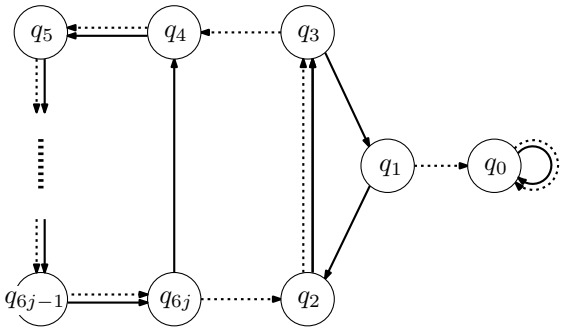


Figure 2.7: The automaton B_j - solid marks a, dotted marks b

2. Next, we show that $v'_p \neq v'_s$ for each distinct $p, s \in Q$. Otherwise, we have $\delta(p, v') = \delta(s, v') = q_0$, where $v' = v'_p = v'_s$. As r is the only merging state in $A\langle k, r\rangle$ except for t_0, we have

$$\delta(p, v''b) = \delta(s, v''b) = r,$$

for some prefix v'' of v' with $\delta(p, v'') \neq \delta(s, v'')$. Because

$$\delta(p, v''), \delta(s, v'') \in \delta^{-1}(r, b) = \{t_0, t_1, \ldots, t_{k-1}, q_0\},$$

we can denote $t_i = \delta(p, v'')$ and $t_j = \delta(s, v'')$, where $i, j \in \{0, \ldots, k\}$ and t_k stands for q_0. As $p \neq s$, we can suppose that $p \neq q_0$ (the case $s \neq q_0$ is symmetrical). From $t_i = \delta(p, v'')$ it follows that v'' ends with ba^i. From $t_j = \delta(s, v'')$ it follows that v'' ends with ba^j or is equal to a^j. As $i \neq j$, we get a contradiction.

3. Together, each of the distinct prefixes v'_s of w' for $s \in Q$ ends by b and is followed by a^k. Thus, w' contains at least n disjoint occurrences of the factor a^k.

Let w be obtained from w' by deleting these factors, so we have $|w| \leq |w'| - nk$. It remains to show that w is a reset word of A. Choose $s \in Q$ and let w'_s be the shortest prefix u' of w' with $\delta'(s, u') = q_0$. As $q_0 \notin \mathrm{rng}(a)$, the word w'_s ends by b or is empty. Thus we can consider the prefix w_s of w obtained by deleting the occurrences of a^k from w'_s. As k is a common multiple of the lengths of cycles of a, the word a^k acts as identity on $Q \setminus \{q_0\}$ in both $A\langle k, r\rangle$ and A. We conclude that

$$\delta(s, w_s) = \delta(s, w'_s) = \delta'(s, w'_s) = q_0,$$

which implies easily that $\delta(s, w) = q_0$.

□

Now we define the infinite series B_j that, after adding tails, constitutes the new examples of automata with high synchronization threshold.
For each $j \geq 1$, we fix an automaton $B_j = (Q, \{a, b\}, \delta)$ with $Q = \{q_0, q_1, \ldots, q_{6j}\}$ according to the following table, where each $q_i \in Q$ is shortened to i:

s	0	1	2	3	4	5	$\cdots$	$6j-1$	$6j$
$\delta(s, a)$	0	2	3	1	5	6	$\cdots$	$6j$	4
$\delta(s, b)$	0	0	3	4	5	6	$\cdots$	$6j$	2

The automaton is also depicted in Figure 2.7. The key idea of this chapter is to add a tail of length $6j - 3$ to each B_j:

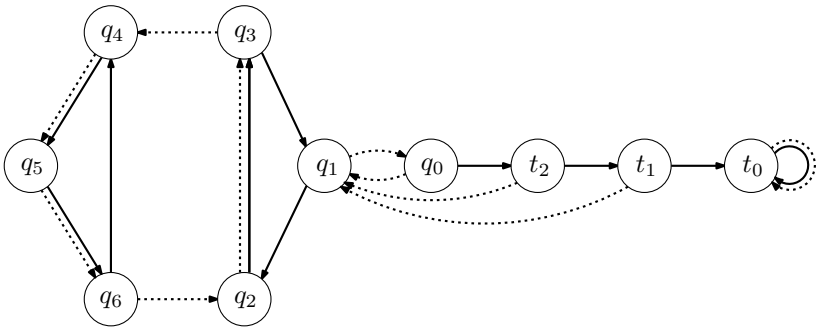


Figure 2.8: The Martyugin's automaton $B_1\langle 3, q_1\rangle$

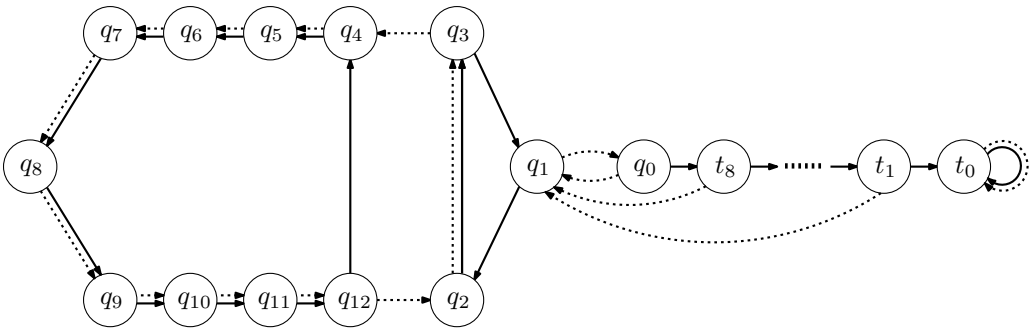


Figure 2.9: The automaton $B_2\langle 9, q_1\rangle$

Lemma 2.15. *For each $j \geq 1$ it holds that*

$$\mathrm{C}(B_j\langle 6j-3, q_1\rangle) = \mathrm{C}(B_j) + 36j^2 - 12j - 3.$$

Proof. We apply Lemma 2.14. The letter a is indeed a permutation and its cycles have lengths 3 and $6j-3$, so we can use $k = 6j-3$. We also observe that $r = q_1$ meets the last two conditions of the lemma. As B_j has $n = 6j+1$ states, it remains to verify that $nk = (6j+1)(6j-3) = 36j^2 - 12j - 3$. □

See Figures 2.8 and 2.9 that depict the automata $B_1\langle 3, q_1\rangle$ and $B_2\langle 9, q_1\rangle$. It was computationally verified that $\mathrm{C}(B_j) = 18j - 2$ for $j = 1, 2, 3, 4, 5, 6$. Thus, due to Lemma 2.15, $\mathrm{C}(B_j\langle 6j-3, q_1\rangle) = 36j^2 + 6j - 5$ for each such j. The automaton $\mathrm{C}(B_j\langle 6j-3, q_1\rangle)$ has $n = 12j - 2$ states, so we can compute that $36j^2 + 6j - 5 = \frac{1}{4}n^2 + \frac{3}{2}n - 3$. Thus, the automata $B_j\langle 6j-3, q_1\rangle$ prove Conjecture 2.13 for $1 \leq j \leq 6$ and witness the following theorem:

Theorem 2.16. *For each $1 \leq j \leq 6$ and $n = 12j - 6$, it holds that $\mathrm{C}_n \geq \frac{1}{4}n^2 + \frac{3}{2}n - 3$.*

Besides the five new computationally verified examples exceeding the bound $\left\lceil \frac{n^2+6n-16}{4} \right\rceil$, the main contribution of this section lies in pointing out that it seems very likely for the whole series $B_j\langle 6j-3, q_1\rangle$ to exceed this bound.

Chapter 3

Computing Synchronization Thresholds in DFA

This chapter presents two results about the classical decision problem SYN introduced in Section 1.5.1. First, we prove that unless polynomial hierarchy collapses, SYN does not have a polynomial kernel if parameterized by the number of states. This concludes a research of Fernau, Heggernes, and Villanger, 2013 [34] (in the latest version of this article [35], our result is already cited). A paper [105] containing the proof was accepted for publishing in *Discrete Mathematics and Theoretical Computer Science*.
Second, we confirm NP-completeness of SYN restricted to Eulerian automata with binary alphabets, as it was conjectured by Martyugin, 2011 [60]. The proof was presented [102] at the conference *LATA 2014* (Madrid, Spain), and an extended paper was submitted to a journal.

3.1 Parameterized Complexity of SYN

The result of this section and the former results of Fernau, Heggernes, and Villanger [34, 35] are summarized by Table 3.1. We have filled the last remaining gap in the corresponding table in [34, Sec. 3]. Thus, the multi-parameter analysis of SYN is complete in the sense that NP-complete restrictions are identified and under several standard assumptions we know which restrictions are FPT and which of them have polynomial kernels.
The following lemma, which is easy to prove using the construction of a power automaton, says that Syn lies in FPT if parameterized by the number of states:

Parameter	**Parameterized Complexity of SYN**	**Polynomial Kernel of SYN**
d	W[2]-hard [34]	—
$\|\Sigma\|$	NP-complete for $\|\Sigma\| = 2, 3, \ldots$ [33]	—
d and $\|\Sigma\|$	FPT, running time $\mathcal{O}^\star(\|\Sigma\|^d)$ [triv.]	Not unless NP $\subseteq$ coNP/poly [34]
$n = \|Q\|$	FPT, running time $\mathcal{O}^\star(2^n)$ [triv.]	Not unless PH collapses ♦

Table 3.1: Results of the complete multi-parameter analysis of SYN and SRCP. Diamonds mark the results of the present paper

Lemma 3.1 ([34, 82])**.** *There exists an algorithm that solves* SYN *in time* $r(n, |\Sigma|) \cdot 2^n$ *for an appropriate polynomial* r*.*

But does there exist a polynomial kernel? In this section we use methods developed by Bodlaender et al. [20] to prove the following:

Theorem 3.2. *If* Syn *parameterized by the number of states has a polynomial kernel, then* $\mathrm{PH} = \Sigma_{\mathrm{p}}^3$.

By PH we denote the union of the entire polynomial hierarchy, so $\mathrm{PH} = \Sigma_{\mathrm{p}}^3$ means that polynomial hierarchy collapses into the third level, which is widely assumed to be false. The key proof method relies on *composition algorithms*. In order to use them immediately, we introduce the formalization of our parameterized problem as a set of string-integer pairs:

$$L_{\mathrm{SYN}} = \{(x, n) \mid x \in \Sigma^\star \text{ encodes an instance of SYN with } n \in \mathbb{N} \text{ states}\},$$

where Σ is an appropriate finite alphabet.

3.1.1 Composition Algorithms

An *or-composition algorithm* for a parameterized problem $L \subseteq \Sigma^\star \times \mathbb{N}$ is an algorithm that

- receives as input a sequence $((x_1, n), \ldots, (x_m, n))$ with $(x_i, n) \in \Sigma^\star \times \mathbb{N}^+$ for each $1 \leq i \leq m$,
- uses time polynomial in $\sum_{i=1}^m |x_i| + n$
- outputs $(y, n') \subseteq \Sigma^\star \times \mathbb{N}^+$ with
  1. $(y, n') \in L \Leftrightarrow$ there is some $1 \leq i \leq m$ with $(x_i, n) \in L$,
  2. n' is polynomial in n.

Let $L \subseteq \Sigma^\star \times \mathbb{N}$ be a parameterized problem. Its *unparameterized version* is

$$\widehat{L} = \{x \# a^n \mid (x, n) \in L\},$$

where $\# \notin \Sigma$ is a special symbol.

Theorem 3.3 ([20])**.** *Let* L *be a parameterized problem having an or-composition algorithm. Assume that its unparameterized version* $\widehat{L}$ *is* NP*-complete. If* L *has a polynomial kernel, then* $\mathrm{PH} = \Sigma_{\mathrm{p}}^3$.

The unparameterized version of L_{SYN} is computationally as hard as the classical SYN, so it is NP-complete. It remains only to describe an or-composition algorithm for L_{SYN}, which is done in the remainder of this section. For each $n \in \mathbb{N}$ we denote $z(n) = \frac{n^3 - n}{6}$, which is the Pin's upper bound for lengths of shortest reset words, see Theorem 1.15.

3.1.2 Preprocessing

Let the or-composition algorithm receive an input

$$((A_1, d_1), n), \ldots, ((A_m, d_m), n)$$

consisting of n-state automata $A_1, \ldots, A_m$, each of them equipped with a number d_i. Assume that the following easy procedures have been already applied:

- For each $i = 1, \ldots, m$ such that $d_i \geq z(n)$, use the polynomial-time synchronizability algorithm from Corollary 1.33 to decide whether $((A_i, d_i), n) \in L_{\mathrm{SYN}}$. If so, return a trivial true instance immediately. Otherwise just delete the i-th member from the sequence.

- For each $i = 1, \ldots, m$, add an additional letter κ to the automaton A_i such that κ acts as the identical mapping: $\delta_i(s, \kappa) = s$.

- For each $i = 1, \ldots, m$ rename the states and letters of A_i such that

$$\begin{aligned} A_i &= (Q_i, I_i, \delta_i), \\ Q_i &= \{1, \ldots, n\}, \\ I_i &= \{\kappa, a_{i,1}, \ldots, a_{i,|I_i|-1}\}. \end{aligned}$$

After that, our algorithm chooses one of the following procedures according to the length m of the input sequence:

- If $m \geq 2^n$, use the exponential-time algorithm from Lemma 3.1: Denote $D = \sum_{i=1}^{m} |(A_i, d_i)| + n$, where we add lengths of descriptions of the pairs. Note that $D \geq m \geq 2^n$ and that D is the quantity used to restrict the running time of or-composition algorithms. By the lemma, in time

$$\sum_{i=1}^{m} r(n, |I_i|) \cdot 2^n \leq m \cdot r(D, D) \cdot 2^n \leq D^2 \cdot r(D, D)$$

  we are able to analyze all the m automata and decide if some of them have a reset word of the corresponding length. It remains just to output some appropriate trivial instance $((A', d'), n')$.

- If $m < 2^n$, we denote $q(m) = \lfloor \log(m+1) \rfloor$. It follows that $q(m) \leq n + 2$. On the output of the or-composition algorithm we put $((A', d'), n')$, where A' is the automaton described in the following paragraphs and

$$d' = z(n) + 1$$

  is our choice of the maximal length of reset words to be found in A'.

3.1.3 Construction of A' and Its Ideas

Here we describe the automaton A' that appears in the output of our or-composition algorithm. We set

$$\begin{aligned} A' &= (Q', I', \delta'), \\ Q' &= \{1, \ldots, n\} \cup \{\mathrm{D}\} \cup (\{0, \ldots, z(n)\} \times \{0, \ldots, q(m)\} \times \{\mathrm{T}, \mathrm{F}\}), \\ \Sigma' &= \left(\bigcup_{i=1}^{m} \Sigma_i\right) \cup \{\alpha_1, \ldots, \alpha_m\} \cup \{\omega_1, \ldots, \omega_n\}. \end{aligned}$$

On the states $\{1, \ldots, n\}$ the letters from $\bigcup_{i=1}^{m} \Sigma_i$ act simply:

$$s \xrightarrow{x_{i,j}} \delta_i(s, x_{i,j})$$

for each $s = 1, \ldots, n$, $i = 1, \ldots, m$, $j = 1, \ldots, |\Sigma_i|$. In other words, we let all the letters from all the automata $A_1, \ldots, A_m$ act on the states $1, \ldots, n$ just as they did in the original automata.

The additional letters act on $\{1,\dots,n\}$ simply as well:

$$s \xrightarrow{\alpha_i} s \qquad\qquad \overline{s} \xrightarrow{\omega_s} \begin{cases} \mathrm{D} & \text{if } \overline{s} = s \\ \overline{s} & \text{otherwise.} \end{cases}$$

for each $s, \overline{s} = 1,\dots,n$, $i = 1,\dots,m$. The state D is a *sink state*, which means that

$$\mathrm{D} \xrightarrow{y} \mathrm{D}$$

for each $y \in \Sigma'$. Note that any reset word of A' have to map all the states of Q' to D.
The remaining $2 \cdot (z(n)+1) \cdot (q(m)+1)$ states form what we call a *guard table*. Its purpose is to guarantee that:

(C1) Any reset word of A' has to be of length at least $d' = z(n)+1$.

(C2) Any reset word w of A', having length exactly $z(n)+1$, is of the form

$$w = \alpha_i y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i} \omega_s \tag{3.1}$$

for some $i \in \{1,\dots,m\}$, $y_1,\dots,y_{d_i} \in \Sigma_i$, and $s \in \{1,\dots,n\}$, such that $y_1 \dots y_{d_i}$ is a reset word of A_i.

(C3) Any word w

- of length $d' = z(n)+1$,
- of the form (3.1),
- and satisfying $\delta_i(Q_i, y_1 \dots y_{d_i}) = \{s\}$

is a reset word of A'.

If the guard table manages to guarantee these three properties of A', we are done: Is is easy to check that they imply all the conditions given in the definition of a composition algorithm. So, let us define the action of the letters from Σ' on the states from $\{0,\dots,z(n)\} \times \{0,\dots,q(m)\} \times \{\mathrm{T},\mathrm{F}\}$. After that the automaton A' will be complete and we will check the properties C1, C2, C3.
The actions of the letters $\alpha_1,\dots,\alpha_m$ should meet the following two conditions:

- Any reset word w of length $z(n)+1$ has to start by some α_i.
- In such short reset word, right after the starting α_i, there must occur at least $z(n)-1$ consecutive letters from Σ_i. Informally, by applying α_i we *choose* the automaton A_i.

How to do that? The number m may be quite large and each of $\alpha_1,\dots,\alpha_m$ needs to have a unique effect. The key tool is what we call *activity patterns*. Let us work with the set

$$R = \{0,\dots,q(m)\},$$

which matches „half of a row" of the guard table. Subsets of R correspond in a canonical way to binary representations of numbers $0,\dots,2^{q(m)+1}-1$. We will actually represent only the numbers $1,\dots,m$. These does not include any of the extreme values corresponding to the empty set and whole R, because we have $m < 2^{q(m)+1}-1$. So let the mapping

$$\mathfrak{b} : \{1,\dots,m\} \to 2^R$$

assign the corresponding subset of R to a number. For instance, it holds that

$$\mathfrak{b}(11) = \{0,1,3\}$$

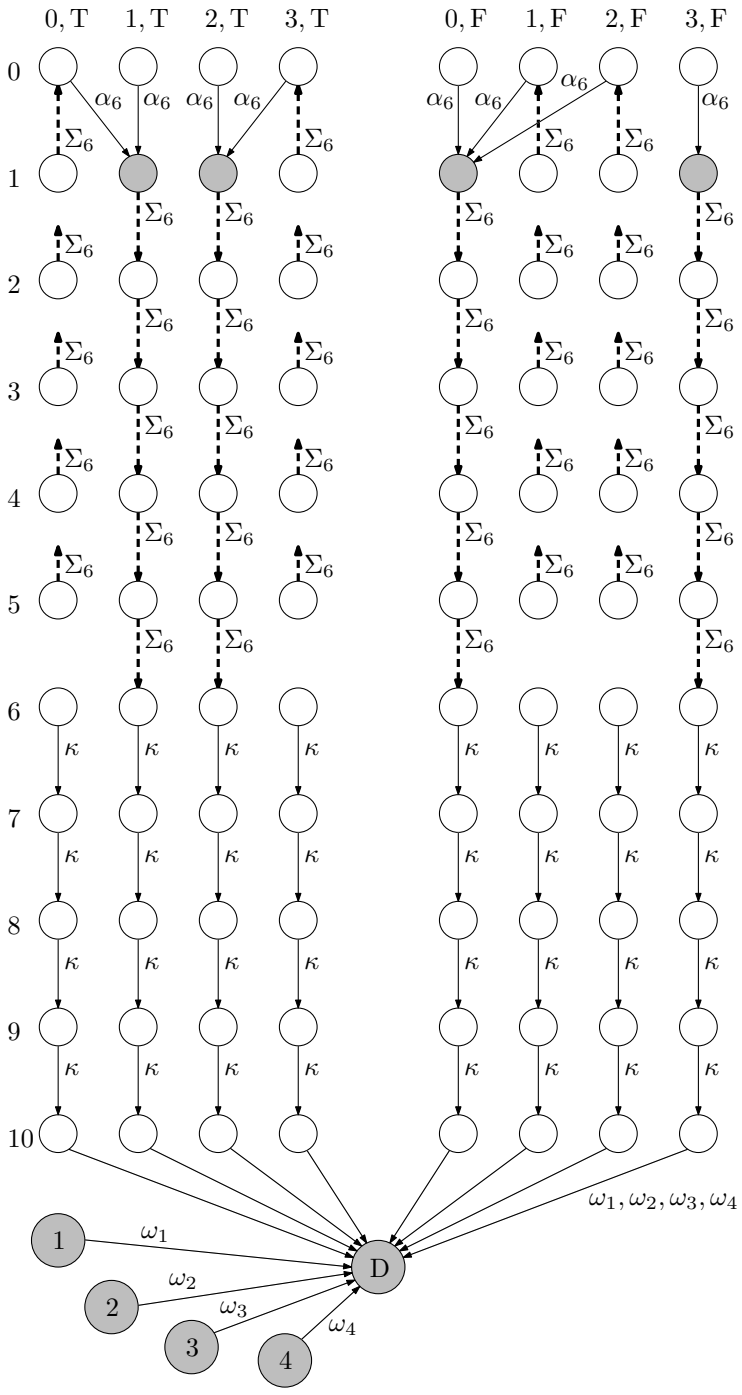


Figure 3.1: Some transitions of the example automaton described in Section 3.1.4. Grey states remain active after applying α_6.

because $11 = 2^0 + 2^1 + 2^3$. For each $i = 1, \ldots, m$ we define specific *pattern functions*

$$\pi_i^{\mathrm{T}}, \pi_i^{\mathrm{F}} : R \to R$$

such that

$$\begin{aligned} \operatorname{rng} \pi_i^{\mathrm{T}} &= \mathfrak{b}(i)\,, \\ \operatorname{rng} \pi_i^{\mathrm{F}} &= R \backslash \mathfrak{b}(i) \end{aligned}$$

for each i. It is irrelevant how exactly are π_i^{T} and π_i^{F} defined. It is sure that they exist, because the range is never expected to be empty. The action of the letters $\alpha_1, \ldots, \alpha_m$ is

$$\begin{aligned} (h, k, \mathrm{T}) &\xrightarrow{\alpha_i} \left(1, \pi_i^{\mathrm{T}}(k)\,, \mathrm{T}\right), \\ (h, k, \mathrm{F}) &\xrightarrow{\alpha_i} \left(1, \pi_i^{\mathrm{F}}(k)\,, \mathrm{F}\right), \end{aligned}$$

for each $i = 1, \ldots, m$, $h = 0, \ldots, z(n)$, and $k = 0, \ldots, q(m)$.

Note that each α_i maps the entire guard table, and in particular the entire row 0, into the row 1. In fact, all „downward" transitions within the guard table will lead only one row down, and the only transitions escaping from the guard table will lead from the bottom row. Thus any reset word will have length at least $d' = z(n) + 1$. Moreover, during its application, at time l the rows $0, \ldots, l-1$ will have to be all inactive. This is a key mechanism that the guard table uses for enforcing necessary properties of short reset words.

Let us define how the letters $x_{i,j}$ act on the guard table. Choose any $i \in \{1, \ldots, m\}$. The action of $x_{i,j}$ within the guard table does not depend on j, all the letters coming from a single automaton act identically here:

- for the rows $h \in \{1, \ldots, d_i\}$ we set

$$\begin{aligned} (h, k, \mathrm{T}) &\xrightarrow{x_{i,j}} \begin{cases} (h+1, k, \mathrm{T}) & \text{if } k \in \mathfrak{b}(i)\,, \\ (0, k, \mathrm{T}) & \text{otherwise,} \end{cases} \\ (h, k, \mathrm{F}) &\xrightarrow{x_{i,j}} \begin{cases} (h+1, k, \mathrm{F}) & \text{if } k \notin \mathfrak{b}(i)\,, \\ (0, k, \mathrm{F}) & \text{otherwise,} \end{cases} \end{aligned}$$

- for the remaining rows $h \in \{0\} \cup \{d_i + 1, \ldots, z(n)\}$ we set

$$\begin{aligned} (h, k, \mathrm{T}) &\xrightarrow{x_{i,j}} (0, k, \mathrm{T})\,, \\ (h, k, \mathrm{F}) &\xrightarrow{x_{i,j}} (0, k, \mathrm{F})\,. \end{aligned}$$

Recall that sending an activity marker along any transition ending in the row 0 is a „suicide". A word that does this cannot be a short reset word. So, if we restrict ourselves to letters from some Σ_i, the transitions defined above imply that only at times $1, \ldots, d_i$ the forthcoming letter can be some $x_{i,j}$. In the following $z(n) - d_i - 1$ steps the only letter from Σ_i that can be applied is κ.

The letter κ maps all the states of the guard table simply one state down, except for the rows 0 and $z(n)$. Set

$$\begin{aligned} (h, k, \mathrm{T}) &\xrightarrow{\kappa} (h+1, k, \mathrm{T})\,, \\ (h, k, \mathrm{F}) &\xrightarrow{\kappa} (h+1, k, \mathrm{F}) \end{aligned}$$

for each $h = 1, \ldots, z(n) - 1$, and

$$\begin{aligned}(0, k, \mathrm{T}) &\xrightarrow{\kappa} (0, k, \mathrm{T}), \\ (0, k, \mathrm{F}) &\xrightarrow{\kappa} (0, k, \mathrm{F}), \\ (z(n), k, \mathrm{T}) &\xrightarrow{\kappa} (0, k, \mathrm{T}), \\ (z(n), k, \mathrm{F}) &\xrightarrow{\kappa} (0, k, \mathrm{F}).\end{aligned}$$

It remains to describe actions of the letters $\omega_1, \ldots, \omega_n$ on the guard table. Set

$$\begin{aligned}(z(n), k, \mathrm{T}) &\xrightarrow{\omega_s} \mathrm{D}, \\ (z(n), k, \mathrm{F}) &\xrightarrow{\omega_s} \mathrm{D}\end{aligned}$$

for each $k = 0, \ldots, q(m)$, $s = 1, \ldots, n$, and

$$\begin{aligned}(h, k, \mathrm{T}) &\xrightarrow{\omega_s} (0, k, \mathrm{T}), \\ (h, k, \mathrm{F}) &\xrightarrow{\omega_s} (0, k, \mathrm{F})\end{aligned}$$

for each $h = 0, \ldots, z(n) - 1$, $k = 0, \ldots, q(m)$, and $s = 1, \ldots, n$. Now the automaton A' is complete.

3.1.4 An Example

Consider an input consisting of $m = 12$ automata $A_1, \ldots, A_{12}$, each of them having $n = 4$ states. Because $z(4) = 10$ and $q(12) = 3$, the output automaton A' has 93 states in total. In Figure 3.1 all the states are depicted, together with some of the transitions. We focus on the transitions corresponding to the automaton A_6, assuming that $d_6 = 5$.
The action of α_6 is determined by the fact that $6 = 2^1 + 2^2$ and thus

$$\begin{aligned}\operatorname{rng} \pi_6^{\mathrm{T}} &= \mathfrak{b}(6) = \{1, 2\}, \\ \operatorname{rng} \pi_6^{\mathrm{F}} &= R \backslash \mathfrak{b}(6) = \{0, 3\}.\end{aligned}$$

If the first letter of a reset word is α_6, after its application only the states

$$(1, 1, \mathrm{T}), (1, 2, \mathrm{T}), (1, 0, \mathrm{F}), (1, 3, \mathrm{F})$$

remain active within the guard table. Now we need to move their activity markers one row down in each of the following $z(n) - 1 = 9$ steps. The only way to do this is to apply $d_6 = 5$ letters of Σ_6 and then $z(n) - 1 - d_6 = 4$ occurrences of κ. Then we are allowed to apply one of the letters $\omega_1, \ldots, \omega_n$. But before that time, there should remain only one active state $s \in \{1, \ldots, n\}$, so that we could use ω_s. The letter κ does not affect the activity within $\{1, \ldots, n\}$ so we need to synchronize these states using $d_6 = 5$ letters from Σ_6.
So, any short reset word of A' starting with α_6 has to contain a short reset word of A_6.

3.1.5 The Guard Table Works

It remains to use ideas informally outlined in Section 3.1.3 to prove that A' has the properties C1, C2, and C3 from Section 3.1.3.

C1. As it has been said, for each letter $x \in \Sigma'$ and each state (h, k, Q), where $\mathrm{Q} \in \{\mathrm{T}, \mathrm{F}\}$ and $h \in \{0, \ldots, z(n) - 1\}$, it holds that

$$(h, k, \mathrm{Q}) \xrightarrow{x} (h', k', \mathrm{Q}),$$

where $h' < h$ or $h' = h + 1$. So the shortest paths from the row 0 to the state D have length at least $z(n) + 1$. □

C2. We should prove that any reset word w, having length exactly $z(n) + 1$, is of the form

$$w = \alpha_i y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i} \omega_s,$$

such that, moreover, $y_1 \dots y_{d_i}$ is a reset word of A_i. The starting α_i is necessary, because $\alpha_1, \dots, \alpha_m$ are the only letters that map states from the row 0 to other rows. Denote the remaining $z(n)$ letters of w by $y_1, \dots, y_{z(n)}$.
Once an α_i is applied, there remain only $|R| = q(m) + 1$ active states in the guard table, all in the row 1, depending on i. The active states are exactly from

$$\{1\} \times \mathfrak{b}(i) \times \{\mathrm{T}\} \text{ and } \{1\} \times R \backslash \mathfrak{b}(i) \times \{\mathrm{F}\},$$

because this is exactly the range of α_i within the guard table. Let us continue by an induction. We claim that for $0 \leq \tau < d_i$ it holds what we have already proved for $\tau = 0$:

1. If $\tau \geq 1$, the letter y_τ lies in Σ_i. Moreover, if $\tau > d_i$, it holds that $w_\tau = \kappa$.

2. After the application of y_τ the active states within the guard table are exactly from

$$\{\tau + 1\} \times \mathfrak{b}(i) \times \{\mathrm{T}\} \text{ and } \{\tau + 1\} \times R \backslash \mathfrak{b}(i) \times \{\mathrm{F}\}.$$

For $\tau = 0$ both the claims hold. Take some $1 \leq \tau < d_i$ and suppose that the claims hold for $\tau - 1$. Let us use the second claim for $\tau - 1$ to prove the first claim for τ. So all the states from

$$\{\tau\} \times \mathfrak{b}(i) \times \{\mathrm{T}\} \text{ and } \{\tau\} \times R \backslash \mathfrak{b}(i) \times \{\mathrm{F}\}$$

are active. Which of the letters could appear as y_τ? The letters $\omega_1, \dots, \omega_n$ and $\alpha_1, \dots, \alpha_m$ would map all the active states to the rows 0 and 1, which is a contradiction. Consider any letter $x_{k,j}$ for $k \neq i$. It holds that $\mathfrak{b}(i) \neq \mathfrak{b}(k)$, so there is some $c \in R$ lying in their symmetrical difference. For such c it holds that

$$(\tau, c, \mathrm{T}) \xrightarrow{x_{k,j}} (0, c, \mathrm{T}) \text{ if } c \in \mathfrak{b}(i) \setminus \mathfrak{b}(k)$$

or

$$(\tau, c, \mathrm{F}) \xrightarrow{x_{k,j}} (0, c, \mathrm{F}) \text{ if } c \in \mathfrak{b}(k) \setminus \mathfrak{b}(i)$$

which necessarily activates some state in the row 0, which is a contradiction again. So, $y_\tau \in \Sigma_i$. Moreover, if $\tau > d_i$, the letters from $\Sigma_i \setminus \{\kappa\}$ map the entire row τ into the row 0, so the only possibility is $y_\tau = \kappa$.
The letter y_τ maps all the active states right down to the row $\tau + 1$, so the second claim for τ holds as well.

□

C3. It is easy to verify that no „suicidal" transitions within the guard table are used, so during the application of

$$y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i}$$

the activity markers just flow down from the row 1 to the row $z(n)$. Since $y_1 \dots y_{d_i}$ is a reset word of A_i, there also remains only one particular state s within $\{1, \dots, n\}$. Finally the letter ω_s is applied which maps s and the entire row $z(n)$ directly to D. □

3.2 Complexity of SYN Restricted to Eulerian Binary Automata

An automaton $A = (Q, \Sigma, \delta)$ is *Eulerian* if

$$\sum_{x \in \Sigma} |\{r \in Q \mid \delta(r, x) = q\}| = |\Sigma|$$

for each $q \in Q$. Informally, there should be exactly $|\Sigma|$ transitions incoming to each state. An automaton is *binary* if $|\Sigma| = 2$. The class of Eulerian automata is denoted by $\mathcal{EU}$.
Previous results about various restrictions of Syn can be found in [33, 58, 60]. Some of these problems turned out to be polynomially solvable, others are NP-complete. In [60] Martyugin conjectured that Syn($\mathcal{EU} \cap \mathcal{AL}_2$) is NP-complete. This conjecture is confirmed in the rest of this section.

3.2.1 Proof Outline

We prove the NP-completeness of Syn($\mathcal{EU} \cap \mathcal{AL}_2$) by a polynomial reduction from 3-SAT. So, for arbitrary propositional formula ϕ in 3-CNF we construct an Eulerian binary automaton A and a number d such that

$$\phi \text{ is satisfiable} \Leftrightarrow A \text{ has a reset word of length } d. \tag{3.2}$$

For the rest of the paper we fix a formula

$$\phi = \bigwedge_{i=1}^{m} \bigvee_{\lambda \in C_i} \lambda$$

on n variables where each C_i is a three-element set of literals, i.e. subset of

$$L_\phi = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}.$$

We index the literals $\lambda \in L_\Phi$ by the following mapping κ:

λ	x_1	x_2	$\ldots$	x_n	$\neg x_1$	$\neg x_2$	$\ldots$	$\neg x_n$
$\kappa(\lambda)$	0	1	$\ldots$	$n-1$	n	$n+1$	$\ldots$	$2n-1$

Let $A = (Q, \Sigma, \delta)$, $\Sigma = \{a, b\}$. Because the structure of the automaton A will be very heterogeneous, we use an unusual method of description. The basic principles of the method are:

- We describe the automaton A via a labeled directed multigraph G, representing the automaton in a standard way: edges of G are labeled by single letters a and b and carry the structure of the function δ. Paths in G are thus *labeled* by words from $\{a, b\}^\star$.
- There is a collection of labeled directed multigraphs called *templates*. The graph G is one of them. Another template is `SINGLE`, which consists of one vertex and no edges.
- Each template `T`$\neq$`SINGLE` is expressed in a fixed way as a disjoint union through a set PARTS_T of its proper subgraphs (the *parts* of `T`), extended by a set of additional edges (the *links* of `T`). Each $H \in \text{PARTS}_\text{T}$ is isomorphic to some template `U`. We say that H *is of type* `U`.

- Let q be a vertex of a template $\mathtt{T}$, lying in subgraph $H \in \mathrm{PARTS}_{\mathtt{T}}$ which is of type $\mathtt{U}$ via vertex mapping $\rho : H \to \mathtt{U}$. The *local address* $\mathrm{adr}_{\mathtt{T}}(q)$ is a finite string of identifiers separated by „|". It is defined inductively by

$$\mathrm{adr}_{\mathtt{T}}(q) = \begin{cases} H \mid \mathrm{adr}_{\mathtt{U}}(\rho(q)) & \text{if } \mathtt{U} \neq \mathtt{SINGLE} \\ H & \text{if } \mathtt{U} = \mathtt{SINGLE}. \end{cases}$$

  The string $\mathrm{adr}_G(q)$ is used as a regular vertex identifier.

Having a word $w \in \Sigma^\star$, we denote a t-th letter of w by w_t and define the set $S_t = \delta(Q, w_1 \dots w_t)$ of *active states at time* t. Whenever we depict a graph, a solid arrow stands for the label a and a dotted arrow stands for the label b.

3.2.2 Description of the Graph G

Let us define all the templates and informally comment on their purpose. Figure 3.2 defines the template `ABS`, which does not depend on the formula ϕ.

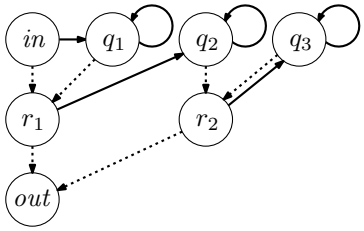


Figure 3.2: The template `ABS`

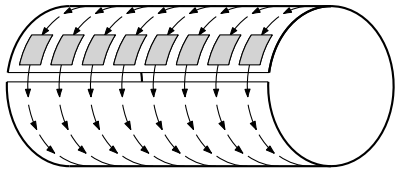


Figure 3.3: A barrier of `ABS` parts

The state *out* of a part of type `ABS` is always inactive after application of a word of length at least 2 which does not contain b^2 as a factor. This allows us to ensure the existence of a relatively short reset word. Actually, large areas of the graph (namely the `CLAUSE(...)` parts) have roughly the shape depicted in Figure 3.3, a cylindrical structure with a horizontal barrier of `ABS` parts. If we use a sufficiently long word with no occurrence of b^2, the edges outgoing from the `ABS` parts are never used and almost all states become inactive.

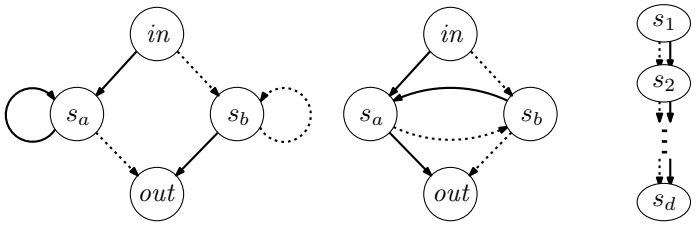


Figure 3.4: The templates `CCA`, `CCI` and `PIPE(`d`)` respectively

Figure 3.4 defines simple templates `CCA`, `CCI` and `PIPE(`d`)` for each $d \geq 1$. The activity of an *out* state depends on the last two letters applied. In the case of `CCA` it is inactive if (and typically only if) the two letters were equal. In the case of `CCI` it works oppositely, equal letters correspond to active *out* state. One of the key ideas of the entire construction is the following. Let there be a subgraph of the form

$$\begin{gathered}\text{part of type } \texttt{PIPE}(d)\\ \downarrow a, b\\ \text{part of type } \texttt{CCA} \text{ or } \texttt{CCI}\\ \downarrow a, b\\ \text{part of type } \texttt{PIPE}(d).\end{gathered} \tag{3.3}$$

Before the synchronization process starts, all the states are active. As soon as the second letter of an input word is applied, the activity of the *out* state starts to depend on the last two letters and the pipe below keeps a record of its previous activity. We say that a part H of type `PIPE(`d`)` *records* a sequence $B_1 \dots B_d \in \{\mathbf{0}, \mathbf{1}\}^d$ *at time* t, if it holds that

$$B_k = \mathbf{1} \Leftrightarrow H|s_k \notin S_t.$$

In order to continue with defining templates, let us define a set M_ϕ containing all the literals from L_ϕ and some auxiliary symbols:

$$M_\phi = L_\phi \cup \{y_1, \dots, y_n\} \cup \{z_1, \dots, z_n\} \cup \{q, q', r, r'\}.$$

We index the $4n + 4$ members $\nu \in M_\phi$ by the following mapping μ:

ν	q	r	y_1	x_1	y_2	x_2	$\dots$	y_n	x_n
$\mu(\nu)$	1	2	3	4	5	6	$\dots$	$2n{+}1$	$2n{+}2$

ν	q'	r'	z_1	$\neg x_1$	z_2	$\neg x_2$	$\dots$	z_n	$\neg x_n$
$\mu(\nu)$	$2n{+}3$	$2n{+}4$	$2n{+}5$	$2n{+}6$	$2n{+}7$	$2n{+}8$	$\dots$	$4n{+}3$	$4n{+}4$

The inverse mapping is denoted by μ'. For each $\lambda \in L_\phi$ we define templates `INC(`λ`)` and `NOTINC(`λ`)`, both consisting of $12n + 12$ `SINGLE` parts identified by elements of $\{1, 2, 3\} \times M_\phi$. As depicted by Figure 3.5a, the links of `INC(`λ`)`are:

$$(1,\nu) \xrightarrow{a} \begin{cases}(2,\lambda) & \text{if } \nu = \lambda \text{ or } \nu = r\\ (2,\nu) & \text{otherwise}\end{cases} \qquad (1,\nu) \xrightarrow{b} \begin{cases}(2,r) & \text{if } \nu = \lambda \text{ or } \nu = r\\ (2,\nu) & \text{otherwise}\end{cases}$$

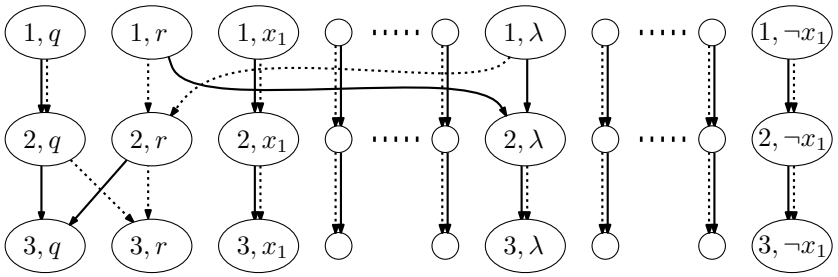
$$(2,\nu) \xrightarrow{a} \begin{cases}(3,q) & \text{if } \nu = r \text{ or } \nu = q\\ (3,\nu) & \text{otherwise}\end{cases} \qquad (2,\nu) \xrightarrow{b} \begin{cases}(3,r) & \text{if } \nu = r \text{ or } \nu = q\\ (3,\nu) & \text{otherwise}\end{cases}$$

Note that we use the same identifier for an one-vertex subgraph and for its vertex. As it is clear from Figure 3.5b, the links of `NOTINC(`λ`)` are:

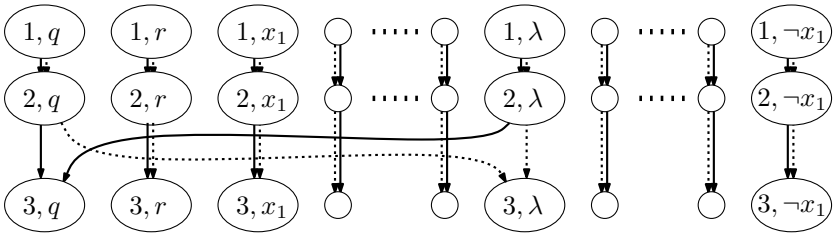
$$(1,\nu) \xrightarrow{a} (2,\lambda) \qquad (1,\nu) \xrightarrow{b} (2,r)$$

$$(2,\nu) \xrightarrow{a} \begin{cases}(3,q) & \text{if } \nu = q \text{ or } \nu = \lambda\\ (3,\nu) & \text{otherwise}\end{cases} \qquad (2,\nu) \xrightarrow{b} \begin{cases}(3,\lambda) & \text{if } \nu = q \text{ or } \nu = \lambda\\ (3,\nu) & \text{otherwise}\end{cases}$$

The key property of such templates comes to light when we need to apply some two-letter word in order to make the state $(3, \lambda)$ inactive assuming $(1, r)$ inactive. If also $(1, \lambda)$ is initially inactive, we can use the word a^2 in both templates. If it is active (which corresponds to the idea of unsatisfied literal λ), we discover the difference between the two templates: The word



(a) INC(λ)



(b) NOTINC(λ)

Figure 3.5: The templates INC(λ) and NOTINC(λ)

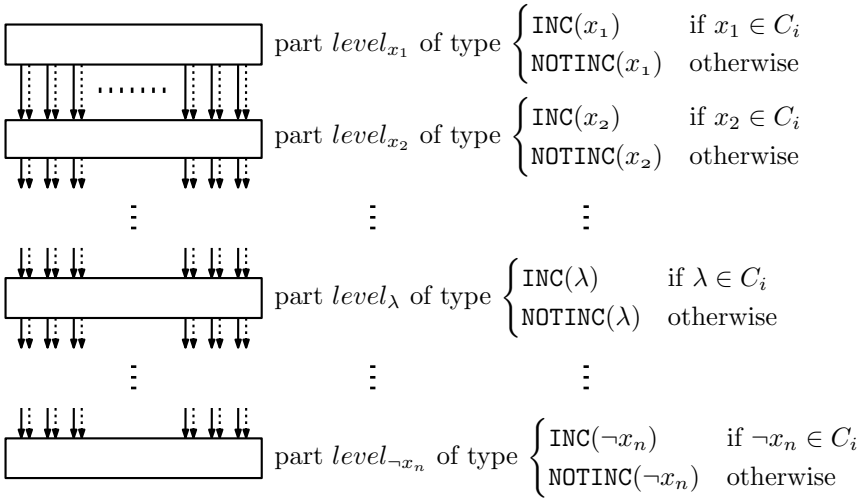


Figure 3.6: The template TESTER

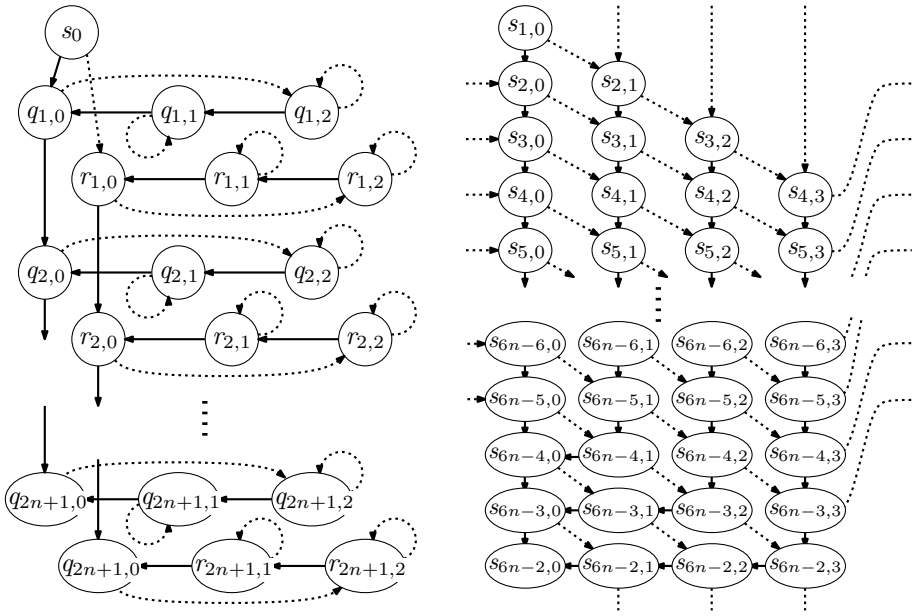


Figure 3.7: The templates `FORCER` and `LIMITER` respectively

a^2 works if the type is `NOTINC(`λ`)`, but fails in the case of `INC(`λ`)`. Such *failure* corresponds to the idea of unsatisfied literal λ occurring in a clause of ϕ.
For each clause (each $i \in \{1, \ldots, m\}$) we define a template `TESTER(`i`)`. It consists of $2n$ serially linked parts, namely $level_\lambda$ for each $\lambda \in L_\phi$, each of type `INC(`λ`)` or `NOTINC(`λ`)`. The particular type of each $level_\lambda$ depends on the clause C_i as seen in Figure 3.6, so exactly three of them are always of type `INC(...)`. If the corresponding clause is unsatisfied, each of its three literals is unsatisfied, which causes three failures within the levels. Three failures imply at least three occurrences of b, which turns up to be too much for a reset word of certain length to exist. Clearly we still need some additional mechanisms to realize this vague vision.
Figure 3.7 defines templates `FORCER` and `LIMITER`. The idea of template `FORCER` is simple. Imagine a situation when $q_{1,0}$ or $r_{1,0}$ is active and we need to deactivate the entire forcer by a word of length at most $2n+3$. Any use of b would cause an unbearable delay, so if such a word exists, it starts by a^{2n+2}.

The idea of `LIMITER` is similar, but we tolerate some occurrences of b here, namely two of them. This works if we assume $s_{1,0}$ active and it is necessary to deactivate the entire limiter by a word of length at most $6n + 1$.

We also need a template `PIPES(`d, k`)` for each $d, k \geq 1$. It consists just of k parallel pipes of length d. Namely there is a `SINGLE` part $s_{d',k'}$ for each $d' \leq d$, $k' \leq k$ and all the edges are of the form $s_{d',k'} \longrightarrow s_{d'+1,k'}$.

The most complex templates are `CLAUSE(`i`)` for each $i \in \{1, \ldots, m\}$. Denote

$$\begin{aligned} \alpha_i &= (i-1)\,(12n-2)\,, \\ \beta_i &= (m-i)\,(12n-2)\,. \end{aligned}$$

As shown in Figure 3.8, `CLAUSE(`i`)` consists of the following parts:

- Parts $sp_1, \ldots, sp_{4n+6}$ of type `SINGLE`.
- Parts $abs_1, \ldots, abs_{4n+6}$ of type `ABS`. The entire template has a shape similar to Figure 3.3, including the barrier of `ABS` parts.

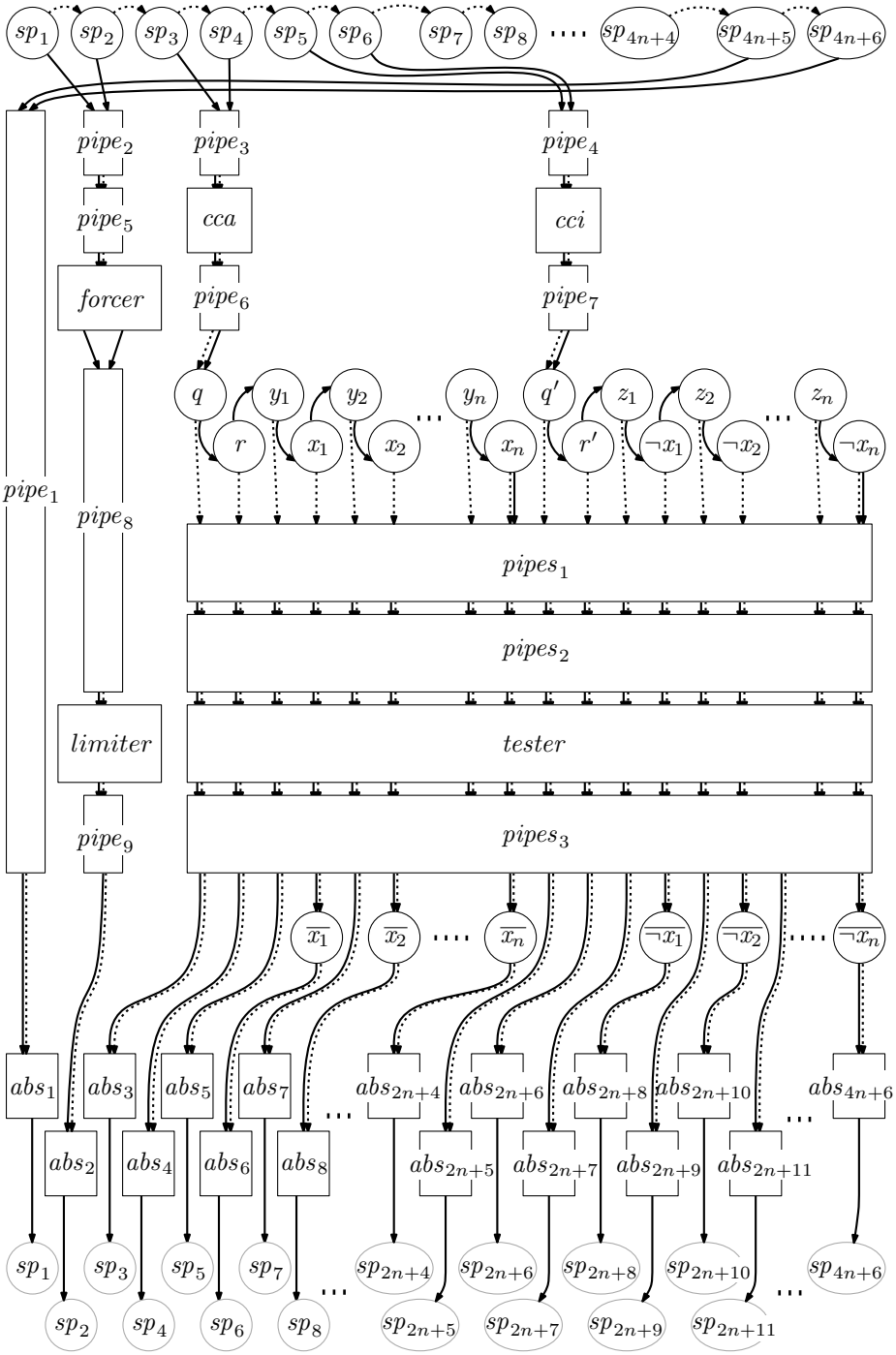


Figure 3.8: The template `CLAUSE(`i`)`

- Parts $pipe_2$, $pipe_3$, $pipe_4$ of types `PIPE(`$2n-1$`)` and $pipe_6$, $pipe_7$ of types `PIPE(`$2n+2$`)`.
- Parts *cca* and *cci* of types `CCA` and `CCI` respectively. Together with the pipes above they realize the idea described in (3.3). As they form two constellations which work simultaneously, the parts $pipe_6$ and $pipe_7$ typically record mutually inverse sequences. We interpret them as an assignment of the variables $x_1, \ldots, x_n$. Such assignment is then processed by the tester.
- A part ν of type `SINGLE` for each $\nu \in M_\phi$.
- A part *tester* of type `TESTER(`i`)`.
- A part $\overline{\lambda}$ of type `SINGLE` for each $\lambda \in L_\phi$. While describing the templates `INC(`λ`)` and `NOTINC(`λ`)` we claimed that in certain case there arises a need to make the state $(3, \lambda)$ inactive. This happens when the border of inactive area moves down through the tester levels. The point is that any word of length $6n$ deactivates the entire tester, but we need to ensure that some tester columns, namely the $\kappa(\lambda)$-th for each $\lambda \in L_\phi$, are deactivated one step earlier. If some of them is still active just before the deactivation of tester finishes, the state $\overline{\lambda}$ becomes active, which slows down the synchronization process.
- Parts $pipes_1$, $pipes_2$ and $pipes_3$ of types `PIPES(`$\alpha_i, 4n+4$`)`, `PIPES(`$6n-2, 4n+4$`)` and `PIPES(`$\beta_i, 4n+4$`)` respectively. There are multiple clauses in ϕ, but multiple testers cannot work in parallel. That is why each of them is padded by a passive `PIPES(...)` part of size depending on particular i. If $\alpha_i = 0$ or $\beta_i = 0$, the corresponding `PIPES` part is not present in cl_i.
- Parts $pipe_1$, $pipe_5$, $pipe_8$, $pipe_9$ of types `PIPE(`$12mn+4n-2m+6$`)`, `PIPE(4)`, `PIPE(`$\alpha_i + 6n - 1$`)`, `PIPE(`β_i`)` respectively.
- The part *forcer* of type `FORCER`. This part guarantees that only the letter a is used in certain segment of the word w. This is necessary for the data produced by *cca* and *cci* to safely leave the parts $pipe_3$, $pipe_4$ and line up in the states of the form ν for $\nu \in M_\phi$, from where they are shifted to the tester.
- The part *limiter* of type `LIMITER`. This part guarantees that the letter b occurs at most twice when the border of inactive area passes through the tester. Because each unsatisfied literal from the clause requests an occurrence of b, only a satisfied clause meets all the conditions for a reset word of certain length to exist.

Links of `CLAUSE`(i), which are not clear from Figure 3.8 are

$$\nu \xrightarrow{a} \begin{cases} pipes_1|s_{1,\mu(\nu)} & \text{if } \nu = \neg x_n \\ \mu'(\mu(\nu)+1) & \text{otherwise} \end{cases} \qquad \nu \xrightarrow{b} pipes_1|s_{1,\mu(\nu)}$$

for each $\nu \in M_\phi$ and

$$pipes_3|s_{\beta_i,k} \xrightarrow{a,b} \begin{cases} \overline{\mu'(k)} & \text{if } \mu'(k) \in L_\phi \\ abs_{k+2}|in & \text{otherwise} \end{cases} \qquad \overline{\lambda} \xrightarrow{a,b} abs_{\mu(\lambda)+2}|in$$

for each $k \in \{1, \ldots, 4n+4\}$, $\lambda \in L_\phi$.
We are ready to form the whole graph G, see Figure 3.9. For each $i, k \in \{1, \ldots m\}$ there are parts cl_k, abs_k of types `CLAUSE(`i`)` and `ABS` respectively and parts q_k, r_k, r'_k, s_1, s_2 of type `SINGLE`. The edge incoming to a cl_i part ends in $cl_i|sp_1$, the outgoing one starts in $cl_i|sp_{4n+6}$. When no states outside `ABS` parts are active within each `CLAUSE(...)` part and no *out*, r_1 nor r_2 state is active in any `ABS` part, the word b^2ab^{4n+m+7} takes all active states to s_2 and completes the synchronization. Graph G does not fully represent the automaton A yet because there are

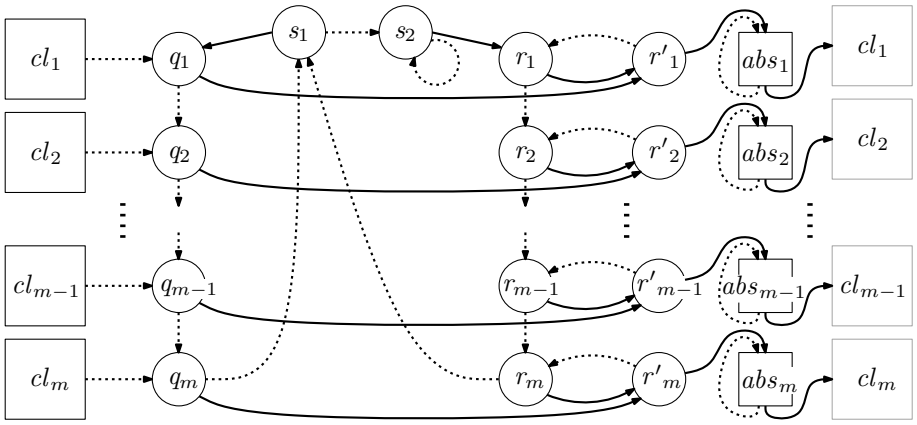


Figure 3.9: The graph G

- $8mn + 4m$ vertices with only one outgoing edge, namely $cl_i|abs_k|out$ and $cl_i|sp_l$ for each $i \in \{1, \ldots, m\}, k \in \{1, \ldots, 4n + 6\}, l \in \{7, \ldots, 4n + 4\}$,

- $8mn + 4m$ vertices with only one incoming edge: $cl_i|\nu$ and $cl_i|pipes_1|(1, \nu')$ for each $i \in \{1, \ldots, m\}, \nu \in M_\phi \setminus \{q, q'\}, \nu' \in M_\phi \setminus \{x_n, \neg x_n\}$.

But we do not need to specify the missing edges exactly, let us just say that they somehow connect the relevant states and the automaton A is complete. Let us set

$$d = 12mn + 8n - m + 18$$

and prove that the equivalence (3.2) holds.

3.2.3 From an Assignment to a Word

First let us suppose that there is an assignment $\xi_1, \ldots, \xi_n \in \{\mathbf{0}, \mathbf{1}\}$ of the variables $x_1, \ldots, x_n$ (respectively) satisfying the formula ϕ and prove that the automaton A has a reset word w of length d. For each $j \in \{1, \ldots, n\}$ we denote

$$\sigma_j = \begin{cases} a & \text{if } \xi_j = \mathbf{1} \\ b & \text{if } \xi_j = \mathbf{0} \end{cases}$$

and for each $i \in \{1, \ldots, m\}$ we choose a satisfied literal $\overline{\lambda}_i$ from C_i. We set

$$w = a^2 (\sigma_n a) (\sigma_{n-1} a) \ldots (\sigma_1 a) aba^{2n+3} b \left(a^{6n-2} v_1\right) \ldots \left(a^{6n-2} v_m\right) b^2 a b^{4n+m+7},$$

where for each $i \in \{1, \ldots, m\}$ we use the word

$$v_i = u_{i,x_1} \ldots u_{i,x_n} u_{i,\neg x_1} \ldots u_{i,\neg x_n},$$

denoting

$$u_{i,\lambda} = \begin{cases} a^3 & \text{if } \lambda = \overline{\lambda}_i \text{ or } \lambda \notin C_i \\ ba^2 & \text{if } \lambda \neq \overline{\lambda}_i \text{ and } \lambda \in C_i \end{cases}$$

for each $\lambda \in L_\phi$. We see that $|v_i| = 6n$ and therefore

$$|w| = 4n + 8 + m(12n - 2) + 4n + m + 10 = 12mn + 8n - m + 18 = d.$$

Let us denote

$$\gamma = 12mn + 4n - 2m + 9$$

and

$$\overline{S}_t = Q \backslash S_t$$

for each $t \leq d$. Because the first occurrence of b^2 in w starts by the γ-th letter, we have:

Lemma 3.4. *Each state of a form $cl_{\dots}|abs_{\dots}|out$ or $abs_{\dots}|out$ lies in $\overline{S}_2 \cap \dots \cap \overline{S}_\gamma$.*

Let us fix an arbitrary $i \in \{1, \dots, m\}$ and describe a growing area of inactive states within cl_i. We use the following method of verifying inactivity of states: Having a state $s \in Q$ and $t, k \geq 1$ such that any path of length k ending in s uses a member of $\overline{S}_{t-k} \cap \dots \cap \overline{S}_{t-1}$, we easily deduce that $s \in \overline{S}_t$. In such case let us just say that k *witnesses* that $s \in \overline{S}_t$. The following claims follow directly from the definition of w. Note that Claim 7 relies on the fact that b occurs only twice in v_i.

Lemma 3.5.

1. $\{cl_i|sp_1, \dots, cl_i|sp_{4n+6}\} \subseteq \overline{S}_2 \cap \dots \cap \overline{S}_\gamma$
2. $cl_i|pipe_2 \cup cl_i|pipe_3 \cup cl_i|pipe_4 \subseteq \overline{S}_{2n+1} \cap \dots \cap \overline{S}_\gamma$
3. $cl_i|cca \cup cl_i|cci \cup cl_i|pipe_5 \subseteq \overline{S}_{2n+5} \cap \dots \cap \overline{S}_\gamma$
4. $cl_i|pipe_6 \cup cl_i|pipe_7 \cup cl_i|forcer \subseteq \overline{S}_{4n+7} \cap \dots \cap \overline{S}_\gamma$
5. $\{cl_i|\nu : \nu \in M_\phi\} \subseteq \overline{S}_{4n+8} \cap \dots \cap \overline{S}_\gamma$
6. $cl_i|pipes_1 \cup cl_i|pipes_2 \cup cl_i|pipe_8 \subseteq \overline{S}_{10n+\alpha_i+6} \cap \dots \cap \overline{S}_\gamma$
7. $cl_i|limiter \cup cl_i|tester \subseteq \overline{S}_{16n+\alpha_i+6} \cap \dots \cap \overline{S}_\gamma$
8. $cl_i|pipe_1 \cup cl_i|pipe_9 \cup cl_i|pipes_3 \subseteq \overline{S}_{\gamma-1} \cap \overline{S}_\gamma$

Proof.

1. *Claim:* $\{cl_i|sp_1, \dots, cl_i|sp_{4n+6}\} \subseteq \overline{S}_2 \cap \dots \cap \overline{S}_\gamma$.
   We have $w_1 w_2 = a^2$ and there is no path labeled by a^2 ending in any $cl_i|sp_{\dots}$ state, so such states lie in $\overline{S}_2$. For each $t = 3, \dots, \gamma$ we can inductively use $k = 1$ to witness the memberships in $\overline{S}_t$. In the induction step we use Lemma 3.4, which excludes the *out* states of the `ABS` parts from each corresponding $\overline{S}_{t-1}$.

2. *Claim:* $cl_i|pipe_2 \cup cl_i|pipe_3 \cup cl_i|pipe_4 \subseteq \overline{S}_{2n+1} \cap \dots \cap \overline{S}_\gamma$.
   All the memberships are witnessed by $k = 2n - 1$, because any path of the length $2n - 1$ ending in such state must use a $cl_i|sp_{\dots}$ state and such states lie in $\overline{S}_2 \cap \dots \cap \overline{S}_\gamma$ by the previous claim.

3. *Claim:* $cl_i|cca \cup cl_i|cci \cup cl_i|pipe_5 \subseteq \overline{S}_{2n+5} \cap \dots \cap \overline{S}_\gamma$.
   We have $w_{2n+2} \dots w_{2n+5} = a^2ba$, which clearly maps each state of $cl_i|cca$, $cl_i|cci$ or $cl_i|pipe_5$ out of those parts. Each path of length 4 leading into the parts from outside starts in $\overline{S}_{2n+1}$, so it follows that all the states lie in $\overline{S}_{2n+5}$. To prove the rest we inductively use the witness $k = 1$.

4. *Claim:* $cl_i|pipe_6 \cup cl_i|pipe_7 \cup cl_i|forcer \subseteq \overline{S}_{4n+7} \cap \dots \cap \overline{S}_\gamma$.
   In the cases of $cl_i|pipe_6$ *and* $cl_i|pipe_7$ we just use the witness $k = 2n + 2$. In the case of

$cl_i|forcer$ we proceed the same way as in the previous claim. We have $w_{2n+6} \dots w_{4n+7} = a^{2n+2}$. Because also $w_{2n+5} = a$, only the states $q_{\dots,0}$ can be active within the part $cl_i|forcer$ in time $2n+6$. The word $w_{2n+7} \dots w_{4n+7}$ maps all such states out of $cl_i|forcer$. Each path of length $2n+2$ leading into $cl_i|forcer$ from outside starts in $\overline{S}_{2n+5}$, so it follows that all states from $cl_i|forcer$ lie in $\overline{S}_{4n+7}$. To handle $t = 4n+8, \dots, \gamma$ we inductively use the witness $k = 1$.

5. *Claim:* $\{cl_i|\nu : \nu \in M_\phi\} \subseteq \overline{S}_{4n+8} \cap \dots \cap \overline{S}_\gamma$.
In the cases of $cl_i|q$ and $cl_i|q'$ we use the witness 1. We have $w_{4n+8} = b$ and the only edges labeled by b incoming to remaining states could be some of the $8mn + 4m$ unspecified edges of G. But we have $w_{4n+6}w_{4n+7} = a^2$, so each *out* state of any `ABS` part lies in $\overline{S}_{4n+7}$ and thus no unspecified edge starts in a state outside $\overline{S}_{4n+7}$.

6. *Claim:* $cl_i|pipes_1 \cup cl_i|pipes_2 \cup cl_i|pipe_8 \subseteq \overline{S}_{10n+\alpha_i+6} \cap \dots \cap \overline{S}_\gamma$.
We use witnesses $k = \alpha_i$ for $cl_i|pipes_1$, $k = 6n-2$ for $cl_i|pipes_2$ and $k = \alpha_i + 6n - 1$ for $cl_i|pipe_8$.

7. *Claim:* $cl_i|limiter \cup cl_i|tester \subseteq \overline{S}_{16n+\alpha_i+6} \cap \dots \cap \overline{S}_\gamma$.
Because
$$w_{4n+\alpha_i+9} \dots w_{10n+\alpha_i+6} = a^{6n-2},$$
there are only states of the form $cl_i|limiter|s_{\dots,0}$ in the intersection of $cl_i|limiter$ and $S_{10n+\alpha_i+6}$. Together with the fact that there are only two occurrences of b in v_i it confirms that the case of $cl_i|limiter$ holds. The case of $cl_i|tester$ is easily witnessed by $k = 6n$.

8. *Claim:* $cl_i|pipe_1 \cup cl_i|pipe_9 \cup cl_i|pipes_3 \subseteq \overline{S}_{\gamma-1} \cap \overline{S}_\gamma$.
We use witnesses $k = 12mn + 4n - 2m + 6$ for $cl_i|pipe_1$ and $k = \beta_i$ for $cl_i|pipe_9, cl_i|pipes_3$.

□

For each $\lambda \in L_\phi$ we ensure by the word $u_{i,\lambda}$ that the $\kappa(\lambda)$-th tester column is deactivated in advance, namely at time $t = 16n + \alpha_i + 5$. The advance allows the following key claim to hold true.

Lemma 3.6. $\left\{cl_i|\overline{\lambda} : \lambda \in L_\phi\right\} \subseteq \overline{S}_{\gamma-1} \cap \overline{S}_\gamma$.

Proof. For each such λ we choose

$$k = 6n - 3\kappa(\lambda) + \beta_i + 1$$

as a witness of $cl_i|\overline{\lambda} \in \overline{S}_{\gamma-1}$. There is only one state where a path of length k ending in $\overline{\lambda}$ starts: the state

$$s = cl_i|tester|level_\lambda|\,(3, \lambda)\,.$$

It holds that

$$s \in \overline{S}_{10n+\alpha_i+3\kappa(\lambda)+6} \cap \dots \cap \overline{S}_\gamma,$$

as is easily witnessed by $k' = 3\kappa(\lambda)$ using Claim 6 of Lemma 3.5. But we are going to show also that

$$s \in \overline{S}_{10n+\alpha_i+3\kappa(\lambda)+5}, \tag{3.4}$$

which will imply that k is a true witness of $\overline{\lambda} \in \overline{S}_{\gamma-1}$, because

$$(\gamma - 1) - k = 10n + \alpha_i + 3\kappa(\lambda) + 5.$$

So let us prove the membership (3.4). We need to observe, using the definition of w, that:

- At time $2n+5$ the part $pipe_6$ records the sequence

$$\mathbf{0}, \mathbf{1}, \xi_1, \xi_1, \xi_2, \xi_2, \ldots, \xi_n, \xi_n$$

and the part $pipe_7$ records the sequence of inverted values. Because

$$w_{2n+6} \ldots w_{4n+7} = a^{2n+2},$$

at time $4n+7$ the states q, r' are active, the states q', r are inactive and for each $j \in \{1, \ldots, n\}$ it holds that

$$x_j \in \overline{S}_{4n+7} \Leftrightarrow y_j \in \overline{S}_{4n+7} \Leftrightarrow \neg x_j \in S_{4n+7} \Leftrightarrow z_j \in S_{4n+7} \Leftrightarrow \xi_j = \mathbf{1}.$$

Because $w_{4n+8} = b$, at time $10n + \alpha_i + 6$ we find the whole structure above shifted to the first row of $cl_i|tester$, so particularly for $\lambda \in L_\phi$:

$$cl_i|tester|level_{x_1}|\,(1, \lambda) \in \overline{S}_{10n+\alpha_i+6} \Leftrightarrow \lambda \text{ is satisfied by } \xi_1, \ldots, \xi_n.$$

- From a simple induction on tester levels it follows that

$$cl_i|tester|level_\lambda|\,(1, r) \in \overline{S}_{10n+\alpha_i+3\kappa(\lambda)+3}.$$

Note that

$$w_{10n+\alpha_i+3\kappa(\lambda)+4} w_{10n+\alpha_i+3\kappa(\lambda)+5} w_{10n+\alpha_i+3\kappa(\lambda)+6} = u_{i,\lambda}$$

and distinguish the following cases:

- If $\lambda = \overline{\lambda}_i$, we have $\lambda \in C_i$, the part $cl_i|tester|level_\lambda$ is of type `INC(λ)` and $u_{i,\lambda} = a^3$. We also know that λ is satisfied, so

$$cl_i|tester|level_{x_1}|\,(1, \lambda) \in \overline{S}_{10n+\alpha_i+6}.$$

The state above is the only state, from which any path of length $3\kappa(\lambda) - 3$ leads to $cl_i|tester|level_\lambda|\,(1, \lambda)$, so we deduce that

$$cl_i|tester|level_\lambda|\,(1, \lambda) \in \overline{S}_{10n+\alpha_i+3\kappa(\lambda)+3}.$$

We see that each path labeled by a^2 ending in $cl_i|tester|level_\lambda|\,(3, \lambda)$ starts in $cl_i|tester|level_\lambda|\,(1, \lambda)$ or in $cl_i|tester|level_\lambda|\,(1, r)$, but each of the two states lies in $\overline{S}_{10n+\alpha_i+3\kappa(\lambda)+3}$. So the membership (3.4) holds.

- If $\lambda \notin C$, the part $cl_i|tester|level_\lambda$ is of type `NOTINC(λ)` and $u_{i,\lambda} = a^3$. Particularly $w_{10n+\alpha_i+3\kappa(\lambda)+5} = a$ but no edge labeled by a comes to $cl_i|tester|level_\lambda|\,(3, \lambda)$ and the membership (3.4) follows trivially.

- If $\lambda \neq \overline{\lambda}_i$ and $\lambda \in C_i$, the part $cl_i|tester|level_\lambda$ is of type `INC(λ)` and $u_{i,\lambda} = ba^2$. Particularly

$$w_{10n+\alpha_i+3\kappa(\lambda)+4} w_{10n+\alpha_i+3\kappa(\lambda)+5} = ba,$$

but no path labeled by ba comes to $cl_i|tester|level_\lambda|\,(3, \lambda)$, so we reach the same conclusion as in the previous case.

We have proven that $cl_i|\overline{\lambda}$ lies in $\overline{S}_{\gamma-1}$. From Claim 8 of Lemma 3.5 it follows directly that it lies also in $\overline{S}_\gamma$. □

We see that within cl_i only states from the `ABS` parts can lie in $S_{\gamma-1}$. Since $w_{\gamma-2}w_{\gamma-1} = a^2$, no state r_1, r_2 or *out* from any `ABS` part lies in $S_{\gamma-1}$. Now we easily check that all the states possibly present in $S_{\gamma-1}$ are mapped to s_2 by the word $w_\gamma \dots w_d = b^2ab^{4n+m+7}$.

3.2.4 From a Word to an Assignment.

Since now we suppose that there is a reset word w of length

$$d = 12mn + 8n - m + 18.$$

The following lemma is not hard to verify.

Lemma 3.7.

1. *Up to labeling there is a unique pair of paths, both of a length $l \leq d-2$, leading from $cl_1|pipe_1|s_1$ and $cl_2|pipe_1|s_1$ to a common end. They are of length $d-2$ and meet in s_2.*
2. *The word w starts by a^2.*

Proof.

1. The leading segments of both paths are similar since they stay within the parts cl_1 and cl_2:

$$pipe_1|s_1 \xrightarrow{a,b} \dots \xrightarrow{a,b} pipe_1|s_{12mn+4n-2m+6} \xrightarrow{a,b} abs_1|in \xrightarrow{b} \\ \xrightarrow{b} abs_1|r_1 \xrightarrow{b} abs_1|out \xrightarrow{a} sp_1 \xrightarrow{b} \dots \xrightarrow{b} sp_{4n+6}.$$

   Once the paths leave the parts cl_1 and cl_2, the shortest way to merge is the following:

$$\begin{array}{l} cl_1|sp_{4n+6} \xrightarrow{b} q_1 \xrightarrow{b} q_2 \xrightarrow{b} \dots \xrightarrow{b} q_{m-1} \xrightarrow{b} q_m \xrightarrow{b} s_1 \xrightarrow{b} \\ cl_2|sp_{4n+6} \xrightarrow{b} q_2 \xrightarrow{b} q_3 \xrightarrow{b} \dots \xrightarrow{b} q_m \xrightarrow{b} s_1 \xrightarrow{b} s_2 \xrightarrow{b} \end{array} s_2$$

   Having the description above it is easy to verify that the length is $d-2$ and there is no way to make the paths shorter.

2. Suppose that $w_1w_2 \neq a^2$. Any of the three possible values of w_1w_2 implies that

$$\left\{ cl_i|sp_3, \dots, cl_i|sp_{4n+6} \right\} \subseteq S_2$$

   for each i. It cannot hold that $w = w_1w_2b^{d-2}$, because in such case all $cl_{\dots}|cca|s_b$ states would be active in any time $t \geq 3$. So the word w has a prefix $w_1w_2b^ka$ for some $k \geq 0$. If $k \leq 4n+3$, it holds that $cl_i|sp_{4n+6} \in S_{k+2}$ and therefore $cl_i|pipe_1|s_1 \in S_{k+3}$, which contradicts the first claim. Let $k \geq 4n+4$. Some state of a form $cl_i|forcer|q_{1,\dots}$ or $cl_i|forcer|r_{1,\dots}$ lies in S_{k+2} for each i. This holds particularly for $i = 1$ and $i = 2$, but there is no pair of paths of length at most

$$d - (4n+4) \geq d - k$$

   leading from such two states to a common end.

□

The second claim implies that $cl_i|pipe_1|s_1 \in S_2$ for each $i \in \{1, \dots, m\}$, so it follows that

$$\delta(Q, w) = \{s_2\}.$$

Let us denote

$$\overline{d} = 12mn + 4n - 2m + 11$$

and

$$\overline{w} = w_1 \dots w_{\overline{d}}.$$

The following lemma holds because no edges labeled by a are available for final segments of the paths described in the first claim of Lemma 3.7.

Lemma 3.8.

1. *The word w can be written as $w = \overline{w}b^{4n+m+7}$ for some word $\overline{w}$.*
2. *For any $t \geq \overline{d}$, no state from any $cl_{\dots}$ part lie in S_t, except for the $sp_{\dots}$ states.*

Proof.

1. Let us write $w = w_1 w_2 w'$. From Lemma 3.7 it follows that

   $$\delta\left(cl_1|pipe_1|s_1, w'\right) = \delta\left(cl_2|pipe_1|s_1, w'\right)$$

   and w' have to label some of the paths determined up to labeling in Lemma 3.7(1). The final $4n+m+7$ edges of the paths lead from $cl_1|sp_1$ and $cl_2|sp_1$ to s_2. All the transitions used here are necessarily labeled by b.

2. The claim is easy to observe, since the first claim implies that S_t is a subset of

   $$S' = \left\{s \in Q \mid (\exists d \in \mathbb{N})\, \delta\left(s, b^d\right) = s_2\right\}.$$

□

The next lemma is based on properties of the parts $cl_{\dots}|forcer$ but to prove that no more a follows the enforced factor a^{2n+1} we also need to observe that each $cl_{\dots}|cca|out$ or each $cl_{\dots}|cci|out$ lies in S_{2n+4}.

Lemma 3.9. *The word $\overline{w}$ starts by $\overline{u}a^{2n+1}b$ for some $\overline{u}$ of length $2n+6$.*

Proof. At first we prove that $\overline{w}$ starts by $\overline{u}a^{2n+1}$. Lemma 3.7(2) implies that $cl_1|pipe_2|s_1 \in S_2$, so obviously some of the states $cl_1|forcer|q_{1,0}$ and $cl_1|forcer|r_{1,0}$ lies in S_{2n+6}. If $w_{2n+6+k} = b$ for some $k \in \{1, \dots, 2n+1\}$, it holds that $cl_i|forcer|q_{k,2}$ or $cl_i|forcer|r_{k,2}$ lies in S_{2n+6+k}. From such state no path of length at most $2n+3-k$ leads to $cl_i|pipe_8|s_1$ and therefore no path of length at most

$$(2n+3-k) + (\alpha_i + 6n - 1) + (6n-2) + \beta_i + 3 = \overline{d} - (2n+6+k)$$

leads into S', which contradicts Lemma 3.8(2). It remains to show that there is b after the prefix $\overline{u}a^{2n+1}$. Lemma 3.7(2) implies that both $cl_1|cca|in$ and $cl_1|cci|in$ lie in S_{2n+1}, from which it is not hard to deduce that $cl_1|cca|out$ or $cl_1|cci|out$ lies in S_{2n+4} and therefore $cl_1|q$ or $cl_1|r$ lies in S_{4n+7}. Any path of length $\overline{d} - (4n+7)$ leading from $cl_1|q$ or $cl_1|r$ into $\overline{S}$ starts by an edge labeled by b. □

Now we are able to write the word $\overline{w}$ as

$$\overline{w} = \overline{u}a^{2n+1}b\left(\overline{v}_1 v'_1 c_1\right) \dots \left(\overline{v}_m v'_m c_m\right) w_{\overline{d}-2} w_{\overline{d}-1} w_{\overline{d}},$$

where $|\overline{v}_k| = 6n-2$, $|v'_k| = 6n-1$ and $|c_k| = 1$ for each k and denote $d_i = 10n + \alpha_i + 6$. At time $2n+5$ the parts $cl_{\dots}|pipe_6$ and $cl_{\dots}|pipe_7$ record mutually inverse sequences. Because there is the factor a^{2n+1} after $\overline{u}$, at time d_i we find the information pushed to the first rows of testers:

Lemma 3.10. *For each $i \in \{1,\dots,m\}$, $j \in \{1,\dots,n\}$ it holds that*

$$\begin{aligned} cl_i|tester|level_{x_1}|\,(1,x_j) \in S_{d_i} \quad &\Leftrightarrow \\ cl_i|tester|level_{x_1}|\,(1,\neg x_j) \notin S_{d_i} \quad &\Leftrightarrow \quad w_{2n-2j+2} \neq w_{2n-2j+3}. \end{aligned}$$

Proof. From the definition of `CCA` and `CCI` it follows that at time $2n+5$ the parts $pipe_6$ and $pipe_7$ record the sequences $B_{(2n+3)}\dots B_{(2)}$ and $B'_{(2n+3)}\dots B'_{(2)}$ respectively, where

$$B_{(k)} = \begin{cases} \mathbf{1} & \text{if } w_k = w_{k+1} \\ \mathbf{0} & \text{otherwise} \end{cases} \qquad B'_{(k)} = \begin{cases} \mathbf{0} & \text{if } w_k = w_{k+1} \\ \mathbf{1} & \text{otherwise.} \end{cases}$$

Whatever the letter w_{2n+6} is, Lemma 3.9 implies that

$$cl_i|x_j \in S_{4n+7} \Leftrightarrow cl_i|\neg x_j \notin S_{4n+7} \Leftrightarrow w_{2n-2j+2} \neq w_{2n-2j+3},$$

from which the claim follows easily using Lemma 3.9 again. □

Let us define the assignment $\xi_1,\dots,\xi_n \in \{\mathbf{0},\mathbf{1}\}$. By Lemma 3.10 the definition is correct and does not depend on i:

$$\xi_j = \begin{cases} \mathbf{1} & \text{if } cl_i|tester|level_{x_1}|\,(1,x_j) \notin S_{d_i} \\ \mathbf{0} & \text{if } cl_i|tester|level_{x_1}|\,(1,\neg x_j) \notin S_{d_i}. \end{cases}$$

The following lemma holds due to $cl_{\dots}|limiter$ parts.

Lemma 3.11. *For each $i \in \{1,\dots,m\}$ there are at most two occurrences of b in the word v'_i.*

Proof. It is easy to see that $cl_i|limiter|s_{1,0} \in S_{10n+\alpha_i+6}$ and to note that

$$v'_i = w_{10n+\alpha_i+7}\dots w_{16n+\alpha_i+5}.$$

Within the part $cl_i|limiter$ no state except for $s_{6n-2,0}$ can lie in $S_{16n+\alpha_i+5}$, because from such states there is no path of length at most

$$\overline{d} - (16n+\alpha_i+5) = \beta_i + 4$$

leading into S'.
The shortest paths from $s_{1,0}$ to $s_{6n-2,0}$ have length $6n-3$ and each path from $s_{1,0}$ into S' uses the state $s_{6n-2,0}$. So there is a path P leading from $s_{1,0}$ to $s_{6n-2,0}$ labeled by a prefix of v'. We distinguish the following cases:

- If P is of length $6n-3$, we just note that such path is unique and labeled by a^{6n-3}. No b occurs in v' except for the last two positions.
- If P is of length $6n-2$, it uses an edge of the form $s_{k,0} \xrightarrow{b} s_{k+1,1}$. Such edges preserve the distance to s_{6n-2}, so the rest of P must be a shortest path from $s_{k+1,1}$ to $s_{6n-2,0}$. Such paths are unique and labeled by a^{6n-2-k}. Any other b can occur only at the last position.
- If P is of length $6n-1$, it is labeled by whole v'. Because any edge labeled by b preserves or increases the distance to s_{6n-2}, the path P can use at most two of them.

□

Now we choose any $i \in \{1,\dots,m\}$ and prove that the assignment $\xi_1,\dots,\xi_n$ satisfies the clause $\bigvee_{\lambda \in C_i} \lambda$. Let $p \in \{0,1,2,3\}$ denote the number of unsatisfied literals in C_i.

As we claimed before, all tester columns corresponding to any $\lambda \in L_\phi$ have to be deactivated earlier than other columns. Namely, if $cl_i|tester|level_{x_1}|\,(1,\lambda)$ is active at time d_i, which happens if and only if λ is not satisfied by $\xi_1, \ldots, \xi_n$, the word $v'_i c_i$ must not map it to $cl_i|pipes_3|s_{1,\mu(\lambda)}$. If $cl_i|tester|level_\lambda$ is of type $\texttt{INC}(\lambda)$, the only way to ensure this is to use the letter b when the border of inactive area lies at the first row of $cl_i|tester|level_\lambda$. Thus each unsatisfied $\lambda \in C_i$ implies an occurrence of b in corresponding segment of v'_i:

Lemma 3.12. *There are at least p occurrences of the letter b in the word v'_i.*

Proof. Let $\lambda_1, \ldots, \lambda_p$ be the unsatisfied literals of C_i. From Lemma 3.10 it follows easily that

$$cl_i|tester|level_{\lambda_k}|\,(1,\lambda_k) \in S_{d_i+3\kappa(\lambda_k)}$$

for each $k \in \{1, \ldots, p\}$. The part $cl_i|tester|level_{\lambda_k}$ is of type $\texttt{INC}(\lambda_k)$, which implies that any path of the length

$$\big(\overline{d} - 3\big) - (d_i + 3\kappa(\lambda_k))$$

starting by a takes $cl_i|tester|level_{\lambda_k}|\,(1,\lambda_k)$ to the state $cl_i|\overline{\lambda}$, which lies outside $S_{\overline{d}-3}$, as it is implied by Lemma 3.8(2). We deduce that $w_{d_i+3\kappa(\lambda_k)+1} = b$. □

By Lemma 3.11 there are at most two occurrences of b in v'_i, so we get $p \leq 2$ and there is at least one satisfied literal in C_i.

Chapter 4

Computing Road Colorings

This chapter deals with computational problems related to road coloring. First, we give a multi-parameter analysis of the basic computational problem called SRCP. This consists mainly of studying a scale of various restrictions and finishes a work that was started by Roman and Drewienkowski [78, 79]. The results are contained in the above-mentioned paper [105].
Second, we give a similar analysis with respect to slightly different computational problem called SRCW. The results are not complete - they leave much space for a further research. They were presented [104] at the conference *LATA 2015* (Nice, France). An extended version was submitted to a journal.

4.1 Parameterized Complexity of SRCP

The results of this section, as well as former results of Roman and Drewienkowski [78, 79], are summarized by Tables 4.1 and 4.2. We have filled all the remaining gaps in the second table (cf. [79, Sec. 6]), so the multi-parameter analysis of SRCP is complete in the sense that NP-complete restrictions are identified and under several standard assumptions we know which restrictions are FPT and which of them have polynomial kernels.

Parameter	**Parameterized Complexity**	**Polynomial Kernel**
d	NP-complete for $d = 4, 5, \ldots$ [79]	—
$\lvert\Sigma\rvert$	NP-complete for $\lvert\Sigma\rvert = 2, 3, \ldots$ ♦	—
d and $\lvert\Sigma\rvert$	*See* Table *4.2*	—
n	FPT, running time $\mathcal{O}^\star(2^{\lvert\Sigma\rvert})$ ♦	Yes ♦

Table 4.1: The complete multi-parameter analysis of SRCP, new results marked by diamonds

	$d = 2$	$d = 3$	$d = 4, 5, \ldots$
$\lvert\Sigma\rvert = 2$	P [78]	P ♦	NPC ♦
$\lvert\Sigma\rvert = 3$	P [78]	P [79]	NPC [78]
$\lvert\Sigma\rvert = 4, 5, \ldots$	P [78]	P [79]	NPC [78]

Table 4.2: Complexity of SRCP and SRCP$^{\mathcal{SC}}$ restricted to particular values of d and $\lvert\Sigma\rvert$

By $\mathcal{SC}$ and $\mathcal{Z}$ we denote the classes of strongly connected graphs and of graphs with a state having no outgoing edges expect for loops, respectively. All the NP-completeness results hold for both the problems SRCP and SRCP$^{\mathcal{SC}}$.

4.1.1 Parameterization by the Number of States

We point out that SRCP parameterized by the number of states has a polynomial kernel, so it necessarily lies in FPT.

Theorem 4.1. *There is a polynomial kernel for* SRCP *parameterized by* $n = |Q|$.

Proof. The algorithm takes an instance of SRCP, i.e., an alphabet Σ, an admissible graph $G = (Q, E)$ with out-degrees $|\Sigma|$, and a number $d \in \mathbb{N}$. It produces another instance of size depending only on $n = |Q|$. If $d \geq z(n)$, we just solve the problem using Corollary 1.33 and output a trivial instance. Otherwise the output instance is denoted by $\Sigma', G' = (Q', E'), d'$, where

$$\begin{aligned} Q' &= Q, \\ k' &= k, \\ |I'| &= \min\{|\Sigma|, t \cdot (z(t) - 1)\}, \end{aligned}$$

and the algorithm just deletes appropriate edges in order to reduce the out-degree to $|\Sigma'|$. Let us use a procedure that:

- takes an admissible graph with out-degree $c > n \cdot (z(n) - 1)$
- for each of its vertices:
  - finds an outgoing multiedge with the largest multiplicity (which is at least $z(n)$),
  - deletes one edge from the multiedge.

Clearly the resulting graph has out-degree $c - 1$. We create the graph G' by repeating this procedure (starting with G) until the out-degree is at most $n \cdot (z(n) - 1)$.
Now we claim that

$$\begin{gathered} (I, G, k) \in \text{SRCP} \\ \Updownarrow \\ (I', G', k') \in \text{SRCP}. \end{gathered}$$

The upward implication is trivial since any coloring of G' can be extended to G and the appropriate reset word can be still used. On the other hand, let us have a coloring δ of G such that $|\delta(Q, w)| = 1$ for a word w of length at most $d < z(n)$, so it uses at most $z(n) - 1$ letters from Σ. If we delete from G all the edges labeled by non-used letters, we get a subgraph of G' because during the reduction of edges we have reduced only multiedges having more than $z(n) - 1$ edges. So we are able to color G' according to the used letters of G and synchronize it by the word w.

□

Corollary 4.2. SRCP *parameterized by* $n = |Q|$ *lies in* FPT.

4.1.2 Restriction to $|\Sigma| = 2$ and $d = 3$

Here we prove that $\text{SRCP}_{2,3}$ (i.e. SRCP restricted to $|\Sigma| = 2$ and $d = 3$) is decidable in polynomial time. If $G = (Q, E)$ is a graph, by $V_i(q)$ we denote the set of vertices from which

there is a path of length i leading to q and there is no shorter one. For any $w \in \Sigma^*$, $\mathbb{G}_w$ denotes the set of graphs with outdegree 2 that admit a coloring δ such that $\delta(Q, w) = \{q\}$ for some $q \in Q$.

Lemma 4.3. *A graph $G = (Q, E)$ lies in $\mathbb{G}_{aaa}$ if and only if there is a state $q \in Q$ such that $(q, q) \in E$ and there is a path of length at most 3 from each $r \in Q$ to q.*

Proof. If there is a coloring δ with $\delta(Q, aaa) = \{q\}$, both the claims follow immediately. On the other hand, if the loop (q, q) and all the paths leading to q are colored by a, we obtain a suitable coloring. □

Lemma 4.4. *Let $G = (Q, E) \in \mathbb{G}_{abb} \backslash \mathbb{G}_{aaa}$. Then at least one of the following two conditions holds:*

1. *There is a vertex $q \in Q$ such that each vertex has an outgoing edge leading into $V_2(q)$, or*
2. *$G \in \mathbb{G}_{aba}$.*

Proof. Let $G = (Q, E) \in \mathbb{G}_{abb} \backslash \mathbb{G}_{aaa}$. Thus G admits a coloring δ such that

$$\delta(Q, abb) = \{q\}$$

for a state $q \in Q$.

- If the coloring δ satisfies $q \notin \delta(Q, a)$, notice that each edge labeled by a has to lead into $V_2(q)$. Indeed:
  - It cannot lead to q due to $q \notin \delta(Q, a)$.
  - It cannot lead into any $r \in V_1(q)$ because in such case, using $q \notin \delta(Q, a)$, it would hold that $\delta(r, b) = q$ and thus $q \in \delta(Q, ab)$. Hence it would be necessary that $\delta(q, b) = q$. According to Lemma 4.3, a loop on q guarantees that $G \in \mathbb{G}_{aaa}$, which is a contradiction.
  - It cannot lead to $V_3(q)$, because there is no path of length 2 from $V_3(q)$ to q.

  Thus, the condition (1) holds.

- Otherwise the coloring δ satisfies $q \in \delta(Q, a)$. Denote

  $$W = \{s \in Q \mid \delta(s, b) = q\}.$$

  Now define another coloring δ' by switching the colors of the two edges leaving each state of W. Elsewhere, δ' and δ coincide. We claim that

  $$\delta'(Q, aba) = \{q\}$$

  and so the condition (2) holds. Indeed:
  - Take $s \in V_3(q)$. In δ there is a path

    $$s \xrightarrow{a} t \xrightarrow{b} u \xrightarrow{b} q. \tag{4.1}$$

    Because $s \in V_3(q)$, it holds that $t \in V_2(q)$ and $u \in V_1(q)$. It follows that $t \notin W, u \in W$ and thus in δ' there is a path

    $$s \xrightarrow{a} t \xrightarrow{b} u \xrightarrow{a} q. \tag{4.2}$$

  - Take $s \in V_2(q)$. In δ there is a path (4.1).

* If $t \in V_2(q)$, we get again that $t \notin W, u \in W$ and thus in δ' there is a path (4.2).
* Otherwise we have $t \in V_1(q)$. Because $G \notin \mathbb{G}_{aaa}$, there is no loop on q, thus $u \neq q$ and $t \notin W$. But $u \in W$, so we get a path (4.2) again.

– Take $s \in V_1(q)$. In δ' there is always an edge $s \xrightarrow{a} q$, so we need just $\delta'(q, ba) = q$. Because we assume that $q \in \delta(Q, a)$, in δ there has to be a cycle $q \xrightarrow{b} r \xrightarrow{b} q$ for some $r \in V_1(q)$. In δ' we have $q \xrightarrow{b} r \xrightarrow{a} q$.

– For $s = q$ we apply the same reasoning as for $s \in V_2(q)$.

□

Lemma 4.5. *For each G with outdegree 2 it holds that*

$$G \in \mathbb{G}_{abb} \setminus (\mathbb{G}_{aba} \cup \mathbb{G}_{aaa})$$

if and only if

- it holds that $G \notin \mathbb{G}_{aba} \cup \mathbb{G}_{aaa}$, and
- there is a vertex $q \in Q$ such that each vertex has an outgoing edge leading into $V_2(q)$.

Proof. The downward implication follows easily from Lemma 4.4. For the upward one we need only to deduce that $G \in \mathbb{G}_{abb}$. We construct the following coloring δ:

- The edges leading into $V_2(q)$ are labeled by a. If two such edges start in a common vertex, they are labeled arbitrarily.
- The other edges are labeled by b.

This works because from any state $s \in V_2(q)$ there is an edge leading to some $t \in V_1(q)$, and from t there is an edge leading to q. We have labeled both these edges by b. It follows that wherever we start, the path labeled by abb leads to q.

□

Theorem 4.6. *The problem* $\mathrm{SRCP}_{2,3}$ *in P.*

Proof. Let the algorithm test the membership of a given graph G for the following sets:

1. $\mathbb{G}_{aaa}$,
2. $\mathbb{G}_{aab} \setminus \mathbb{G}_{aaa}$,
3. $\mathbb{G}_{aba} \setminus \mathbb{G}_{aaa}$,
4. $\mathbb{G}_{abb} \setminus (\mathbb{G}_{aba} \cup \mathbb{G}_{aaa})$.

For the sets 1, 2, 3 the membership is polynomially testable due to results from [79]. Lemma 4.5 provides a polynomially testable characterization of the set 4. It is easy to see that a graph G should be accepted if and only if it lies in some of the sets.

□

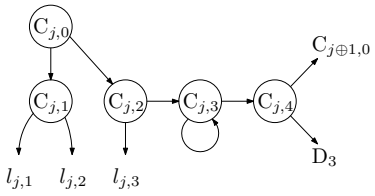


Figure 4.1: A part $\mathbf{C}_j$. Note the three edges that depend on Φ: they end in vertices labeled by literals from $\mathcal{C}_j$.

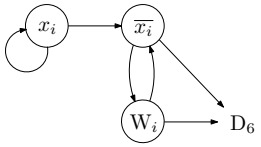


Figure 4.2: A part $\mathbf{V}_i$

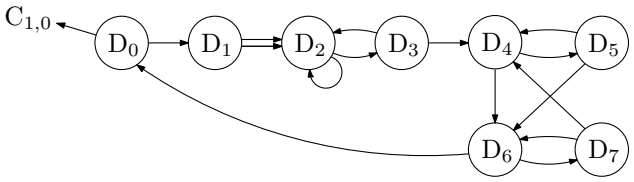


Figure 4.3: The part $\mathbf{D}$

4.1.3 Restriction to $|\Sigma| = 2$ and $d = 4$

Theorem 4.7. *The problem* $\mathrm{SRCP}_{2,4}^{\mathcal{SC}}$ *is NP-complete.*

Let us perform a reduction from 3-SAT. Consider a propositional formula of the form

$$\Phi = \bigwedge_{j=1}^{m} \mathcal{C}_j \,,$$

where $\mathcal{C}_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$ and $l_{j,k} \in \{x_1, \ldots, x_p, \overline{x_1}, \ldots, \overline{x_p}\}$ for each $j \in \{1, \ldots, m\}$, $k \in \{1, 2, 3\}$. We construct a directed multigraph $G_\Phi = (Q, E)$ with

$$|Q| = 5m + 3n + 8$$

states, each of them having exactly two outgoing edges. We describe the set Q as a disjoint union of the sets

$$Q = \mathbf{C}_1 \cup \cdots \cup \mathbf{C}_m \cup \mathbf{V}_1 \cup \cdots \cup \mathbf{V}_p \cup \mathbf{D},$$

where

$$\begin{aligned} \mathbf{C}_j &= \{\mathrm{C}_{j,0}, \mathrm{C}_{j,1}, \mathrm{C}_{j,2}, \mathrm{C}_{j,3}, \mathrm{C}_{j,4}\}\,, \\ \mathbf{V}_i &= \{x_i, \overline{x_i}, \mathrm{W}_i\}\,, \\ \mathbf{D} &= \{\mathrm{D}_0, \ldots, \mathrm{D}_7\}\,, \end{aligned}$$

for each $j \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, p\}$. The parts $\mathbf{C}_j$ correspond to clauses, the parts $\mathbf{V}_i$ correspond to variables. In each $\mathbf{V}_i$ there are two special states labeled by literals x_i and $\overline{x_i}$. All the edges of G_Φ are defined by Figures 4.1, 4.2, 4.3. Figure 4.4 gives an overall picture of G_Φ. Let us prove that

Φ is satisfiable

$\Updownarrow$

some labeling of G_Φ has a reset word of length 4.

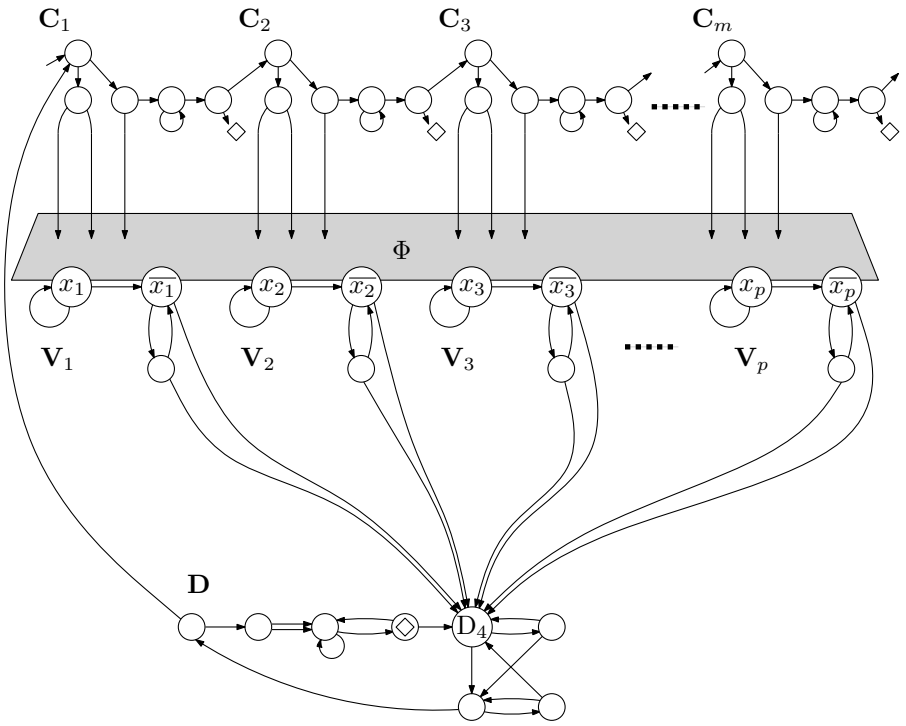


Figure 4.4: The entire G_Φ

The Upward Implication

Suppose that there is a labeling δ by letters a (solid) and b (dotted) such that a word

$$w = y_1 \dots y_4 \in \{a, b\}^4$$

satisfies

$$|\delta(Q, w)| = 1.$$

Let a be the first letter of w. By a *k-path* (resp. *k-reachable*) we understand a path of length exactly k (resp. reachable by a path of length exactly k).

Lemma 4.8. *The synchronization takes place in* D_4.

Proof. From D_1 only states from **D** are 4-reachable. From D_0 the only states within **D** that are 4-reachable are $\mathrm{D}_2, \mathrm{D}_3, \mathrm{D}_4$. From $\mathrm{C}_{1,0}$ only D_4 is 4-reachable. □

Lemma 4.9. *All edges outgoing from states of* **D** *are labeled as in Figure 4.5.*

Proof. Since D_4 is not 3-reachable from D_0 nor D_6, all the edges incoming to D_0 and D_6 are labeled by b. The remaining labeling follows easily. □

Corollary 4.10. *It holds that*

$$w = aba^2.$$

Lemma 4.11. *For each $j = 1, \dots, m$ we have*

$$\delta(\mathrm{C}_{j,0}, ab) \in \{x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}\}.$$

Proof. None of the other states 2-reachable from $\mathrm{C}_{j,0}$ offers a 2-path leading to D_4. □

Lemma 4.12. *There are no $j, l \in \{1, \dots, m\}$ and $i \in \{1, \dots, p\}$ such that*

$$\delta(\mathrm{C}_{j,0}, ab) = x_i$$

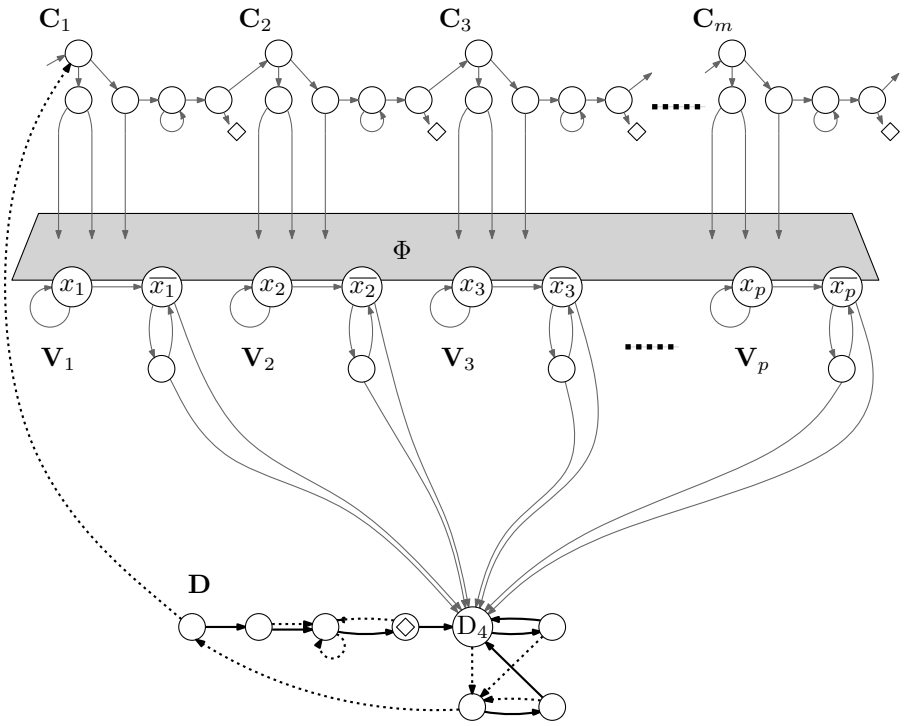


Figure 4.5: The entire G_Φ with the edges outgoing from D colored. Bold arrows: a, dotted arrows: b.

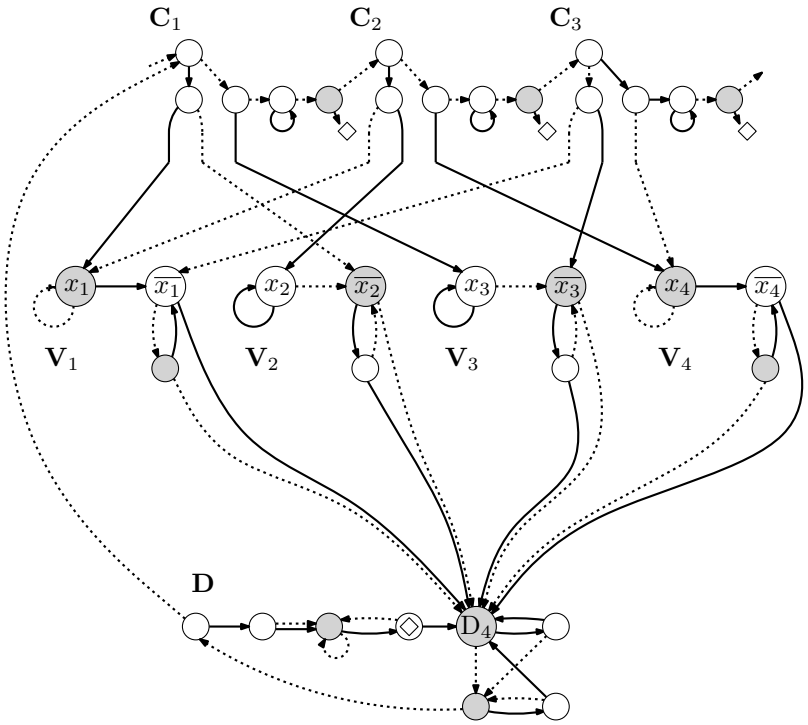


Figure 4.6: An example of G_Φ for $\Phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$. The filling marks states that are active after applying $y_1 y_2 = ab$.

and

$$\delta(C_{l,0}, ab) = \overline{x_i}.$$

Proof. If both x_i and $\overline{x_i}$ are active after applying $y_1y_2 = ab$, there have to be 2-paths labeled by a^2 from both the states $x_i, \overline{x_i}$ to D_4. It is easy to see that it is not possible to find such labeling. □

Corollary 4.13. *There is a partial assignment making all the literals*

$$\delta(\mathrm{C}_{1,0}, ab), \delta(\mathrm{C}_{2,0}, ab), \ldots, \delta(\mathrm{C}_{m,0}, ab)$$

satisfied, because none of them is the negation of another. Each clause contains some of these literals.

We are done, the existence of a satisfying assignment is guaranteed.

The Downward Implication

For a given satisfying assignment we make a coloring based on the above-mentioned ideas and the example given by Figure 4.6.

- For each j, the coloring of edges outgoing from $\mathrm{C}_{j,0}, \mathrm{C}_{j,1}, \mathrm{C}_{j,2}$ depends on which of the three literals of the clause $\mathcal{C}_j$ are satisfied by the assignment (the example assigns $x_1 = \mathbf{1}, x_2 = \mathbf{0}, x_3 = \mathbf{0}, x_4 = \mathbf{1}$). The 2-path from $\mathrm{C}_{j,0}$ labeled by ab should lead to a state labeled by a satisfied literal. The edges outgoing from $\mathrm{C}_{j,3}$ and $\mathrm{C}_{j,4}$ are always colored the same way.
- For each i, all the edges outgoing from the states of the $\mathbf{V}_i$ part are colored in one of two ways depending on the truth value assigned to x_i.
- The edges outgoing from the states of $\mathbf{D}$ admit the only possible coloring.

Note that in our example the edges outgoing from the states of $\mathbf{V}_3$ could be colored in the opposite way as well. None of the literals $x_3, \overline{x_3}$ is chosen by the coloring to satisfy a clause.

Strong Connectivity

If there is a non-negated occurrence of each x_i in Φ, the graph G_Φ is strongly connected. This assumption can be easily guaranteed by adding tautological clauses like $x_i \vee \overline{x_i} \vee \overline{x_i}$.

4.2 Fixed Parameter Complexity of SRCW

In this section we study the problem SRCW restricted to fixed values of both the alphabet size $|\Sigma|$ and the set of prescribed reset words W. We show that the problem SRCW becomes NP-complete even if restricted to $|\Sigma| = 2$ and $W = \{abb\}$ or to $|\Sigma| = 2$ and $W = \{aba\}$, which may seem surprising. Moreover, we provide a complete classification of sets $W = \{w\}$ where w is a binary word: The NP-completeness holds for $|\Sigma| = 2$ and any $w \in \{a, b\}^\star$ that does not equal a^k, b^k, a^kb, nor b^ka for any $k \geq 1$. On the other hand, for any w that matches some of these patterns, the restricted problem is solvable in polynomial time. Finally, we give partial results about SRCW restricted to strongly connected graphs. Table 4.3 summarizes the results (including the notion of a *sink device* from Definition 4.20).
First, we prove a basic lemma, which is useful in our classifications of words: adding a prefix to a prescribed reset word preserves NP-completeness:

Lemma 4.14. *Let $|\Sigma| \geq 1$ and $u, w \in \{a, b\}^\star$. Then:*

	$\lvert\Sigma\rvert = 2$		$\lvert\Sigma\rvert = 3, 4, \ldots$	
	general	S. C.	general	S.C.
$w = a^k$ $(k \geq 1)$	P	P	P	P
$w = a^k b$ $(k \geq 1)$	P	P	P	P
$w = abb$	NPC	P	**?**	**?**
$w = ava$ $\left(v \notin a^\star\right)$	NPC	NPC		
$w \neq a^k b^l$ with a s.c. incomplete sink device	NPC	NPC		
otherwise	NPC	**?**		

Table 4.3: Complexity of SRCW

1. *If* $\mathrm{SRCW}_{k,\{w\}}$ *is NP-complete, so is* $\mathrm{SRCW}_{k,\{uw\}}$.

2. *If* $\mathrm{SRCW}^{\mathcal{Z}}_{k,\{w\}}$ *is NP-complete, so is* $\mathrm{SRCW}^{\mathcal{Z}}_{k,\{uw\}}$.

Proof. We perform a polynomial-time reduction from $\mathrm{SRCW}_{k,\{w\}}$ to $\mathrm{SRCW}_{k,\{wu\}}$. For a given graph $G = (Q, E)$ we construct a suitable graph $\overline{G} = \left(\overline{Q}, \overline{E}\right)$, with the additional property that $\overline{G} \in \mathcal{Z}$ whenever $G \in \mathcal{Z}$. Let

$$\overline{Q} \quad = \quad Q \cup (Q \times \{1, \ldots, |u|\})$$

and let $\overline{E}$ consist of E and the additional edges

$$(s, 1) \rightrightarrows (s, 2) \rightrightarrows \cdots \rightrightarrows (s, |u|) \rightrightarrows s$$

for each $s \in Q$. Suppose that there is a coloring δ of G such that $|\delta(Q, w)| = 1$. If we use the coloring δ in $\overline{G}$ to color the edges within E, we get a unique coloring $\overline{\delta}$ of $\overline{G}$ that satisfies $\left|\overline{\delta}\left(\overline{Q}, uw\right)\right| = 1$. On the other hand, let $\overline{\delta}$ be a coloring of $\overline{G}$ such that $\left|\overline{\delta}\left(\overline{Q}, uw\right)\right| = 1$. Observe that $\overline{\delta}\left(\overline{Q}, u\right) = Q$ and thus $\left|\overline{\delta}(Q, w)\right| = 1$. The coloring $\overline{\delta}$, restricted to Q, gives a coloring of G with $|\delta(Q, w)| = 1$. □

4.2.1 A Complete Classification of Binary Words According to Complexity of $\mathrm{SRCW}_{2,\{w\}}$

The theorem below presents one of the main results of the present paper. Assuming that P does not equal NP, it introduces an exact dichotomy concerning the words over binary alphabets. Let us fix the following partition of $\{a, b\}^\star$:

$$T_1 = \left\{a^k, b^k \mid k \geq 0\right\}, \qquad T_3 = \left\{a^l b^k, b^l a^k \mid k \geq 2, l \geq 1\right\},$$
$$T_2 = \left\{a^k b, b^k a \mid k \geq 1\right\}, \qquad T_4 = \{a, b\}^\star \setminus (T_1 \cup T_2 \cup T_3).$$

For the NP-completeness reductions throughout the present paper we use a suitable variant of the satisfiability problem. The following is proved in Section 4.2.3 using the Schaefer's dichotomy theorem:

Lemma 4.15. *It holds that W-SAT is NP-complete.*

W-SAT	
Input:	Finite set X of *variables*, finite set $\Phi \subseteq X^4$ of *clauses*.
Output:	Is there an assignment $\xi : X \to \{\mathbf{0}, \mathbf{1}\}$ such that for each clause $(z_1, z_2, z_3, z_4) \in \Phi$ it holds that: (1) $\xi(z_i) = \mathbf{1}$ for some i, (2) $\xi(z_i) = \mathbf{0}$ for some $i \in \{1, 2\}$, (3) $\xi(z_i) = \mathbf{0}$ for some $i \in \{3, 4\}$?

In this section we use reductions from W-SAT to prove the NP-completeness of $\mathrm{SRCW}_{2,\{w\}}$ for each $w \in T_3$ and $w \in T_4$. In the case of $w \in T_4$ the reduction produces only graphs having sink states. This shows that for $w \in T_4$ the problem $\mathrm{SRCW}^{\mathcal{Z}}_{2,\{w\}}$ is NP-complete as well, which turns out to be very useful in Section 4.2.2, where we deal with strongly connected graphs. For $w \in T_3$ we also prove NP-completeness, but we use automata without sink states. We show that the cases with $w \in T_1 \cup T_2$ are decidable in polynomial time.

In all the figures below we use bold solid arrows and bold dotted arrows for the letters a and b respectively.

Theorem 4.16. *Let $w \in \{a, b\}^\star$.*

1. *If $w \in T_1 \cup T_2$, the problem $\mathrm{SRCW}_{2,\{w\}}$ is solvable in polynomial time.*
2. *If $w \in T_3 \cup T_4$, the problem $\mathrm{SRCW}_{2,\{w\}}$ is NP-complete. Moreover, if $w \in T_4$, the problem $\mathrm{SRCW}^{\mathcal{Z}}_{2,\{w\}}$ is NP-complete.*

Proof for $w \in T_1$. It is easy to see that $G \in \mathbb{G}_{a^k}$ if and only if there is $q_0 \in Q$ such that there is a loop on q_0 and for each $s \in Q$ we have $\mathrm{d}_G(s, q_0) \leq k$. □

Proof for $w \in T_2$. For a fixed $q_0 \in Q$, we denote $Q_1 = \{s \in Q \mid s \longrightarrow q_0\}$ and

$$R = \{s \in Q_1 \mid H_1 \text{ has a cycle reachable from } s\},$$

where H_1 is obtained from $G[Q_1]$ by decreasing multiplicity by 1 for each edge ending in q_0. If $q_0 \notin Q_1$, we have $H_1 = G[Q_1]$. Let us prove that $G \in \mathbb{G}_{a^k b}$ if and only if there is $q_0 \in Q$ such that:

1. It holds that $\mathrm{d}_G(s, q_0) \leq k + 1$ for each $s \in Q$.
2. For each $s \in Q$ there is a $q \in R$ such that $\mathrm{d}_G(s, q) \leq k$.

First, check the backward implication. For each $r \in R$, we color by b an edge of the form $r \longrightarrow q_0$ that does not appear in H_1. Then we fix a forest of shortest paths from all the vertices of $Q \backslash R$ into R. Due to the second condition above, the branches have length at most k. We color by a the edges used in the forest. We have completely specified a coloring of edges. Now, for any $s \in Q$ a prefix a^j of $a^k b$ takes us into R, the factor a^{k-j} keeps us inside R, and with the letter b we end up in q_0.
As for the forward implication, the first condition is trivial. For the second one, take any $s \in Q$ and denote $s_j = \delta\big(s, a^j\big)$ for $j \geq 0$. Clearly, $s_k \in Q_1$, but we show also that $s_k \in R$, so we can set $q = s_k$ in the last condition. Indeed, whenever $s_j \in Q_1$ for $j \geq k$, we remark that $\delta\big(s_{j-k+1}, a^k\big) = q_0$ and thus $s_{j+1} \in Q_1$ as well. Since j can grow infinitely, there is a cycle within Q_1 reachable from s_k. □

Proof for $w \in T_3$. Due to Lemma 4.14, it is enough to deal with $w = ab^k$ for each $k \geq 2$. For a polynomial-time reduction from W-SAT, take an instance $X = \{x_1, \ldots, x_n\}$, $\Phi = \{C_1, \ldots, C_m\}$, where $C_j = (z_{j,1}, z_{j,2}, z_{j,3}, z_{j,4})$ for each $j = 1, \ldots, m$. We construct the graph $G_{k,\phi} = (Q, E)$ defined by Figure 4.7. Note that:

- In Fig. 4.7, states are represented by discs. For each $j = 1, \ldots, m$, the edges outgoing from C'_i and C''_i represent the formula Φ by leading to the states $z_{j,1}, z_{j,2}, z_{j,3}, z_{j,4} \in \{x_1, \ldots, x_n\} \subseteq Q$.
- In the case of $k = 2$ the state $\mathrm{V}_{i,2}$ does not exist, so we set $x_i \longrightarrow \mathrm{D}_0$ and $\mathrm{V}_{i,1} \longrightarrow \mathrm{D}_0$ instead of $x_i \longrightarrow \mathrm{V}_{i,2}$ and $\mathrm{V}_{i,1} \longrightarrow \mathrm{V}_{i,2}$.

We show that $G_{k,\Phi} \in \mathbb{G}_{ab^k}$ if and only if there is an assignment $\xi : X \to \{\mathbf{0}, \mathbf{1}\}$ satisfying the conditions given by Φ. The presented construction of $G_{k,\Phi}$ can be obviously performed in time polynomial in the size of X and Φ.
First, let there be a coloring δ of $G_{k,\Phi}$ such that $\left|\delta\big(Q, ab^k\big)\right| = 1$. Necessarily $\delta\big(Q, ab^k\big) = \{\mathrm{D}_0\}$, while there is no loop on D_0. Indeed, let C_j and $C_{j'}$ be clauses that do not share any variable. Due to possible adding of artificial clauses to Φ, we can assume that C_j and $\mathrm{C}_{j'}$ exist. It is easy to see that D_0 is the only vertex reachable from C_j and $\mathrm{C}_{j'}$ by paths of length at most $k+1$. We use this fact to observe that whenever $x_i \in \delta(Q, a)$, the edges outgoing from $x_i, \mathrm{V}_{i,1}, \ldots, \mathrm{V}_{i,k-1}$ must be colored according to Figure 4.8, but if $x_i \in \delta(Q, ab)$, then they must be colored according to Figure 4.9. Indeed: First, let $x_i \in \delta(Q, a)$. Then we have $\delta\big(x_i, b^k\big) = \mathrm{D}_0$. Since there is no loop on D_0, there are at most two paths of length k leading from x_i to D_0. The first one is

$$x_i \longrightarrow \mathrm{V}_{i,1} \longrightarrow \mathrm{V}_{i,2} \longrightarrow \cdots \longrightarrow \mathrm{V}_{i,k-1} \longrightarrow \mathrm{D}_0.$$

If we label this path by b^k, we get exactly Figure 74. If $k > 2$, there is also the second path

$$x_i \longrightarrow \mathrm{V}_{i,2} \longrightarrow \mathrm{V}_{i,2} \longrightarrow \mathrm{V}_{i,3} \longrightarrow \cdots \longrightarrow \mathrm{V}_{i,k-1} \longrightarrow \mathrm{D}_0,$$

which uses the loop on $\mathrm{V}_{i,2}$. But labeling this path by b^k involves labeling both the edges outgoing from $\mathrm{V}_{i,2}$ by b, which is a contradiction. Second, let $x_i \in \delta(Q, ab)$. The path

$$x_i \longrightarrow \mathrm{V}_{i,2} \longrightarrow \mathrm{V}_{i,3} \longrightarrow \cdots \longrightarrow \mathrm{V}_{i,k-1} \longrightarrow \mathrm{D}_0$$

is the only path of length $k-1$ leading from x_i to D_0, so it is necessarily labeled by b^{k-1} and we get the coloring depicted by Figure 74.
Let $\xi(x_i) = \mathbf{1}$ if $x_i \in \delta(Q, ab)$ and $\xi(x_i) = \mathbf{0}$ otherwise. Choose any $j \in \{1, \ldots, m\}$ and observe that

$$\xi(\delta(\mathrm{C}_j, ab)) = \mathbf{1}, \quad \xi\big(\delta\big(\mathrm{C}'_j, a\big)\big) = \mathbf{0}, \quad \xi\big(\delta\big(\mathrm{C}''_j, a\big)\big) = \mathbf{0}.$$

Indeed, $\delta(\mathrm{C}_j, ab) = x_i \in \delta(Q, ab)$, so ξ assigns $\mathbf{1}$ to x_i by definition. The remaining two claims are similar.
Thus we can conclude that all the conditions from the definition of W-SAT hold for the clause C_j. Indeed, the graph $G_{k,\Phi}$ is constructed such that

$$\begin{aligned} \delta(\mathrm{C}_j, ab) &\in \{z_{j,1}, z_{j,2}, z_{j,3}, z_{j,4}\}, \\ \delta\big(\mathrm{C}'_j, a\big) &\in \{z_{j,1}, z_{j,2}\}, \\ \delta\big(\mathrm{C}''_j, a\big) &\in \{z_{j,3}, z_{j,4}\}. \end{aligned}$$

On the other hand, let ξ be a satisfying assignment of Φ. For each j we color the edges outgoing from $\mathrm{C}_j, \mathrm{C}'_j, \mathrm{C}''_j$ such that the ab-path from C_j leads to the $z_{j,p}$ with $\xi(z_{j,p}) = \mathbf{1}$ and the a-paths from $\mathrm{C}'_j, \mathrm{C}''_j$ lead to the $z_{j,p'}$ and $z_{j,p''}$ with $\xi(z_{j,p'}) = \mathbf{0}, \xi(z_{j,p''}) = \mathbf{0}$, where

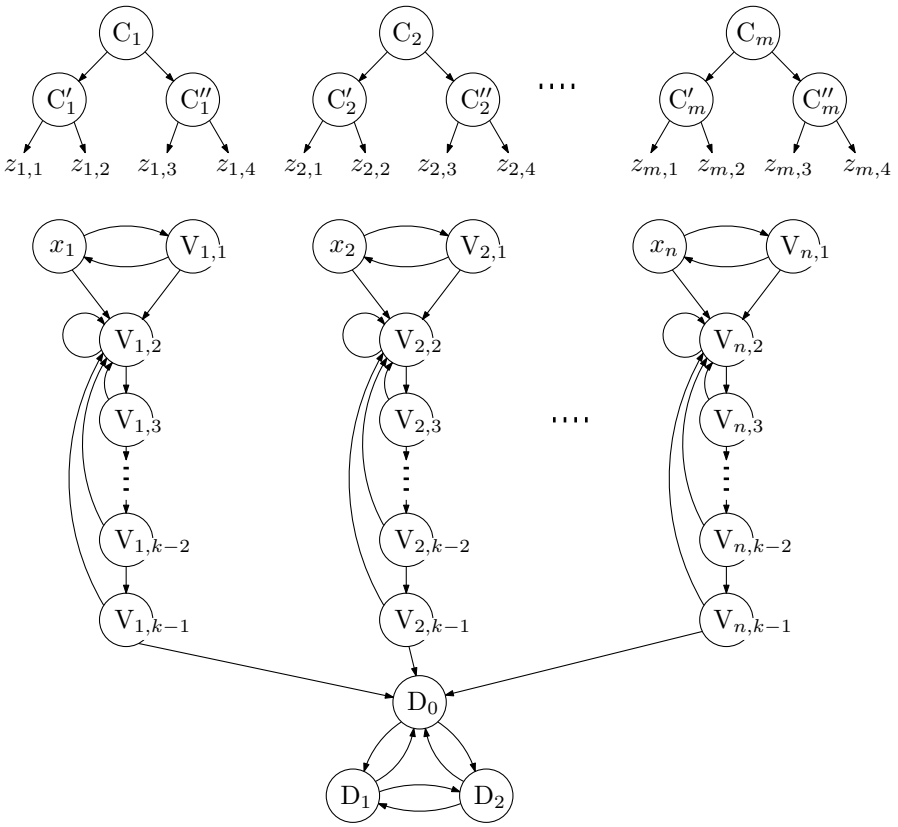


Figure 4.7: The graph $G_{k,\Phi}$ reducing W-SAT to $\mathrm{SRCW}_{|\Sigma|=2,W=\{ab^k\}}$ for $k \geq 2$

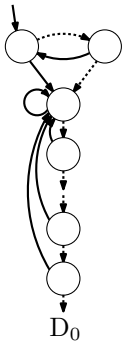


Figure 4.8: A coloring corresponding to $\xi(x_i) = \mathbf{0}$

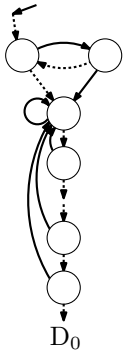


Figure 4.9: A coloring corresponding to $\xi(x_i) = \mathbf{1}$

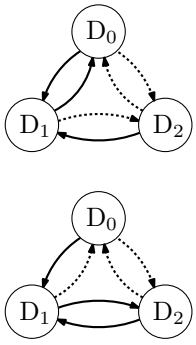


Figure 4.10: Colorings for k even (top) and odd (bottom)

$p' \in \{1,2\}, p'' \in \{3,4\}$. For the edges outgoing from $x_i, \mathrm{V}_{i,1}, \ldots, \mathrm{V}_{i,k-1}$ we use Figure 4.8 if $\xi(x_i) = \mathbf{0}$ and Figure 4.9 if $\xi(x_i) = \mathbf{1}$. The transitions within $\mathrm{D}_0, \mathrm{D}_1, \mathrm{D}_2$ are colored according to Figure 4.10, depending on the parity of k. Observe that for each $i \in \{1, \ldots, n\}$ we have

$$x_i \notin \delta(Q, ab) \quad \text{if} \quad \xi(x_i) = \mathbf{0}, \tag{4.3}$$

$$x_i \notin \delta(Q, a) \quad \text{if} \quad \xi(x_i) = \mathbf{1}. \tag{4.4}$$

Indeed: first, let $\xi(x_i) = \mathbf{0}$. For the edges outgoing from $x_i, \mathrm{V}_{i,1}, \ldots, \mathrm{V}_{i,k-1}$ we used Figure 4.8, so any possible ab-path ending in x_i starts in some C_j. But any ab-path starting in some C_j ends in the $z_{j,i}$ with $\xi(z_{j,i}) = \mathbf{1}$. Second, let $\xi(x_i) = \mathbf{1}$. For the edges outgoing from $x_i, \mathrm{V}_{i,1}, \ldots, \mathrm{V}_{i,k-1}$ we used Figure 4.9, so any possible a-transition ending in x_i starts in some C'_j or C''_j. But any a-transition starting in some C'_j or C''_j ends in the $z_{j,i'}$ and $z_{j,i''}$ with $\xi(z_{j,i'}) = \mathbf{0}$ and $\xi(z_{j,i''}) = \mathbf{0}$.
Using (4.3) and (4.4) we can check that $\delta(Q, w) = \{\mathrm{D}_0\}$: First, recall that $w = ab^k$ and describe the set $\delta(Q, ab)$. We have

$$\begin{aligned}
\mathrm{C}_j &\xrightarrow{ab} x_i \text{ for some } i \text{ with } \xi(x_i) = \mathbf{1},\\
\mathrm{C}'_j &\xrightarrow{ab} \mathrm{V}_{i,1} \text{ for some } i \text{ with } \xi(x_i) = \mathbf{0},\\
\mathrm{C}''_j &\xrightarrow{ab} \mathrm{V}_{i,1} \text{ for some } i \text{ with } \xi(x_i) = \mathbf{0},\\
x_i &\xrightarrow{ab} \begin{cases} \mathrm{V}_{i,3} & \text{if } \xi(x_i) = \mathbf{0} \text{ and } k \geq 4,\\ \mathrm{D}_0 & \text{if } \xi(x_i) = \mathbf{0} \text{ and } k = 3,\\ \mathrm{D}_2 & \text{if } \xi(x_i) = \mathbf{0} \text{ and } k = 2,\\ x_i & \text{if } \xi(x_i) = \mathbf{1}, \end{cases}\\
\mathrm{V}_{i,1} &\xrightarrow{ab} \begin{cases} \mathrm{V}_{i,3} & \text{if } \xi(x_i) = \mathbf{1} \text{ and } k \geq 4,\\ \mathrm{D}_0 & \text{if } \xi(x_i) = \mathbf{1} \text{ and } k = 3,\\ \mathrm{D}_2 & \text{if } \xi(x_i) = \mathbf{1} \text{ and } k = 2,\\ \mathrm{V}_{i,1} & \text{if } \xi(x_i) = \mathbf{0}, \end{cases}\\
\mathrm{V}_{i,h} &\xrightarrow{ab} \begin{cases} \mathrm{V}_{i,3} & \text{if } k \geq 4\\ \mathrm{D}_0 & \text{if } k = 3 \end{cases} \text{for } h \in \{2, \ldots, k-1\},\\
\mathrm{D}_0, \mathrm{D}_1, \mathrm{D}_2 &\xrightarrow{ab} \begin{cases} \mathrm{D}_2 & \text{if } k \text{ is even},\\ \mathrm{D}_0 & \text{if } k \text{ is odd}. \end{cases},
\end{aligned}$$

Thus, we can easily observe that

$$\delta(Q, ab) \quad \subseteq \quad \{\mathrm{V}_{i,1}, \mathrm{V}_{i,3} \mid \xi(x_i) = \mathbf{0}\} \cup \{x_i, \mathrm{V}_{i,3} \mid \xi(x_i) = \mathbf{1}\} \cup \{\mathrm{D}_p\},$$

where $p = 2$ for k even and $p = 0$ for k odd. Now it is easy to conclude that $\delta\big(s, b^{k-1}\big) = \mathrm{D}_0$ for each $s \in \delta(Q, ab)$. □

Proof for $w \in T_4$. Any $w \in T_4$ can be written as $w = va^j b^k a^l$ or $w = vb^j a^k b^l$ for $j, k, l \geq 1$. Due to Lemma 4.14 it is enough to deal with $w = ab^k a^l$ for each $k, l \geq 1$. Take an instance of W-SAT as above and construct the graph $G_{w,\Phi} = (Q, E)$ defined by Figure 4.11. Note that:

- In the case of $l = 1$, the state $\mathrm{Z}_{i,1}$ does not exist, so we set $\mathrm{W}'_i \longrightarrow \mathrm{D}_0$ and $\mathrm{V}_{i,k-1} \longrightarrow \mathrm{D}_0$ instead of $\mathrm{W}'_i \longrightarrow \mathrm{Z}_{i,1}$ and $\mathrm{V}_{i,k-1} \longrightarrow \mathrm{Z}_{i,1}$.
- In the case of $k = 1$, the state $V_{i,1}$ does not exist, so we set $x_i \longrightarrow \mathrm{Z}_{i,1}$ (or $x_i \longrightarrow \mathrm{D}_0$ if $l = 1$) and $x_i \longrightarrow \mathrm{W}_i$ instead of $x_i \rightrightarrows \mathrm{V}_{i,1}$.

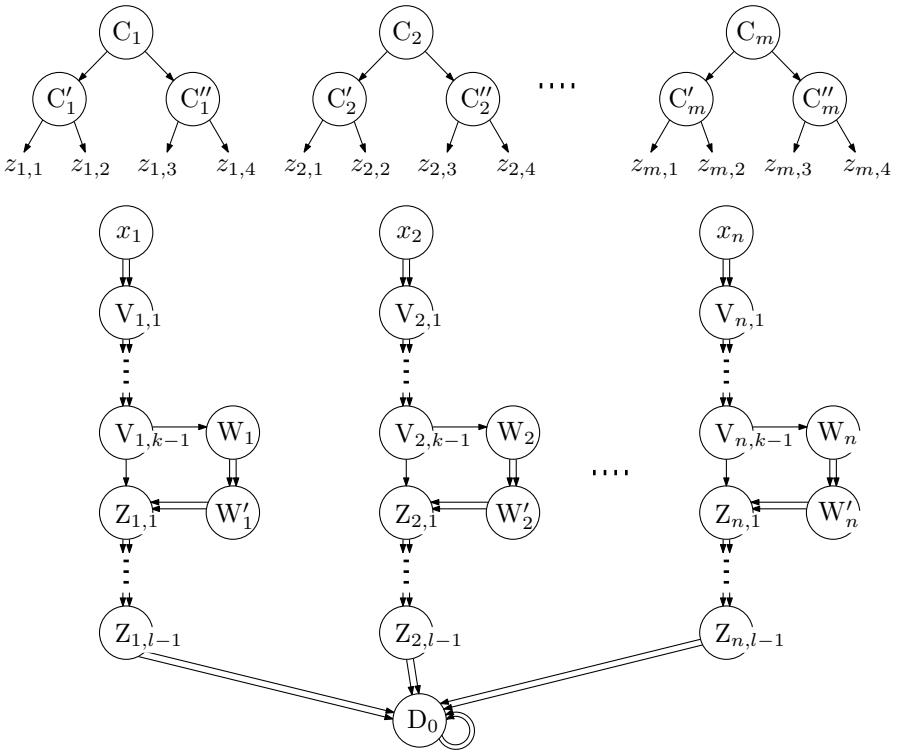


Figure 4.11: The graph $G_{w,\Phi}$ reducing W-SAT to $\mathrm{SRCW}^{\mathcal{Z}}_{|\Sigma|=2,W=\{ab^ka^l\}}$ for $k,l \geq 1$

Let there be a coloring δ of $G_{w,\Phi}$ such that $|\delta(Q,w)| = 1$. Observe that $\delta(Q,w) = \{\mathrm{D}_0\}$. (Indeed, let C_j and $C_{j'}$ be clauses that does not share any variable. It is easy to see that D_0 is the only vertex reachable from C_j and $\mathrm{C}_{j'}$ by paths of length at most $k+l+1$).
Next, we claim that

$$x_i \in \delta(Q,a) \quad \Rightarrow \quad \mathrm{V}_{i,k-1} \xrightarrow{b} \mathrm{Z}_{i,1}, \tag{4.5}$$

$$x_i \in \delta(Q,ab) \quad \Rightarrow \quad \mathrm{V}_{i,k-1} \xrightarrow{a} \mathrm{Z}_{i,1}. \tag{4.6}$$

Note that if $k = 1$ or $l = 1$, the claim should hold for x_i instead of $\mathrm{V}_{i,k-1}$ or D_0 instead of $\mathrm{Z}_{i,1}$ respectively. As for (4.5), let $x_i \in \delta(Q,a)$. If $k \geq 2$, we have $\mathrm{V}_{i,k-1} \in \delta\big(ab^{k-1}\big)$ and there is exactly one path of length at most $l+1$ leading from $\mathrm{V}_{i,k-1}$ to D_0:

$$\mathrm{V}_{i,k-1} \longrightarrow \mathrm{Z}_{i.1} \longrightarrow \cdots \longrightarrow \mathrm{Z}_{i,l-1} \longrightarrow \mathrm{D}_0. \tag{4.7}$$

Thus, we have $\mathrm{V}_{i,k-1} \xrightarrow{b} \mathrm{Z}_{i,1}$. If $k = 1$, we use the fact that there is exactly one path of length at most $l+1$ leading from x_i to D_0. As for (4.6), let $x_i \in \delta(Q,ab)$. If $k \geq 2$, we have $\mathrm{V}_{i,k-1} \in \delta\big(ab^k\big)$. The path (4.7) above is the only path of length at most l leading from $\mathrm{V}_{i,k-1}$ to D_0. Thus, we have $\mathrm{V}_{i,k-1} \xrightarrow{a} \mathrm{Z}_{i,1}$. If $k = 1$, we use the unique path of length at most l leading from x_i to D_0.
Let $\xi(x_i) = \mathbf{1}$ if $x_i \in \delta(Q,ab)$ and $\xi(x_i) = \mathbf{0}$ otherwise. We choose any $j \in \{1,\ldots,m\}$ and observe that

$$\xi(\delta(\mathrm{C}_j,ab)) = \mathbf{1}, \qquad \xi\big(\delta\big(\mathrm{C}'_j,a\big)\big) = \mathbf{0}, \qquad \xi\big(\delta\big(\mathrm{C}''_j,a\big)\big) = \mathbf{0}.$$

Then we conclude that all the conditions from the definition of W-SAT hold for the clause C_j, since

$$\begin{aligned} \delta(\mathrm{C}_j,ab) &\in \{z_{j,1},z_{j,2},z_{j,3},z_{j,4}\}, \\ \delta\big(\mathrm{C}'_j,a\big) &\in \{z_{j,1},z_{j,2}\}, \end{aligned}$$

$$\delta\big(\mathrm{C}''_j, a\big) \quad \in \quad \{z_{j,3}, z_{j,4}\}$$

due to the construction of $G_{k,\Phi}$.
On the other hand, let ξ be a satisfying assignment of Φ. For each j, we color the edges outgoing from $\mathrm{C}_j, \mathrm{C}'_j, \mathrm{C}''_j$ as we did in the case of T_3. For each i, we put $\mathrm{V}_{i,k-1} \xrightarrow{a} \mathrm{Z}_{i,1}, \mathrm{V}_{i,k-1} \xrightarrow{b} \mathrm{W}_i$ if $\xi(x_i) = \mathbf{1}$ and the reversed variant if $\xi(x_i) = \mathbf{0}$. Note that if $k = 1$ or $l = 1$, we consider x_i instead of $\mathrm{V}_{i,k-1}$ or D_0 instead of $\mathrm{Z}_{i,1}$ respectively. Let us show that $\delta(Q, w) = \{\mathrm{D}_0\}$. Denote

$$C = \big\{\mathrm{C}_j, \mathrm{C}'_j, \mathrm{C}''_j \mid j = 1, \ldots, m\big\}$$

and observe that from any vertex s of $Q \backslash C$, all the paths of length $k+l+1$ starting in s end in D_0. Thus, it is enough to show that $\delta(C, w) = \{\mathrm{D}_0\}$. Choose any $j = 1, \ldots, m$. First, observe that $\delta\left(\mathrm{C}_j, ab\right) = x_i$ such that $\xi(x_i) = \mathbf{1}$. Thus, the edge $\mathrm{V}_{i,k-1} \longrightarrow \mathrm{Z}_{i,1}$ is labeled by a and the path

$$\mathrm{C}_j \xrightarrow{ab} x_i \xrightarrow{b} \mathrm{V}_{i,1} \xrightarrow{b} \cdots \xrightarrow{b} \mathrm{V}_{i,k-1} \xrightarrow{a} \mathrm{Z}_{i,1}$$

guarantees that $\delta\big(\mathrm{C}_j, ab^k a\big) = \mathrm{Z}_{i,1}$ and thus $\delta\big(\mathrm{C}_j, ab^k a^l\big) = \mathrm{D}_0$. Second, choose $s \in \big\{\mathrm{C}'_j, \mathrm{C}''_j\big\}$ and observe that $\delta\left(s, ab\right) = \mathrm{V}_{i,1}$ such that $\xi(x_i) = \mathbf{0}$. Thus, the edge $\mathrm{V}_{i,k-1} \longrightarrow \mathrm{Z}_{i,1}$ is labeled by b and the path

$$s \xrightarrow{ab} \mathrm{V}_{i,1} \xrightarrow{b} \cdots \xrightarrow{b} \mathrm{V}_{i,k-1} \xrightarrow{b} \mathrm{Z}_{i,1}$$

guarantees that $\delta\big(\mathrm{C}_j, ab^k\big) = \mathrm{Z}_{i,1}$ and thus $\delta\big(\mathrm{C}_j, ab^k a^l\big) = \mathrm{D}_0$. □

4.2.2 A Partial Classification of Binary Words According to Complexity of $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}}$

Clearly, for any $w \in T_1 \cup T_2$ we have $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}} \in \mathrm{P}$. In Section 4.2.2 we show that

$$\mathrm{SRCW}^{\mathcal{SC}}_{2,\{abb\}} \in \mathrm{P},$$

which is a surprising result because the general $\mathrm{SRCW}_{2,\{w\}}$ is NP-complete for any $w \in T_3$, including $w = abb$. We are not aware of any other words that witness this difference between $\mathrm{SRCW}^{\mathcal{SC}}$ and SRCW.
In Section 4.2.2 we introduce a general method using *sink devices* that allows us to prove the NP-completeness of $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}}$ for infinitely many words $w \in T_4$, including any $w \in T_4$ with the first and last letter being the same. However, we are not able to apply the method to each $w \in T_4$.

A Polynomial-Time Case

A graph $G = (Q, E)$ is said to be *k-lifting* if there exists $q_0 \in Q$ such that for each $s \in Q$ there is an edge leading from s into $V_k(q_0)$. Instead of 2-lifting we just say *lifting*.

Lemma 4.17. *If G is a k-lifting graph, then $G \in \mathbb{G}_{ab^k}$.*

Proof. We produce a suitable coloring of $G = (Q, E)$ as follows: For each $s \in Q$ choose an edge leading from s into $V_k(q_0)$ and color it by a. All the other edges are colored by b. Then $\delta(a, Q) \subseteq V_2(q_0)$ and $\delta\big(ab^k\big) = \{q_0\}$. □

Lemma 4.18. *If G is strongly connected, G is not lifting, and $G \in \mathbb{G}_{abb}$ via δ and q_0, then δ has no b-transition ending in $V_2(q_0) \cup V_3(q_0)$. Moreover, $V_3(q_0) = \emptyset$.*

Proof. First, suppose for a contradiction that some $s \in V_2(q_0) \cup V_3(q_0)$ has an incoming b-transition. Together with its outgoing b-transition we have

$$r \xrightarrow{b} s \xrightarrow{b} t,$$

where $s \neq q_0$ and $t \neq q_0$. Due to the strong connectivity there is a shortest path P from q_0 to r (possibly of length 0 if $r = q_0$). The path P is made of b-transitions. Indeed, if there were some a-transitions, let $r' \xrightarrow{a} r''$ be the last one. The abb-path outgoing from r' ends in $\delta(r'', bb)$, which either lies on P or in $\{s, t\}$, so it is different from q_0 and we get a contradiction.
It follows that $\delta(q_0, b) \neq q_0$ and $\delta(q_0, bb) \neq q_0$, so there cannot be any a-transition incoming to q_0. Hence for any $s \in V_1(q_0)$ there is a transition $s \xrightarrow{b} q_0$ and thus there is no a-transition ending in $V_1(q_0)$. Because there is also no a-transition ending in $V_3(q_0)$, all the a-transitions end in $V_2(q_0)$ and thus G is lifting, which is a contradiction.
Second, we show that $V_3(q_0)$ is empty. Suppose that $s \in V_3(q_0)$. No a-transition comes to s since there is no path of length 2 from s to q_0. Thus, s has no incoming transition, which contradicts the strong connectivity.

□

Theorem 4.19. $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{abb\}}$ *is decidable in polynomial time.*

Proof. As the input we have a strongly connected $G = (Q, E)$. Suppose that q_0 is fixed (we can just try each $q_0 \in Q$) and so we should decide if there is some δ with $\delta(Q, abb) = \{q_0\}$. First we do some preprocessing:

- If G is lifting, according to Lemma 4.17 we accept.
- If $V_3(q_0) \neq \emptyset$, according to Lemma 4.18 we reject.
- If there is a loop on q_0, we accept, since due to $V_3(q_0) = \emptyset$ we have $G \in \mathbb{G}_{bb}$.

If we are still not done, we try to find some labeling δ, assuming that none of the three conditions above holds. We deduce two necessary properties of δ. First, Lemma 4.18 says that we can safely label all the transitions ending in $V_2(q_0)$ by a. Second, we have $q_0 \in \delta(Q, a)$. Indeed, otherwise all the transitions incoming to q_0 are labeled by b, and there cannot be any a-transition ending in $V_1(q_0)$ because we know that the b-transition outgoing from q_0 is not a loop. Thus G is lifting, which is a contradiction.
Let the sets $B_1, \ldots, B_\beta$ denote the connected components (not necessarily strongly connected) of $G[V_1(q_0)]$. Note that maximum out-degree in $G[V_1(q_0)]$ is 1. Let $e = (r, s), e' = (s, t)$ be consecutive edges with $s, t \in V_1(q_0)$ and $r \in Q$. Then the labeling δ has to satisfy

$$e \text{ is labeled by } a \;\Leftrightarrow\; e' \text{ is labeled by } b.$$

Indeed:

- The left-to-right implication follows easily from the fact that there is no loop on q_0.
- As for the other one, suppose for a contradiction that both e', e are labeled by b. We can always find a path P (possibly trivial) that starts outside $V_1(q_0)$ and ends in r. Let $\overline{r}$ be the last vertex on P that lies in $\delta(Q, a)$. Such vertex exists because we have $V_2(q_0) \cup \{q_0\} \subseteq \delta(Q, a)$ and $V_3(q_0) = \emptyset$. Now we can deduce that $\delta(\overline{r}, bb) \neq q_0$, which is a contradiction.

It follows that for each B_i there are at most two possible colorings of its inner edges (fix *variant* **0** and *variant* **1** arbitrarily). Moreover, a labeling of any edge incoming to B_i enforces a particular variant for whole B_i.
Let the set A contain the vertices $s \in V_2(q_0) \cup \{q_0\}$ whose outgoing transitions lead both into $V_1(q_0)$. Edges that start in vertices of $(V_2(q_0) \cup \{q_0\}) \setminus A$ have only one possible way of coloring due to Lemma 4.18, while for each vertex of A there are two possibilities. Now any possible coloring can be described by $|A| + \beta$ Boolean propositions:

$$\begin{aligned} \mathbf{x}_s &\equiv e_s \text{ is labeled by } a \\ \mathbf{y}_B &\equiv B \text{ is labeled according to variant } \mathbf{1} \end{aligned}$$

for each $s \in A$ and $B \in \{B_1, \ldots, B_\beta\}$, where e_s is a particular edge outgoing from s. Moreover, the claim $\delta(Q, abb) = \{q_0\}$ can be equivalently formulated as a conjunction of implications of the form $\mathbf{x}_s \to \mathbf{y}_B$, so we reduce the problem to 2-SAT: For each $s \in A$, with e_s, e'_s leading into B, B' respectively, we construct two implications. For $s \neq q_0$ they are

$$\mathbf{x}_s \quad \to \quad \begin{cases} \mathbf{y}_B & \text{if labeling } e_s \text{ by } a \text{ enforces variant } \mathbf{1} \text{ for } B \\ \neg\mathbf{y}_B & \text{if labeling } e_s \text{ by } a \text{ enforces variant } \mathbf{0} \text{ for } B \end{cases}$$

$$\neg\mathbf{x}_s \quad \to \quad \begin{cases} \mathbf{y}_{B'} & \text{if labeling } e'_s \text{ by } a \text{ enforces variant } \mathbf{1} \text{ for } B' \\ \neg\mathbf{y}_{B'} & \text{if labeling } e'_s \text{ by } a \text{ enforces variant } \mathbf{0} \text{ for } B' \end{cases}$$

Now, $\delta(Q, abb) = \{q_0\}$ is equivalent to the conjunction of all the implications. Indeed, from the claims above it follows easily that the implications are necessary for δ being abb-synchronizing. On the other hand, a satisfying assignment of the variables induces a labeling of G that, as can be easily checked, is abb-synchronizing.

□

NP-Complete Cases

We introduce a method based on *sink devices* to prove the NP-completeness for a wide class of words even under the restriction to strongly connected graphs.
In the proofs below we use the notion of a *partial finite automaton* (*PFA*), which can be defined as a triple $P = (Q, \Sigma, \delta)$, where Q is a finite set of states, Σ is a finite alphabet, and δ is a partial function $Q \times \Sigma \to Q$ which can be naturally extended to $Q \times \Sigma^\star \to Q$. Again, we write $r \xrightarrow{x} s$ instead of $\delta(r, x) = s$. We say that a PFA is *incomplete* if there is some undefined value of δ. A *sink state* in a PFA has a defined loop for each letter.

Definition 4.20. Let $w \in \{a, b\}^\star$. We say that a PFA $B = (Q, \{a, b\}, \delta)$ is a *sink device* for w, if there exists $q_0 \in Q$ such that:

1. $\delta(q_0, u) = q_0$ for each prefix u of w,
2. $\delta(s, w) = q_0$ for each $s \in Q$.

Note that the trivial automaton consisting of a single sink state is a sink device for any $w \in \{a, b\}^\star$. However, we are interested in strongly connected sink devices that are incomplete. In Lemma 4.21 we show how to prove the NP-completeness using a non-specific sink device in the general case of $w \in T_4$ and after that we construct explicit sink devices for a wide class of words from T_4.

Lemma 4.21. *Let $w \in T_4$ and assume that there exists a strongly connected incomplete sink device B for w. Then* $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}}$ *is NP-complete.*

Proof. We assume that w starts by a and write $w = a^\alpha b^\beta a u$ for $\alpha, \beta \geq 1$ and $u \in \{a, b\}^\star$. Denote $B = (Q_B, \{a, b\}, \delta_B)$. For a reduction from W-SAT, take an instance X, Φ with the notation used before, assuming that each $x \in X$ occurs in Φ. We construct a graph $\overline{G}_{w,\Phi} = \left(\overline{Q}, \overline{E}\right)$ as follows. Let $q_1 \in Q_B$ have an undefined outgoing transition, and let B' be an automaton obtained from B by arbitrarily defining all the undefined transitions except for one transition outgoing from q_1. Let $G_{B'}$ be the underlying graph of B'. By Theorem 4.16, $\mathrm{SRCW}^{\mathcal{Z}}_{2,\{w\}}$ is NP-complete, so it admits a reduction from W-SAT. Let $G_{w,\Phi} = (Q, E)$ be the graph obtained from such reduction, removing the loop on the sink state $q'_0 \in Q$. Let $s_1, \ldots, s_{|Q|-1}$ be an enumeration of all the states of $G_{w,\Phi}$ different from q'_0. Then we define $\overline{G}_{w,\Phi}$ as shown in Figure 4.12. We merge the state $q'_0 \in Q$ with the state $q_0 \in Q_B$, which is fixed by the definition of a sink device.

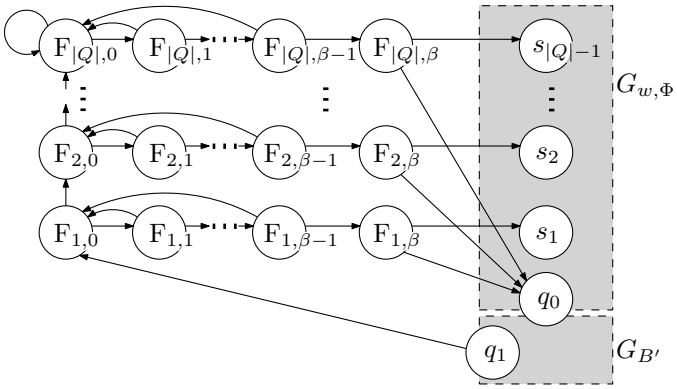


Figure 4.12: The graph $\overline{G}_{w,\Phi}$

First, let there be a coloring $\overline{\delta}$ of $\overline{G}_{w,\Phi}$ such that $\left|\overline{\delta}\left(\overline{Q}, w\right)\right| = 1$. It follows easily that $\overline{\delta}$, restricted to Q, encodes a coloring δ of $G_{w,\Phi}$ such that $|\delta(Q, w)| = 1$. The choice of $G_{w,\Phi}$ guarantees that there is a satisfying assignment ξ for Φ.
On the other hand, let ξ be a satisfying assignment of Φ. By the choice of $G_{w,\Phi}$, there is a coloring δ of $G_{w,\Phi}$ such that $|\delta(Q, w)| = 1$. We use the following coloring of $\overline{G}_{w,\Phi}$: The edges outgoing from $s_1, \ldots, s_{|Q|-1}$ are colored according to δ. The edges within $G_{B'}$ are colored according to B'. The edge $q_1 \longrightarrow \mathrm{F}_{1,0}$ is colored by b. All the other edges incoming to the states $\mathrm{F}_{1,0}, \ldots, \mathrm{F}_{|Q|,0}$, together with the edges of the form $\mathrm{F}_{i,\beta} \longrightarrow q_0$, are colored by a, while the remaining ones are colored by b.

□

For any $w \in \{a, b\}^\star$ we construct a strongly connected sink device $\mathbf{D}(w) = (Q_w, \{a, b\}, \delta_w)$. However, for some words $w \in T_4$ (e.g. for $w = abab$) the device $\mathbf{D}(w)$ is not incomplete and thus is not suitable for the reduction above. Take any $w \in \{a, b\}^\star$ and let $\mathfrak{C}^{\mathrm{P}}_w, \mathfrak{C}^{\mathrm{S}}_w, \mathfrak{C}^{\mathrm{F}}_w$ be the sets of all prefixes, suffixes and factors of w respectively, including the empty word ϵ. Let

$$Q_w \quad = \quad \left\{[u] \mid u \in \mathfrak{C}^{\mathrm{F}}_w,\, v \notin \mathfrak{C}^{\mathrm{S}}_w \text{ for each nonempty prefix } v \text{ of } u\right\},$$

while the partial transition function δ_w consists of the following transitions:

1. $[u] \xrightarrow{x} [ux]$ whenever $[u], [ux] \in Q_w$,
2. $[u] \xrightarrow{x} [\epsilon]$ whenever $ux \in \mathfrak{C}^{\mathrm{S}}_w$,
3. $[u] \xrightarrow{x} [\epsilon]$ whenever $[ux] \notin Q_w$, $ux \notin \mathfrak{C}^{\mathrm{S}}_w$, and $vx \in \mathfrak{C}^{\mathrm{P}}_w$ for a suffix v of u.

Lemma 4.22. *For any $w \in \{a, b\}^\star$, $\mathbf{D}(w)$ is a strongly connected sink device.*

Proof. First, we have to check that the definition of $\mathbf{D}(w) = (Q_w, \{a, b\}, \delta_w)$ yields a PFA, i.e. that no state has two outgoing transitions having the same label. Choose any $[u] \in Q_w$ and $x \in \{a, b\}$. It is enough to check that at most one of the three construction rules on Page 80 applies to $[u]$ and x. Indeed, by the definition of Q_w the claims $[ux] \in Q_w$ and $ux \in \mathfrak{C}^{\mathrm{S}}_w$ are contradictory. With this remark, it is obvious that any two of the rules have contradictory conditions.
Next, we show that $\mathbf{D}(w)$ is a sink device for w. We choose $q_0 = [\epsilon]$ and verify the two defining conditions of a sink device (see Definition 4.20):

1. Let $\preceq$ denote the partial order on $\mathfrak{C}^{\mathrm{S}}_w$ defined as $v_1 \preceq v_2$ if v_1 is a prefix of v_2. We start by verifying the claim for each $\preceq$-minimal nonempty $u \in \mathfrak{C}^{\mathrm{S}}_w$. Write $u = u_0 x$ for $x \in \{a, b\}$

and observe that $[u_0] \in Q_w$. Thus $[\epsilon] \xrightarrow{u_0} [u_0]$ and in the same time $[u_0] \xrightarrow{x} \epsilon$ due to the second construction rule.

Next, consider any $u \in \mathfrak{C}^{\mathrm{S}}_w$ and proceed by induction on the length of u. If $|u| = 0$, the claim is trivial. If $|u| > 0$, we consider two cases:

- If u is $\preceq$-minimal, we are done.
- If u is not $\preceq$-minimal, write $u = u_0 u_1$ where $u_0 \in \mathfrak{C}^{\mathrm{S}}_w$ satisfies $u_0 \preceq u$. Since both u_0 and u_1 lie in $\mathfrak{C}^{\mathrm{S}}_w$ and are shorter than u, we have $[\epsilon] \xrightarrow{u_0} [\epsilon]$ and $[\epsilon] \xrightarrow{u_1} [\epsilon]$ by the induction hypothesis.

2. For the case of $s = [\epsilon]$ it is enough to apply the first condition with $u = w$. So, choose any $s = [v_0] \in Q_w$ with $v_0 \neq \epsilon$. Find the longest prefix v_1 of w such that $[v_0 v_1] \in Q_w$. Observe that $[v_0] \xrightarrow{v_1} [v_0 v_1]$. Since $|v_0| \geq 1$, we have $v_1 \neq w$, thus we can denote $w = v_1 x v_2$ for $x \in \{a, b\}$. There is a transition $[v_0 v_1] \xrightarrow{x} [\epsilon]$ since:

   - If $v_0 v_1 x \in \mathfrak{C}^{\mathrm{S}}_w$, then the second construction rule applies easily.
   - Otherwise, the transition is defined by the third construction rule via $u = v_0 v_1$ and $v = v_1$. Indeed, from the choice of v_1 it follows that $[v_0 v_1 x] \notin Q_w$.

   We conclude by observing that $[\epsilon] \xrightarrow{v_2} [\epsilon]$ due to the first defining condition.

Finally, we have to check that $\mathbf{D}(w)$ is strongly connected. The first construction rule implies easily that each state is reachable from $[\epsilon]$. On the other hand, there is a path from any state to $[\epsilon]$ due to the second defining condition of a sink state.

□

Lemma 4.23. *Suppose that $w \in \{a, b\}^\star$ starts by x, where $\{x, y\} = \{a, b\}$. If there is $u \in \{a, b\}^\star$ satisfying all the following conditions, then $\mathbf{D}(w)$ is incomplete:*

1. *$[u] \in Q_w$,*
2. *$uy \notin \mathfrak{C}^{\mathrm{F}}_w$,*
3. *for each nonempty suffix v of uy, $v \notin \mathfrak{C}^{\mathrm{P}}_w$.*

Proof. We claim that there is no y-transition outgoing from the state $[u] \in Q_w$. Indeed, none of the three construction rules from Page 80 defines such transition:

1. The first rule does not apply since $uy \notin \mathfrak{C}^{\mathrm{F}}_w$ and thus $uy \notin Q_w$.
2. The second rule does not apply since $uy \notin \mathfrak{C}^{\mathrm{F}}_w$ and thus $uy \notin \mathfrak{C}^{\mathrm{S}}_w$.
3. The third rule is explicitly eliminated by the third condition above.

□

Theorem 4.24. *If a word $w \in T_4$ satisfies some of the following conditions, then $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}}$ is NP-complete:*

1. *w is of the form $w = x\overline{w}x$ for $\overline{w} \in \{a, b\}^\star, x \in \{a, b\}$,*
2. *w is of the form $w = x\overline{w}y$ for $\overline{w} \in \{a, b\}^\star, x, y \in \{a, b\}, x \neq y$, and $x^k y^l x \in \mathfrak{C}^{\mathrm{F}}_w$, $x^{k+1} \notin \mathfrak{C}^{\mathrm{F}}_w$, $y^{l+1} \notin \mathfrak{C}^{\mathrm{F}}_w$ for some $k, l \geq 1$.*

Proof. Due to Lemmas 4.21 and 4.22, it is enough to show that $\mathbf{D}(w)$ is incomplete. Let $m \geq 1$ be the largest integer such that y^m is a factor of w. It is straightforward to check that $u = y^m$ (in the first case) or $u = x^k y^l$ (in the second case) satisfies the three conditions from Lemma 4.23. □

4.2.3 W-SAT Is NP-Complete

In this section we prove Lemma 4.15, which claims that the following problem is NP-complete:

W-SAT	
Input:	Finite set X of *variables*, finite set $\Phi \subseteq X^4$ of *clauses*.
Output:	Is there an assignment $\xi : X \to \{\mathbf{0}, \mathbf{1}\}$ such that for each clause $(z_1, z_2, z_3, z_4) \in \Phi$ it holds that: (1) $\xi(z_i) = \mathbf{1}$ for some i, (2) $\xi(z_i) = \mathbf{0}$ for some $i \in \{1, 2\}$, (3) $\xi(z_i) = \mathbf{0}$ for some $i \in \{3, 4\}$?

According to Schaefer [84] we use the following formalism. Let S be a finite set of Boolean functions (or equivalently Boolean relations), i.e. let

$$S = \{R_1, \ldots, R_k\}$$

where

$$R_i : \{\mathbf{0}, \mathbf{1}\}^{\alpha_i} \to \{\mathbf{0}, \mathbf{1}\}$$

for each $i = 1, \ldots, k$. Note that α_i is the arity of the function R_i. Having such S, we define the following computational task:

SAT(S)	
Input:	Number $n \in \mathbb{N}$ of variables. Finite list of *clauses*, i.e. strings of the form $R_i(x_{j_1}, \ldots, x_{j_\alpha})$ where $i \in \{1, \ldots, k\}$, $\alpha = \alpha_i$, and $j_1, \ldots, j_\alpha \in \{1, \ldots, n\}$.
Output:	Is there an assignment of $x_1, \ldots, x_n$ such that all the clauses are satisfied?

Note that the clauses cannot contain negated literals. If there is a need for expressing negation, it requires an appropriate function R_i. For example 3-SAT is the same as SAT(S) where S contains four ternary functions, each for a possible number of negations (from 0 to 3).
However, we will use a one-element set

$$S_\diamondsuit = \{R_\diamondsuit\}$$

where

$$R_\diamondsuit(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_3 \vee \neg x_4)\,.$$

So the function $R_\diamondsuit$ gives $\mathbf{1}$ if and only if some input bit is $\mathbf{1}$ and there is some $\mathbf{0}$ in each half of the input. Observe that SAT($S_\diamondsuit$) is just an alternative formulation of W-SAT.

Definition 4.25. A function $R : \{\mathbf{0}, \mathbf{1}\}^\alpha \to \{\mathbf{0}, \mathbf{1}\}$ is

- *weakly negative* if it is equivalent to a Horn formula,
- *weakly positive* if it is equivalent to a dual-Horn formula (i.e. conjunction of disjunctions having at most one negated literal),
- *affine* if it is equivalent to a system of linear equations over the two-element field,
- *bijunctive* if it is equivalent to a 2-CNF (at most two literals in each conjunct).

Theorem 4.26 (Schaefer Dichotomy Theorem)**.** *Let S be of the above form. If S satisfies one of the conditions below, then* $\mathrm{SAT}(S)$ *is polynomial-time decidable. Otherwise,* $\mathrm{SAT}(S)$ *is NP-complete.*

1. *Each non-constant R_i has $R_i(\mathbf{1},\dots,\mathbf{1}) = \mathbf{1}$.*
2. *Each non-constant R_i has $R_i(\mathbf{0},\dots,\mathbf{0}) = \mathbf{1}$.*
3. *Each function R_i is weakly negative.*
4. *Each function R_i is weakly positive.*
5. *Each function R_i is affine.*
6. *Each function R_i is bijunctive.*

Definition 4.27. Let $R(x) = \mathbf{1}$ for $x \in \{\mathbf{0},\mathbf{1}\}^\alpha$. Then $C \subseteq \{1,\dots,n\}$ is a *change set* for R, x if $R(x \oplus e_C) = \mathbf{1}$, where $(e_C)_i = \mathbf{1}$ exactly for $i \in C$.

Theorem 4.28 ([84])**.** *A function $R : \{\mathbf{0},\mathbf{1}\}^\alpha \to \{\mathbf{0},\mathbf{1}\}$ is*

- *Affine if and only if for each $x, y, z \in \{\mathbf{0},\mathbf{1}\}^\alpha$ it holds that*

$$\text{if } R(x) = \mathbf{1},\ R(y) = \mathbf{1},\ \text{and } R(z) = \mathbf{1},\ \text{then } R(x \oplus y \oplus z) = \mathbf{1}.$$

- *Bijunctive if and only if for any x with $R(x) = \mathbf{1}$ and any change sets C_1, C_2 for R, x it holds that also $C_1 \cap C_2$ is a change set for R, x.*

Corollary 4.29. $\mathrm{SAT}(S_\diamondsuit)$ *is NP-complete.*

Proof. We just need to show that no of the conditions from Schaefer Dichotomy Theorem (Theorem 4.26) hold for $S_\diamondsuit$. Indeed:

1. The function $R_\diamondsuit$ is non-constant and we have $R_\diamondsuit(\mathbf{1},\dots,\mathbf{1}) = \mathbf{0}$.
2. The function $R_\diamondsuit$ is non-constant and we have $R_\diamondsuit(\mathbf{0},\dots,\mathbf{0}) = \mathbf{0}$.
3. Any Horn formula is either satisfied by $\mathbf{0},\dots,\mathbf{0}$ or contains a one-element clause x_j. The first case does not hold for $R_\diamondsuit$ and the second case implies that $x_j = \mathbf{1}$ in any assignment satisfying $R_\diamondsuit$, which does not hold for any x_j.
4. Any dual-Horn formula is either satisfied by $\mathbf{1},\dots,\mathbf{1}$ or contains an one-element clause $\neg x_j$. The first case does not hold for $R_\diamondsuit$ and the second case implies that $x_j = \mathbf{0}$ in any assignment satisfying $R_\diamondsuit$, which does not hold for any x_j.
5. It is enough to apply Theorem 4.28 to the following assignments:

$$\begin{aligned} x &= (\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0}), \\ y &= (\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0}), \\ z &= (\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1}), \\ x \oplus y \oplus z &= (\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{1}). \end{aligned}$$

6. It is enough to apply Theorem 4.28 to $x = (\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0})$, $C_1 = \{1,2\}$, and $C_2 = \{1,3\}$ since

$$\begin{aligned} x \oplus e_{C_1} &= (\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0}), \\ x \oplus e_{C_2} &= (\mathbf{0},\mathbf{0},\mathbf{1},\mathbf{0}), \\ x \oplus e_{C_1 \cap C_2} &= (\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}). \end{aligned}$$

□

Chapter 5

Jumping Finite Automata and Contextual Deleting

This chapter presents results about languages accepted by *jumping finite automata* and *clearing restarting automata*. Section 5.1 gives basic definitions and relates the two key models, which have been introduced recently, with notions considered in more classical literature.
In Section 5.2 we complete the initial study of jumping finite automata, which was started in a former article of Meduna and Zemek [64, 65]. The open questions about basic closure properties are solved. Besides of that, we correct erroneous results presented in [64, 65]. Finally, we point out important relations between jumping finite automata and other models studied in the literature. An article presenting these results was submitted to a journal.
In Section 5.3, we construct a clearing restarting automaton with two-letter contexts that accepts a language over a two-letter alphabet lying outside the class CFL, thus closing the study raised by Černo and Mráz, 2010 [25].

5.1 Models and their Relations

More specifically, our results and remarks deal with the following models:

1. *Jumping finite automata*, described recently by Meduna [64, 65]. They are equivalent to graph-controlled insertion systems with empty contexts.

2. *Clearing restarting automata*, which is a subclass of *Context rewriting systems*, both introduced in [25]. They are closely related to *semicontextual grammars*, as studied e.g. in [55, 75].

3. *Insertion systems*, as a special type of *insertion-deletion systems*, both introduced in [76] and widely studied in the last years. We also consider *graph-controlled* insertion systems, as described in [2, 98]. Insertion systems (possibly graph controlled) are equivalent to *semicontextual grammars without appearance checking* (possibly with *regular control*) introduced in [55, 75].

5.1.1 Preliminary Definitions

In this chapter we heavily use the natural notion of sequential insertion, as it was described e.g. in [44] and [47]:

Definition 5.1. Let $K, L \subseteq \Sigma^\star$ be languages. The *insertion* of K to L is

$$L \leftarrow K = \{u_1 v u_2 \mid u_1 u_2 \in L, v \in K\}\,.$$

More generally, for each $k \geq 1$ we denote

$$\begin{aligned} L \leftarrow^{k} K &= \left(L \leftarrow^{k-1} K\right) \leftarrow K, \\ L \leftarrow^{\star} K &= \bigcup_{i \geq 0} L \leftarrow^{i} K, \end{aligned}$$

where $L \leftarrow^{0} K$ stands for L. In expressions with $\leftarrow$ and $\leftarrow^{\star}$, a singleton set $\{w\}$ may be replaced by w. A chain $L_1 \leftarrow L_2 \leftarrow \cdots \leftarrow L_d$ of insertions is evaluated from the left, e.g. $L_1 \leftarrow L_2 \leftarrow L_3$ means $(L_1 \leftarrow L_2) \leftarrow L_3$. According to [32], $L \subseteq \Sigma^{\star}$ is a *unitary language* if $L = w \leftarrow^{\star} K$ for $w \in \Sigma^{\star}$ and finite $K \subseteq \Sigma$.

Definition 5.2. For languages $K, L \subseteq \Sigma^{\star}$, a word $w \in \Sigma^{\star}$ belongs to shuffle(K, L) if and only if there are words $u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k \in \Sigma^{\star}$ such that $u_1 u_2 \ldots u_k \in K$, $v_1 v_2 \ldots v_k \in L$, and $w = u_1 v_1 u_2 v_2 \ldots u_k v_k$. Furthermore, we denote $L^{\mathrm{R}} = \left\{w^{\mathrm{R}} \mid w \in L\right\}$, where w^{R} is the reversal of w.

Definition 5.3. For $w \in \Sigma^{\star}$ and $k, l \in \{1, \ldots, |w|\}$, the term $w[k]$ denotes the k-th letter of w and the term $w[k..l]$ denotes $w[k]\, w[k+1] \ldots w[l]$. We write $w[..k]$ and $w[k..]$ instead of $w[1..k]$ and $w[k..\,|w|]$.

5.1.2 Insertion-Deletion Systems

An *insertion-deletion system*, as introduced in [76], is described by a tuple $\mathbf{I} = (\Gamma, \Sigma, A, I, D)$ specifying a finite alphabet Γ, a set $\Sigma \subseteq \Gamma$ of *terminals*, a finite set $A \subseteq \Gamma^{\star}$ of *axioms*, and finite sets I, D of *insertion rules* and *deletion rules* from $\Gamma^{\star} \times \Gamma^{\star} \times \Gamma^{\star}$. We write $u_1 u_2 \Rightarrow u_1 v u_2$ if $(u_{\mathrm{L}}, v, u_{\mathrm{R}}) \in I$, where u_{L} is a suffix of u_1 and u_{R} is a prefix of u_2. Similarly, $u_1 v u_2 \Rightarrow u_1 u_2$ if $(u_{\mathrm{L}}, v, u_{\mathrm{R}}) \in D$, where u_{L} is a suffix of u_1 and u_{R} is a prefix of u_2. The system accepts the language

$$L = \{w \in \Sigma^{\star} \mid u \Rightarrow^{\star} w, u \in A\},$$

where $\Rightarrow^{\star}$ is the reflexive-transitive closure of $\Rightarrow$. If $D = \emptyset$, the system is an *insertion system*. In insertion systems we can assume that $\Gamma = \Sigma$. For each $m, m', n \geq 0$, an insertion system has *size* (n, m, m') if each insertion rule $(u_{\mathrm{L}}, v, u_{\mathrm{R}}) \in I$ satisfies $|u_{\mathrm{L}}| \leq m, |u_{\mathrm{R}}| \leq m'$. The class $\mathrm{ins}_n^{m,m'}$ contains languages accepted by insertion systems of the corresponding size. If some of n, m, m' is replaced by $*$, the parameter is not bounded. For example, $\mathrm{ins}_*^{0,0}$ contains exactly finite unions of unitary languages (one unitary language for each axiom).

In [2] and [98], the authors introduce *graph-controlled insertion systems*. Informally, such system may be defined by an insertion system $S = (\Gamma, \Sigma, A, I, D)$, a set Q of *components*, a set R of *rules* from $Q \times I \times Q$, an *initial component* $q_0 \in Q$ and a *final component* $q_{\mathrm{f}} \in Q$. We write $(s, u_1 u_2) \Rrightarrow (r, u_1 v u_2)$ if $(u_{\mathrm{L}}, v, u_{\mathrm{R}}) \in I$, where u_{L} is a suffix of u_1 and u_{R} is a prefix of u_2, and $(s, v, r) \in R$. The system accepts the language

$$L = \{w \in \Sigma^{\star} \mid (q_0, u) \Rrightarrow^{\star} (q_{\mathrm{f}}, w), u \in A\},$$

where $\Rrightarrow^{\star}$ is the reflexive-transitive closure of the relation $\Rrightarrow$ over $Q \times \Sigma^{\star}$. According to [2], the term $\mathrm{LStP}_k\left(\mathrm{ins}_n^{m,m'}\right)$ denotes the class of languages accepted by graph-controlled insertion systems with at most k components where the properties of each insertion rule are bounded by n, m, m' as above.

Insertion-deletion systems have been widely studied since the beginning of 21th century, see e.g. references of [2] and [98].

5.1.3 Jumping Finite Automata

In 2012, Meduna and Zemek [64] introduced *general jumping finite automata* as a model of discontinuous information processing. A general jumping finite automaton (GJFA) is described by a finite set Q of states, a finite alphabet Σ, a finite set R of *rules* from $Q \times \Sigma^\star \times Q$, an initial state $q_0 \in Q$, and a set $F \subseteq Q$ of final states. In a step of computation, the automaton switches from a state r to a state s using a rule $(r, v, s) \in R$, and deletes a factor equal to v from any part of the input word. The choices of the rule used and of the factor deleted are made nondeterministically. A word is accepted if there is a computation resulting in the empty word.

There is an infinite hierarchy of GJFA according to the maximum length of factor deleted in a single step - a GJFA is of *degree* n if $|v| \leq n$ for each $(r, v, s) \in R$. A GJFA of degree 1 is called a jumping finite automaton (JFA). Bold symbols **JFA** and **GJFA** denote the classes of languages accepted by these types of automata.

As described above, a GJFA is a quintuple $M = (Q, \Sigma, R, q_0, F)$. The following formal description of computation performed by a GJFA was introduced in [64].

Definition 5.4. Any string from the language $\Sigma^\star Q \Sigma^\star$ is called a *configuration* of M. For $r, s \in Q$ and $u_1, u_2, u_1', u_2', v \subseteq \Sigma^\star$, we write

$$u_1 r v u_2 \curvearrowright_M u_1' s u_2'$$

if $u_1 u_2 = u_1' u_2'$ and $(r, v, s) \in R$. By $\curvearrowright_M^\star$ we denote the reflexive-transitive closure of the binary relation $\curvearrowright_M$ over configurations. Finally,

$$L(M) = \{uv \mid u, v \in \Sigma^\star, f \in F, u q_0 v \curvearrowright_M^\star f\}$$

is the language accepted by M. If M is fixed, we write just $\curvearrowright$ and $\curvearrowright^\star$.

The placement of the state symbol s in a configuration $u_1 s u_2$ marks the position of an imaginary tape head. Note that this information is redundant - the head is allowed to move anywhere in each step.

Next, we give two simple lemmas that imply the membership in **GJFA** for each language that can be described using finite languages and insertions.

Lemma 5.5. *Each finite language $L \subseteq \Sigma^\star$ lies in* **GJFA**.

Proof. The language L is accepted by the two-state GJFA M with

$$\begin{aligned} M &= \{\{q_0, q_1\}, \Sigma, R, q_0, \{q_1\}\}, \\ R &= \{(q_0, w, q_1) \mid w \in S\}, \end{aligned}$$

which accepts if and only if it can delete the whole input in a single step. □

Lemma 5.6. *Let $L, K \subseteq \Sigma^\star$ lie in* **GJFA**. *Then $L \leftarrow K$ and $L \leftarrow^\star K$ lie in* **GJFA**.

Proof. Let $M_L = (Q_L, \Sigma, R_L, q_{0,L}, F_L)$ and $M_K = (Q_K, \Sigma, R_K, q_{0,K}, F_K)$ be GJFA recognizing K and L respectively, assuming $Q_L \cap Q_K = \emptyset$. To obtain M with $L(M) = L \leftarrow K$, we put

$$\begin{aligned} M &= (Q, \Sigma, R, q_{0,K}, F_L), \\ Q &= Q_L \cup Q_K, \\ R &= R_L \cup R_K \cup \{(f, \epsilon, q_{0,L}) \mid f \in F_K\}. \end{aligned}$$

To obtain M' with $L(M') = L \leftarrow^\star K$, we put

$$\begin{aligned} M' &= (Q, \Sigma, R', q_{0,K}, F_L)\,, \\ R' &= R \cup \{(f, \epsilon, q_{0,K}) \mid f \in F_K\}\,. \end{aligned}$$

□

Let us give a few examples of GJFA languages that are used later in this paper and follow easily from the above lemmas. Note that a GJFA over an alphabet Σ can be seen as operating over any alphabet $\Sigma' \supseteq \Sigma$. The symbol ϵ stands for the empty word.

Example 5.7. The following languages lie in **GJFA**:

1. The trivial language $\Sigma^\star = \epsilon \leftarrow^\star \Sigma$ over an arbitrary Σ.
2. The language $\Sigma^\star u \Sigma^\star = \Sigma^\star \leftarrow u$ for $u \in \Sigma^\star$ over an arbitrary Σ.
3. The Dyck language D over $\Sigma = \{a, \overline{a}\}$. We have $D = \epsilon \leftarrow^\star a\overline{a}$.
4. Any semi-Dyck language D_k over $\Sigma = \{a_1, \ldots, a_k, \overline{a}_1, \ldots, \overline{a}_k\}$. We have $D_k \leftarrow^\star \{a_1\overline{a}_1, \ldots, a_k\overline{a}_k\}$.
5. Any unitary language.

However, there are GJFA languages that cannot be obtained from finite languages by applying Lemma 5.6, such as the following classical language that is not context-free and lies even in **JFA**. By $|w|_x$ we denote the number of occurrences of a letter $x \in \Sigma$ in a word $w \in \Sigma^\star$.

Example 5.8. The JFA M with

$$\begin{aligned} M &= (\{q_0, q_1, q_2\}, \Sigma, R, q_0, \{q_0\})\,, \\ R &= \{(q_0, a, q_1), (q_1, b, q_2), (q_2, c, q_0)\} \end{aligned}$$

accepts the language $L = \{w \in \Sigma^\star \mid |w|_a = |w|_b = |w|_c\}$ over $\Sigma = \{a, b, c\}^\star$.

We have shown that the class **GJFA** is not a subclass of context-free languages, but it was pointed out in [64] that each GJFA language is context-sensitive. The class **GJFA** does not stick to classical measures of expressive power - in the next section we give examples of regular languages that do not lie in **GJFA**. As for JFA languages, in [65] the authors show that a language lies in **JFA** if and only if it is equal to the permutation closure of a regular language. Next, we fix additional notation that turns out to be very useful in our proofs. The notions of *paths* and *labels* naturally correspond to graphical representations of GJFA, where vertices stand for states and labeled directed edges stand for rules.

Definition 5.9. A *path* from $s_0 \in Q$ to $s_d \in Q$ in a GJFA $M = (Q, \Sigma, R, q_0, F)$ is a sequence

$$(s_0, v_1, s_1), (s_1, v_2, s_2), \ldots, (s_{d-1}, v_d, s_d)$$

of rules from R. The path is *accepting* if $s_0 = q_0$ and $s_d \in F$. The *labeling* of the path is the sequence $v_1, v_2, \ldots, v_d$ of words from $\Sigma^\star$. The *total label* of the path is the word $v_1 v_2 \ldots v_d \in \Sigma^\star$. An empty path from any state to itself has total label ϵ.

Lemma 5.10. *Let $M = (Q, \Sigma, R, q_0, F)$ be a GJFA and $w \in \Sigma^\star$. Then $w \in L(M)$ if and only if*

$$w \in \epsilon \leftarrow v_d \leftarrow v_{d-1} \leftarrow \cdots \leftarrow v_2 \leftarrow v_1,$$

where $v_1, v_2, \ldots, v_d$ is a labeling of an accepting path in M.

The above lemma suggests a generative approach to GJFA - the computation of a GJFA may be equivalently described in terms of inserting factors instead of deleting them. A word is accepted by a GJFA if and only if it can be composed by inserting factors to the empty word according to labels of a reversed accepting path.
Let us point out that the class **GJFA** can be put into the frameworks of graph-controlled insertion-deletion systems. The following theorem is actually an easy observation.

Theorem 5.11. *It holds that* $\mathbf{GJFA} = \mathrm{LStP}_*\left(\mathrm{ins}_*^{0,0}\right)$.

Indeed, with the generative approach to GJFA in mind, a GJFA may be transformed to a graph-controlled insertion system with the same structure, using only the axiom ϵ and insertion rules with empty contexts. For the backward inclusion we just encode the axioms to rules specifying that the computation ends by deleting an axiom.
Finally, an interesting result follows from the fact that each unitary language lies in **GJFA**. According to the main result of the Haussler's article [44],∪ there is an alphabet Σ such that for given finite sets $S, T \subseteq \Sigma^\star$ it is undecidable whether the intersection of $\epsilon \leftarrow^\star S$ and $\epsilon \leftarrow^\star T$ contains a non-empty string. It is trivial to construct a GJFA accepting $(\epsilon \leftarrow^\star S) \setminus \{\epsilon\}$, so we obtain the following theorem.

Theorem 5.12. *Given GJFA* M_1, M_2*, it is undecidable whether* $L(M_1) \cap L(M_2) = \emptyset$.

5.1.4 Clearing Restarting Automata

In [25], Černo and Mráz introduced the following models of linguistical analysis of natural language sentences.

Definition 5.13. For $k \geq 0$, a *k-context rewriting system* is a tuple $R = (\Sigma, \Gamma, I)$, where Σ is an input alphabet, $\Gamma \supseteq \Sigma$ is a working alphabet not containing the special symbols ¢ and \$, called *sentinels*, and I is a finite set of instructions of the form

$$(u_\mathrm{L}, v \to t, u_\mathrm{R}),$$

where u_L is a *left context*, $x \in \Gamma^k \cup ¢\Gamma^{k-1}$, y is a *right context*, $y \in \Gamma^k \cup \Gamma^{k-1}\$$, and $v \to t$ is a *rule*, $z, t \in \Gamma^\star$. A word $w = u_1 v u_2$ can be rewritten into $u_1 t u_2$ (denoted as$u_1 v u_2 \to_R u_1 t u_2$) if and only if there exists an instruction $(u_\mathrm{L}, v \to t, u_\mathrm{R}) \in I$ such that u_L is a suffix of $¢u_1$ and u_R is a prefix of $u_2\$$. The symbol $\to_R^\star$ denotes the reflexive-transitive closure of $\to_R$.

Definition 5.14. For $k \geq 0$, a *k-clearing restarting automaton* (k-cl-RA) is a system $M = (\Sigma, I)$, where (Σ, Σ, I) is a k-context rewriting system such that for each $\mathbf{i} = (u_\mathrm{L}, v \to t, u_\mathrm{R}) \in I$ it holds that $v \in \Sigma^+$ and $t = \epsilon$. Since t is always the empty word, we use the notation $\mathbf{i} = (u_\mathrm{L}, v, u_\mathrm{R})$. A k-cl-RA M accepts the language

$$L(M) = \{w \in \Sigma^\star \mid w \vdash_M^\star \epsilon\},$$

where $\vdash_M$ denotes the rewriting relation $\to_{\overline{M}}$ of $\overline{M} = (\Sigma, \Sigma, I)$. The term $\mathcal{L}(k\text{-cl-RA})$ denotes the class of languages accepted by k-cl-RA.

Like in jumping finite automata, one may consider the generative approach to languages accepted by clearing restarting automata. In this case, the generative approach is formalized by writing $w_2 \dashv w_1$ instead of $w_1 \vdash w_2$.
The notions of k-cl-RA and insertion systems of size $(*, k, k)$ are very similar since the generative approach makes it possible to consider insertions instead of deletions in a k-cl-RA. The differences between the models are:

1. Insertion systems use finite sets of atoms, while k-cl-RA use only ϵ as an atom.

2. k-cl-RA use the markers ¢ and \$ in contexts.

The first point does not make a real difference:

Theorem 5.15. *It holds that* $\mathrm{INS}_*^{k,k} \subseteq k$-cl-RA.

Proof. Let $\mathbf{I} = (\Sigma, \Sigma, A, I, \emptyset)$ be an insertion system. The language of $\mathbf{I}$ is accepted by the k-cl-RA $R = (\Sigma, I \cup I_A)$, where $I_A = \{(¢, v, \$) \mid v \in A\}$. □

The following observation says that the second point may be overcame by explicit endmarking:

Theorem 5.16. *If* $L \in k$-cl-RA, *then* $¢L\$ \in \mathrm{INS}_*^{k,k}$.

Proof. Let $R = (\Sigma, I)$ be a k-cl-RA. The language of R is accepted by the insertion system $I = (\Sigma, \Sigma, \{\epsilon\}, I, \emptyset)$. □

5.2 Closure Properties of the Class GJFA

The present section contains the following contributions:

1. We correct erroneous claims from [64] and [65] about closure properties of the class **GJFA** - if fact it is not closed under homomorphism nor under inverse homomorphism.

2. We answer the open questions about closure properties of **GJFA** formulated in the two publications. Specifically, we disprove the closure under shuffle, Kleene star and Kleene plus, and prove the closure under reversal.

	GJFA	**JFA**
Endmarking	−	−
Concatenation	−	−
Shuffle	− ♢	+
Union	+	+
Complement	−	+
Intersection	−	+
Int. with regular languages	−	−
Kleene star	− ♢	−
Kleene plus	− ♢	−
Reversal	+ ♢	+
Substitution	−	−
Regular substitution	−	−
Finite substitution	− ♦	−
Homomorphism	− ♦	−
ϵ-free homomorphism	− ♦	−
Inverse homomorphism	− ♦	+

5.2.1 A Necessary Condition for Membership in GJFA

In order to formulate our main tools for disproving membership in GJFA, the following technical notions remain to be defined.

Definition 5.17. A language $K \subseteq \Sigma^\star$ is a *composition* if it can be written as

$$K = \epsilon \leftarrow v_d \leftarrow v_{d-1} \leftarrow \cdots \leftarrow v_2 \leftarrow v_1,$$

where $v_1, \ldots, v_d \in \Sigma^\star$ and $d \geq 0$. A composition is of degree n if $|v_i| \leq n$ for each $i \in \{1, \ldots, d\}$. For each $n \geq 0$, let $\mathbf{UC}_n$ denote the class of languages L that can be written as

$$L = \bigcup_{K \in \mathcal{C}} K,$$

where $\mathcal{C}$ is any (possibly infinite) set of compositions of degree n. We also denote $\mathbf{UC} = \bigcup_{n \geq 0} \mathbf{UC}_n$.

Actually, the class $\mathbf{UC}_n$ for $n \geq 1$ consists of the languages accepted by infinite-state machines that work like GJFA of weight n. The corresponding languages may not be recursive. However, we use the membership in $\mathbf{UC}$ only as a technical necessary condition for membership in $\mathbf{GJFA}$.

Lemma 5.18. $\mathbf{GJFA} \subseteq \mathbf{UC}$.

Proof. Let $M = (Q, \Sigma, R, q_0, F)$ be a GJFA. Let $\mathcal{P}$ be the set of all accepting paths in M. According to Lemma 5.10, we have

$$L(M) = \bigcup_{\mathbf{p} \in \mathcal{P}} \left(\epsilon \leftarrow v_{\mathbf{p},d} \leftarrow v_{\mathbf{p},d-1} \leftarrow \cdots \leftarrow v_{\mathbf{p},2} \leftarrow v_{\mathbf{p},1}\right),$$

where $v_{\mathbf{p},1}, \ldots, v_{\mathbf{p},d}$ is the labeling of $\mathbf{p}$. □

The following lemma deals with the language $L = \{ab\}^\star$, which serves as a canonical non-GJFA language in the proofs of our main results.

Lemma 5.19. *The language $L = \{ab\}^\star$ does not lie in* GJFA.

Proof. Suppose for a contradiction that $L \in \mathbf{UC}_n$ with $n \geq 0$. Fix $w = (ab)^{n+1}$. According to the definition of $\mathbf{UC}_n$, w lies in a composition $K \subseteq L$ of the form

$$K = K' \leftarrow v$$

of degree n, denoting the last inserted word by v instead of v_1.Thus, $w = u_1 v u_2$ for $u_1 u_2 \in K'$. As $|v| \leq n$, at least one of the following assumptions is fulfilled:

1. Assume that $|u_1| \geq 2$ and write $u_1 = ab\overline{u}_1$. If v starts by a, we have $avb\overline{u}_1 \in K$. If v starts by b, we have $abv\overline{u}_1 \in K$.

2. Assume that $|u_2| \geq 2$ and write $u_2 = \overline{u}_2 ab$. If v starts by a, we have $\overline{u}_2 avb \in K$. If v starts by b, we have $\overline{u}_2 abv \in K$.

In each case, K contains a word having some of the factors aa and bb. Thus $K \nsubseteq L$, which is a contradiction.

□

5.2.2 The Main Results

The table below lists various unary and binary operators on languages. The symbols $+, -$ tell that a class is closed or is not closed under an operator, respectively. A similar table was presented in [64, 65], containing several questionmarks. In this section we complete and correct these results. The symbol ♢ marks answers to open questions and the symbol ⧫ marks corrections.
Before proving the new results, let us deal with the closure under intersection. The following theorem is stated also in [64, 65], but we find the presented proof insufficient.

Theorem 5.20. GJFA *is not closed under intersection.*

Proof. Let $\Sigma = \{a, \overline{a}\}$ and $L = L(M)$ for

$$\begin{aligned} M &= (\{q, r\}, \Sigma, R, q, \{r\}), \\ R &= \{(q, \overline{a}a, q), (q, a\overline{a}, r)\}, \end{aligned}$$

as depicted in Figure 5.1. For each $d \geq 1$ there is exactly one accepting path of length d in M. According to Lemma 5.10, we have

$$L = \bigcup_{d \geq 1} K_d,$$

where $K_1 = \epsilon \leftarrow a\overline{a}$ and $K_{i+1} = K_i \leftarrow \overline{a}a$ for $i \geq 1$. We show that

$$D \cap L = \{a\overline{a}\}^\star, \tag{5.1}$$

where $D \in \mathbf{GJFA}$ is the Dyck language from Example 5.7, and $\{a\overline{a}\}^\star$ does not lie in **GJFA** due to Lemma 5.19. The backward inclusion is easy. As for the forward one, it is enough to verify that $D \cap K_d \subseteq \{a\overline{a}\}^\star$ for each $d \geq 1$. The case $d = 1$ is trivial since $K_1 = \{a\overline{a}\}$. In order to continue inductively, fix $d \geq 2$. For any $w \in D \cap K_d$, we have $w = u_1\overline{a}au_2$ for $u_1u_2 \in K_{d-1}$. From $D = \epsilon \leftarrow^\star a\overline{a}$ it follows that $u_1 \in DaD$, $u_2 \in D\overline{a}D$, and thus, $u_1u_2 \in D$. By the induction assumption, $u_1u_2 \in \{a\overline{a}\}^\star$. Hence $w \in \{a\overline{a}\}^\star (\overline{a}a) \{a\overline{a}\}^\star$ or $w \in \{a\overline{a}\}^\star a (\overline{a}a) \overline{a} \{a\overline{a}\}^\star$. The first case implies $w \notin D$, which is a contradiction, and the second case implies $w \in \{a\overline{a}\}^\star$. □

The next theorem shows that some of the announced results actually follow very easily from Lemma 5.19, which claims that $\{ab\}^\star \notin \mathbf{GJFA}$. Theorems 5.22 and 5.23 provide special counter-examples for the closure under inverse homomorphism and shuffle.

Theorem 5.21. GJFA *is not closed under:*

1. Kleene star,
2. Kleene plus,
3. ϵ-free homomorphism,
4. homomorphism,
5. finite substitution.

Proof. We have $\{ab\} \in \mathbf{GJFA}$ and $\{ab\}^\star \notin \mathbf{GJFA}$ due to Lemma 5.19. As **GJFA** is closed under union, $\{ab\}^+ \notin \mathbf{GJFA}$ as well. As for ϵ-free homomorphism, consider $\varphi : \{a\}^\star \to \{a, b\}^\star$ with $\varphi(a) = ab$. We have $L = \{a\}^\star \in \mathbf{GJFA}$ and $\varphi(L) = \{ab\}^\star \notin \mathbf{GJFA}$. Trivially, φ is also a general homomorphism and a finite substitution. □

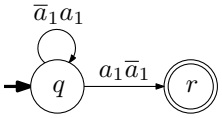


Figure 5.1: The GJFA M with $D \cap L(M) = \{a\overline{a}\}^\star$

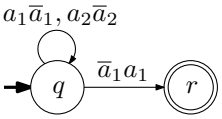


Figure 5.2: The GJFA M with $L(M) = D_2\overline{a}_1D_2a_1D_2$

Theorem 5.22. GJFA *is not closed under inverse homomorphism.*

Proof. Let $\Sigma = \{a_1, \overline{a}_1, a_2, \overline{a}_2\}$ and

$$\begin{aligned} M &= (\{q, r\}, \Sigma, R, q, \{r\}), \\ R &= \{(q, a_1\overline{a}_1, q), (q, a_2\overline{a}_2, q), (q, \overline{a}_1 a_1 r)\}, \end{aligned}$$

see Figure 5.2. Let $L = L(M)$. Observe that $L = D_2\overline{a}_1 D_2 a_1 D_2$, where D_2 is the semi-Dyck language with two types of brackets: $a_1, \overline{a}_1$ and $a_2, \overline{a}_2$. According to Example 5.7, $D_2 \in \mathbf{GJFA}$. Let $\varphi : \{a, b\}^\star \to \Sigma^\star$ be defined as

$$\begin{aligned} \varphi(a) &= \overline{a}_1 a_2, \\ \varphi(b) &= \overline{a}_2 a_1. \end{aligned}$$

We claim that $\varphi^{-1}(L) = \{ab\}^\star$, which means

$$L \cap \mathrm{rng}(\varphi) = \{\overline{a}_1 a_2 \overline{a}_2 a_1\}^\star.$$

The backward inclusion is easy - we have $\{\overline{a}_1 a_2 \overline{a}_2 a_1\}^\star \subseteq \overline{a}_1 D_2 a_1$. As for the forward inclusion, take any $w \in L \cap \mathrm{rng}(\varphi)$ and fix $v = x_1 \dots x_m$ such that $\varphi(v) = w$ and $x_1, \dots, x_m \in \{a, b\}$. As $w \in \mathrm{rng}(\varphi)$, w starts by $\overline{a}_1$ or $\overline{a}_2$ and ends by a_1 or a_2. Thus $w \in \overline{a}_1 D_2 a_1$, $x_1 = a$, $x_m = b$, and

$$w = \overline{a}_1 a_2 \varphi(x_2) \dots \varphi(x_{n-1})\, \overline{a}_2 a_1,$$

where

$$a_2 \varphi(x_2) \dots \varphi(x_{n-1})\, \overline{a}_2 \quad \in \quad D_2.$$

None of the factors $a_2\overline{a}_1$ and $a_1\overline{a}_2$ can occur in D_2. It follows that $x_2 = b$ and we continue by induction: for each $i = 2, \dots, m-2$ it holds that

$$x_i = a \Leftrightarrow x_{i+1} = b,$$

which implies $v \in \{ab\}^\star$ and $w \in \{\overline{a}_1 a_2 \overline{a}_2 a_1\}^\star$. □

Theorem 5.23. GJFA *is not closed under shuffle.*

Proof. Again, we fix $\Sigma = \{a_1, \overline{a}_1, a_2, \overline{a_2}\}$ and consider the semi-Dyck language $L = D_2 \in \mathbf{GJFA}$ over Σ. We claim that $\mathrm{shuffle}(D_2, D_2) \notin \mathbf{GJFA}$. According to Lemma 5.18 we assume for a contradiction that $\mathrm{shuffle}(D_2, D_2) \in \mathbf{UC}_n$ for $n \geq 1$. Denote $w = a_1^n a_2^n \overline{a}_1^n \overline{a}_2^n$. The word w lies in a composition K of degree n having the form $K = K' \leftarrow v$, so $w = u_1 v u_2$ for $u_1 u_2 \in \Sigma^\star$. Clearly, there is $x \in \{a_1, a_2\}$ such that at least one of the following assumptions is fulfilled:

1. Assume that v contains x. As $|v| \leq n$, it cannot contain $\overline{x}$. The word $u_1 u_2 v$ lies in K but it contains an occurrence of x with no occurrence of $\overline{x}$ on the right, so it does not lie in $\mathrm{shuffle}(D_2, D_2)$.

2. Assume that v contains $\overline{x}$. As $|v| \leq n$, it cannot contain x. The word $v u_1 u_2$ lies in K but it contains an occurrence of $\overline{x}$ with no occurrence x on the left, so it does not lie in $\mathrm{shuffle}(D_2, D_2)$.

□

Theorem 5.24. GJFA *is closed under reversal.*

Proof. Let $M = (Q, \Sigma, R, q_0, F)$ be a GJFA. We claim that the automaton

$$\begin{aligned} M^{\mathrm{R}} &= \left(Q, \Sigma, R^{\mathrm{R}}, q_0, F\right), \\ R^{\mathrm{R}} &= \left\{\left(q, v^{\mathrm{R}}, r\right) \mid (q, v, r) \in R\right\} \end{aligned}$$

accepts $L(M)^{\mathrm{R}}$. Due to symmetry, it is enough to prove $L(M)^{\mathrm{R}} \subseteq L\left(M^{\mathrm{R}}\right)$. According to Lemma 5.10, we just observe that

$$\begin{aligned} \left(\epsilon \leftarrow v_d \leftarrow v_{d-1} \leftarrow \cdots \leftarrow v_2 \leftarrow v_1\right)^{\mathrm{R}} &\subseteq \epsilon \leftarrow v_d^{\mathrm{R}} \leftarrow v_{d-1}^{\mathrm{R}} \leftarrow \cdots \leftarrow v_2^{\mathrm{R}} \leftarrow v_1^{\mathrm{R}} \\ &\subseteq L\left(M^{\mathrm{R}}\right) \end{aligned}$$

for each accepting path in M with labeling $v_1, \ldots, v_d$. □

5.3 Clearing Restarting Automata with Small Contexts

Though the basic model of clearing restarting automata is not able to describe all context-free languages nor to handle basic language operations (e.g. concatenation and union) [25], it has been deeply studied in order to design suitable generalizations. The study considered also restrictions of the maximum context length in rewriting rules:

Theorem 5.25 ([25])**.**

1. *For each $k \geq 3$, the class $\mathcal{L}(k\text{-cl-RA})$ contains a binary language, which is not context-free.*
2. *The class $\mathcal{L}(2\text{-cl-RA})$ contains a language $L \subseteq \Sigma^\star$ with $|\Sigma| = 6$, which is not context-free.*
3. *The class $\mathcal{L}(1\text{-cl-RA})$ contains only context-free languages.*

The present section is devoted to proving the following theorem, which completes the results listed above.

Theorem 5.26. *The class $\mathcal{L}(2\text{-cl-RA})$ contains a binary language, which is not context-free.*

In order to prove Theorem 5.26, we define two particular rewriting systems:

1. A 1-context rewriting system $R_{\mathrm{uV}} = (\{\mathrm{u}, \mathrm{V}\}, \{\mathrm{u}, \mathrm{V}\}, I_{\mathrm{uV}})$. The set I_{uV} is listed in Table 5.1.
2. A 2-clearing restarting automaton $R_{01} = (\{0, 1\}, I_{01})$. The set I_{uV} is listed in Table 5.2.

We write $\rightarrow_{\mathrm{uV}}$ for the rewriting relation of R_{uV} and $\dashv_{01}$ for the production relation of R_{01}.

0)	$(¢, \epsilon \rightarrow \mathrm{uu}, \$)$
1)	$(¢, \mathrm{u} \rightarrow \mathrm{uuV}, \epsilon)$
2)	$(\epsilon, \mathrm{Vu} \rightarrow \mathrm{uuuV}, \epsilon)$
3)	$(\epsilon, \mathrm{Vu} \rightarrow \mathrm{uuuu}, \$)$

Table 5.1: The rules I_{uV}

	(a)	(b)	c)	d)
0)	$(¢, 00, \$)$	-	-	-
1)	$(¢, 10, 00)$	$(¢, 00, 10)$	-	-
2)	$(01, 10, 00)$	$(00, 11, 01)$	$(11, 00, 10)$	$(10, 01, 11)$
3)	$(01, 10, 0\$)$	$(00, 11, 0\$)$	-	-

Table 5.2: The rules I_{01} sorted by types 0 to 3

The key feature of the system R_{uV} is:

Lemma 5.27. *Let $w \in L(R_{\mathrm{uV}}) \cap \{\mathrm{u}\}^\star$. Then $|w| = 2 \cdot 3^n$ for some $n \geq 0$.*

The proof is postponed to Section 5.3.1. We also define:

1. A length-preserving mapping $\varphi : \{0,1\}^\star \to \{\mathrm{u},\mathrm{V}\}^\star$ as $\varphi(x_1 \dots x_n) = \overline{x}_1 \dots \overline{x}_n$, where

$$\overline{x}_k = \begin{cases} \mathrm{V} & \text{if } 1 < k < n \text{ and } x_{k-1} = x_{k+1} \\ \mathrm{u} & \text{otherwise} \end{cases}$$

   for each $k \in \{1, \dots, n\}$.

2. A regular language $K \subseteq \{0,1\}^\star$:

$$K \quad = \quad \left\{ w \in \{0,1\}^\star \mid w \text{ has none of the factors } 000, 010, 101, 111 \right\}.$$

The following is a trivial property of φ and K:

Lemma 5.28. *Let $u \in \{0,1\}^\star$. Then $u \in K$ if and only if $\varphi(u) \in \{\mathrm{u}\}^\star$.*

The next lemma expresses how the systems R_{01} and R_{uV} are related:

Lemma 5.29. *Let $u, v \in \{0,1\}^\star$. If $u \dashv_{01} v$, then $\varphi(u) \to_{\mathrm{uV}} \varphi(v)$.*

Proof. For $u = v$ the claim is trivial, so we suppose $u \neq v$. Denote $m = |u|$. As u can be rewrote to v using a single rule of R_{01}, we can distinguish which of the four kinds of rules (the rows 0 to 3 of Table 5.2) is used:

0) If the rule 0 is used, we have $u = \epsilon$ and $v = 00$. Thus $\varphi(u) = \epsilon$ and $\varphi(v) = \mathrm{uu}$.

1) If a rule $(\dot{\varsigma}, z_1 z_2, y_1 y_2)$ of the kind 1 is used, we see that v has some of the prefixes $1000, 0010$ and so $\varphi(v)$ starts with uuV. Trivially, $\varphi(u)$ starts with u. Because $u[1..] = v[3..]$, we have $\varphi(u)[2..] = \varphi(v)[4..]$ and we conclude that applying the rule $(\dot{\varsigma}, \mathrm{u} \to \mathrm{uuV}, \epsilon)$ rewrites $\varphi(u)$ to $\varphi(v)$.

2) If a rule $(x_1 x_2, z_1 z_2, y_1 y_2)$ of the kind 2 is used, we have

$$\begin{aligned} u[k..k+3] &= x_1 x_2 y_1 y_2, \\ v[k..k+5] &= x_1 x_2 z_1 z_2 y_1 y_2. \end{aligned}$$

   for some $k \in \{1, \dots, m-3\}$. As $x_1 x_2 y_1 y_2$ equals some of the factors $0100, 0001, 1110, 1011$, we have

$$\varphi(u)[k+1..k+2] = \mathrm{Vu}.$$

   As $x_1 x_2 z_1 z_2 y_1 y_2$ equals some of the factors $011000, 001101, 110010, 100111$, we have

$$\varphi(v)[k+1..k+4] = \mathrm{uuuV}.$$

   Because $u[..k+1] = v[..k+1]$ and $u[k+2..] = v[k+4..]$, we have

$$\begin{aligned} \varphi(u)[..k] &= \varphi(v)[..k], \\ \varphi(u)[k+3..] &= \varphi(v)[k+5..]. \end{aligned}$$

   Now it is clear that the rule $(\epsilon, \mathrm{Vu} \to \mathrm{uuuV}, \epsilon)$ rewrites $\varphi(u)$ to $\varphi(v)$.

3) If a rule $(x_1 x_2, z_1 z_2, y\$)$ of the kind 3 is used, we have

$$\begin{aligned} u[m-2..m] &= x_1 x_2 y, \\ v[m-2..m+2] &= x_1 x_2 z_1 z_2 y. \end{aligned}$$

As x_1x_2y equals some of the factors $010, 000$, we have

$$\varphi(u)[m-1..m] \quad = \quad \mathrm{Vu}.$$

As $x_1x_2z_1z_2y$ equals some of the factors $01100, 00110$, we have

$$\varphi(v)[m-1..m+2] \quad = \quad \mathrm{uuuV}.$$

Because $u[..m-1] = v[..m-1]$, we have

$$\varphi(u)[..m-2] \quad = \quad \varphi(v)[..m-2]\,,$$

Now it is clear that the rule $(\epsilon, \mathrm{Vu} \to \mathrm{uuuu}, \$)$ rewrites $\varphi(u)$ to $\varphi(v)$.

□

Corollary 5.30. *If $u \in L(R_{01})$, then $\varphi(u) \in L(R_{\mathrm{uV}})$.*

Proof. Follows from the fact that $\varphi(\epsilon) = \epsilon$ and a trivial inductive use of Lemma 5.29. □

The last part of the proof of Theorem relies of the following lemma, whose proof is postponed to Section 5.3.1:

Lemma 5.31. *For each $\alpha, \beta > 0$ it holds that*

$$00\,(1100)^{\alpha}\,1000\,(1100)^{\beta} \dashv_{01}^{\star} 00\,(1100)^{\alpha+9}\,1000\,(1100)^{\beta-1}\,.$$

Corollary 5.32. *For each $\beta > 0$ it holds that*

$$001000\,(1100)^{\beta} \dashv_{01}^{\star} 00\,(1100)^{9\beta}\,1000.$$

Proof. As the left-hand side is equal to $00\,(1100)^{0}\,1000\,(1100)^{\beta}$ and the right-hand side is equal to $00\,(1100)^{9\beta}\,1000\,(1100)^{0}$, the claim follows from an easy inductive use of Lemma 5.31. □

Corollary 5.33. *The language $L(R_{01}) \cap K$ is infinite.*

Proof. We show that for each $k \geq 0$,

$$00\,(1100)^{\frac{2\cdot 9^k-2}{4}} \quad \in \quad L(R_{01})\,.$$

In the case $k = 0$ we just check that $00 \in L(R_{01})$. Next we suppose that the claim holds for a fixed $k \geq 0$ and show that

$$00\,(1100)^{\frac{2\cdot 9^k-2}{4}} \dashv_{01}^{\star} 00\,(1100)^{\frac{2\cdot 9^{k+1}-2}{4}}\,.$$

Using the rules 1a and 1b we get

$$00\,(1100)^{\frac{2\cdot 9^k-2}{4}} \dashv_{01}^{\star} 1000\,(1100)^{\frac{2\cdot 9^k-2}{4}} \dashv_{01}^{\star} 001000\,(1100)^{\frac{2\cdot 9^k-2}{4}}\,,$$

while Corollary 5.32 continues with

$$001000\,(1100)^{\frac{2\cdot 9^k-2}{4}} \dashv_{01}^{\star} 00\,(1100)^{\frac{2\cdot 9^{k+1}-18}{4}}\,1000.$$

Finally, denoting $p = 00\,(1100)^{\frac{2\cdot 9^{k+1}-18}{4}}$, using rules 2b, 2a, 2b, 2d, 2c, and 2a respectively we get

$$p1000 \dashv_{01} p100\underline{11}0 \dashv_{01} p1\underline{10}00110 \dashv_{01} p1100\underline{11}0110 \dashv_{01} p110011\underline{01}110 \dashv_{01}$$
$$\dashv_{01} p11001100\underline{10}010 \dashv_{01} p1100110011001\underline{10}0 = 00\,(1100)^{\frac{2\cdot 9^{k+1}-2}{4}}\,.$$

□

We conclude the proof of Theorem 2.8 by pointing out that Lemmas 5.28, 5.29, and 5.27 say that for each $w \in \{0,1\}^\star$ we have

$$\begin{aligned} w \in L(R_{01}) \cap K &\Rightarrow \varphi(w) \in L(R_{\mathrm{uV}}) \cap \{\mathrm{u}\}^\star \\ &\Rightarrow (\exists n \geq 0)\ |w| = 2 \cdot 3^n \end{aligned}$$

This, together with the pumping lemma for context-free languages and the infiniteness of $L(R_{01}) \cap K$, implies that $L(R_{01}) \cap K$ is not a context-free language. As the class of context-free languages is closed under intersections with regular languages, nor $L(R_{01})$ is context-free.

5.3.1 Proofs of Lemmas 5.27 and 5.31

Proof of Lemma 5.27. We should prove that $w \in L(R_{\mathrm{uV}}) \cap \{\mathrm{u}\}^\star$ implies $|w| = 2 \cdot 3^n$ for some $n \geq 0$. Let $\Phi : \{\mathrm{u},\mathrm{V}\}^\star \to \mathbb{N}$ be defined inductively as follows:

$$\begin{aligned} \Phi(\epsilon) &= 0, \\ \Phi\big(\mathrm{u}^k w\big) &= k + \Phi(w)\,, \\ \Phi(\mathrm{V}w) &= 1 + 3 \cdot \Phi(w) \end{aligned}$$

for each $k \geq 1$ and $w \in \{\mathrm{u},\mathrm{V}\}^\star$. Observe that we have assigned a unique value of Φ to each word from $\{\mathrm{u},\mathrm{V}\}^\star$. Next, we describe effects of the rules of R_{uV} to the value of Φ.

0) The rule 0 can only rewrite $w_1 = \epsilon$ to $w_2 = \mathrm{uu}$. We have $\Phi(w_1) = 0$ and $\Phi(w_2) = 2$.

1) The rule 1 rewrites $w_1 = \mathrm{u}w$ to $w_2 = \mathrm{uuV}w$ for some $w \in \{\mathrm{u},\mathrm{V}\}^\star$. We have $\Phi(w_1) = 1 + \Phi(w)$ and $\Phi(w_2) = 3 + 3 \cdot \Phi(w)$. Thus, $\Phi(w_2) = 3 \cdot \Phi(w_1)$.

2) The rule 2 rewrites $w_1 = \overline{w}\mathrm{Vu}w$ to $w_2 = \overline{w}\mathrm{uuuV}w$ for some $w, \overline{w} \in \{\mathrm{u},\mathrm{V}\}^\star$. We have

$$\Phi(\mathrm{Vu}w) = \Phi(\mathrm{uuuV}w) = 4 + 3 \cdot \Phi(w)\,.$$

   It follows that $\Phi(w_1) = \Phi(w_2)$.

3) The rule 3 rewrites $w_1 = \overline{w}\mathrm{Vu}$ to $w_2 = \overline{w}\mathrm{uuuu}$ for some $\overline{w} \in \{\mathrm{u},\mathrm{V}\}^\star$. We have $\Phi(\mathrm{Vu}) = \Phi(\mathrm{uuuu}) = 4$ and thus $\Phi(w_1) = \Phi(w_2)$.

Together, each $w \in L(R_{\mathrm{uV}})$ has $\Phi(w) = 2 \cdot 3^n$ for some $n \geq 0$. As $\Phi(w) = |w|$ for each $w \in \{\mathrm{u}\}^\star$, the proof is complete.

□

Proof of Lemma 5.31. We should show that

$$00\,(1100)^\alpha\,1000\,(1100)^\beta \dashv\!\!\!\vdash^\star_{01} 00\,(1100)^{\alpha+9}\,1000\,(1100)^{\beta-1}$$

holds for each $\alpha, \beta > 0$. Indeed, it is enough to apply the following rules:

$$
\begin{aligned}
00\,(1100)^{\alpha}\,1000\,(1100)^{\beta} &\dashv_{01} 00\,(1100)^{\alpha}\,100\underline{11}0\,(1100)^{\beta} \dashv_{01}\\
00\,(1100)^{\alpha}\,1\underline{10}00110\,(1100)^{\beta} &\dashv_{01} 00\,(1100)^{\alpha+1}\,\underline{11}0110\,(1100)^{\beta} \dashv_{01}\\
00\,(1100)^{\alpha+1}\,110\underline{01}110\,(1100)^{\beta} &\dashv_{01} 00\,(1100)^{\alpha+2}\,111\underline{01}\,(1100)^{\beta} \dashv_{01}\\
00\,(1100)^{\alpha+2}\,11\underline{00}1001\,(1100)^{\beta} &\dashv_{01} 00\,(1100)^{\alpha+3}\,1\underline{10}001\,(1100)^{\beta} \dashv_{01}\\
00\,(1100)^{\alpha+4}\,\underline{11}01\,(1100)^{\beta} &\dashv_{01} 00\,(1100)^{\alpha+4}\,1101\underline{00}100\,(1100)^{\beta-1} \dashv_{01}\\
00\,(1100)^{\alpha+4}\,110\underline{01}1100100\,(1100)^{\beta-1} &\dashv_{01} 00\,(1100)^{\alpha+5}\,11\underline{00}100100\,(1100)^{\beta-1} \dashv_{01}\\
00\,(1100)^{\alpha+6}\,1\underline{10}00100\,(1100)^{\beta-1} &\dashv_{01} 00\,(1100)^{\alpha+7}\,01\underline{10}00\,(1100)^{\beta-1} \dashv_{01}\\
00\,(1100)^{\alpha+7}\,\underline{11}011000\,(1100)^{\beta-1} &\dashv_{01} 00\,(1100)^{\alpha+7}\,110\underline{01}11000\,(1100)^{\beta-1} \dashv_{01}\\
00\,(1100)^{\alpha+8}\,11\underline{00}1000\,(1100)^{\beta-1}.
\end{aligned}
$$

□

Chapter 6

Conclusions and Future Work

Synchronization Thresholds

In Chapter 2 we have proved that the subset synchronization threshold of DFA and the careful synchronization threshold of PFA are both strongly exponential even under two heavy restrictions: binary alphabets and strong connectivity. The multiplicative constants in the exponents do not seem to be the largest possible, so it may deserve a more precise study to determine the threshold functions. There is also no method for giving upper bounds concerning various alphabet sizes. If the Černý conjecture holds, binary cases are the hardest possible for the classical synchronization of DFA, but this still may not hold in the generalized settings.

From a more general viewpoint, our results give a partial answer to the informal question: *Which features of DFA are needed for obtaining strongly exponential thresholds of subset synchronization?* However, for many interesting restrictions we do not even know whether the corresponding thresholds are superpolynomial. Namely, such restricted classes include monotonic and aperiodic automata, cyclic and one-cluster automata, Eulerian automata, commutative automata and others. For each of these classes it is also an open question whether Subset synchronizability (or Careful synchronizability) is solvable in polynomial time with the corresponding restriction. Moreover, for the careful synchronization threshold $\mathrm{car}(n)$ the gap between the lower bound $\mathcal{O}\big(n^2 \cdot 4^{\frac{n}{3}}\big)$ and the upper bound $\Omega\big(3^{\frac{n}{3}}\big)$ is open, though it is subject to an active research.

There are several current research directions related to the classical synchronization of DFA. One of them, concerning binary automata with sink states, was discussed in Section 2.2. Our new series does not seem to present the worst possible cases, but it still may be useful in further research. Other such directions include the study of Eulerian automata: a current common work of the author and Marek Szykuła should present certain series of Eulerian DFA (see Figure 6.1), where each n-state automaton with odd n has reset threshold equal to $\frac{n^2-2}{3}$, which is the value that is conjectured to be the worst possible according to computational search.

As for general upper bounds, the new method used recently by Trahtman (see Section 1.3.2) still

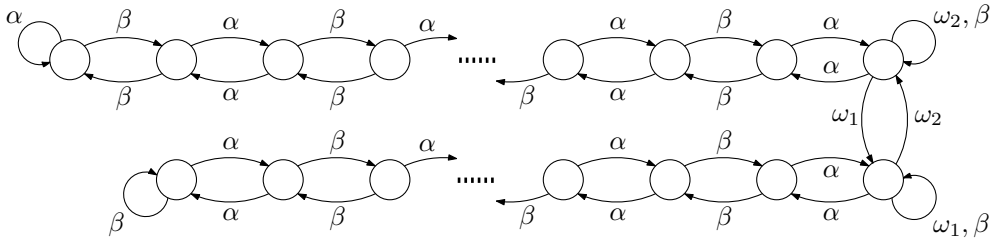


Figure 6.1: A series of n-state Eulerian automata, which is conjectured to have synchronization threshold equal to $\frac{n^2-2}{3}$

seems to provide a possible way for lowering the bound, though the Trahtman's particular proof was wrong. Namely, from the correct claims in [93] it follows that if the following conjecture holds true, then $\mathrm{C}_n \leq \frac{7}{48}n^3 + \mathcal{O}(n^2)$ for each $n \geq 1$.

Conjecture 6.1. *There exists $K \geq 1$ such that for each n-state synchronizing DFA $A = (Q, \Sigma, \delta)$ and each $s \in Q$ there is a word $w \in \Sigma^\star$ with $|w| \leq Kn$ and $s \notin \delta(Q, w)$.*

However, correctness of this simply-looking conjecture is still unknown.
Clever methods were developed for efficient enumeration of non-isomorphic DFA in order to produce useful experimental data regarding Černý conjecture. So far, the conjecture was verified for all binary automata with at most 11 states [52]. Besides of that, the experimental results show certain interesting trends in numerical distributions of possible synchronization thresholds.

Computational Complexity

In Chapter 3 we have closed a former research of restrictions and parameterized complexity of SYN. Chapter 4 considered restrictions and parameterized complexity of SRCP and restrictions of SRCW. In the last case we have completely characterized binary words w that make the problem SRCW NP-complete if restricted to $|\Sigma| = 2$,$W = \{w\}$, and proved that if we require strong connectivity, the case with $w = abb$ becomes solvable in polynomial time, though in the basic form it is NP-complete. For any w such that the first letter equals to the last one and both a, b occur in w, we have proved that the NP-completeness holds even under the requirement of strong connectivity. However, current joint work with Adam Roman should provide a full classification of binary words even in the strongly connected case (i.e. fill the first column of Table 4.3). A proof of the following claim is currently being inspected:

Conjecture 6.2. *Let w be a word from $\{a, b\}^\star$ distinct from abb and baa. If $\mathrm{SRCW}_{2,\{w\}}$ is NP-complete, then $\mathrm{SRCW}^{\mathcal{SC}}_{2,\{w\}}$ is also NP-complete.*

Other immediate goals of future research are to give classifications of words over non-binary alphabets and to study SRCW restricted to non-singleton sets of words. As for other computational problems related to road coloring, there is e.g. an interest in graphs that result in a synchronizing automaton after *any* valid coloring of edges. The complexity of recognizing such graphs is unknown.
Important computational tasks also include recognizing automata with various order-preserving properties that guarantee low synchronization thresholds. While it is known that recognizing monotonic automata is NP-complete [90], for many related classes there is no such result.

Jumps and Contextual Deleting

It is a task for future research to provide really alternative descriptions of the class **GJFA**. There also remain open questions about decidability, specifically regarding equivalence, universality, inclusion, and regularity of GJFA, see [65]. It also seems that general jumping finite automata are not an ideal formal model of systems with discontinuous information processing, so it may deserve a suitable modification or generalization.
As for clearing restarting automata, key open questions deal with λ-*confluence* [67]. Informally, a clearing restarting automaton is λ-confluent if $w \vdash^\star v$ together with $w \vdash^\star \epsilon$ imply $v \vdash^\star \epsilon$. For systems with this property, language membership can be tested in linear time. In general, λ-confluence of a given clearing restarting automaton is undecidable, but it is not known whether it becomes decidable for 1-clearing restarting automata [67].
Two main generalizations of clearing restarting automata were introduced [25, 97] in order to enlarge the descriptional power, following trends that occur in related models and keeping certain simplicity:

1. So-called Δ-*clearing restarting automata* are able not only to delete factors but also to replace them with a special nonterminal symbol Δ, which can be then used in contexts. Surprisingly, it turns out that Δ-clearing restarting automata accept all context-free languages [97].

2. In a $\Delta^\star$-*clearing restarting automaton*, a factor may be also replaced with a word of the form Δ^j for $j \geq 0$.

There are multiple open questions regarding the classes of languages accepted by these two types of automata [97]. In practical use in linguistics, *grammatical inference* (i.e. *learning*) of such automata from positive and negative examples comes into play. Results and challenges about grammatical inference of the above models were presented in [24]. There exists software due to Černo, which is suitable for basic experimental work with the generalized variants of clearing restarting automata and also implements sophisticated learning algorithms.

Bibliography

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

[2] A. Alhazov, A. Krassovitskiy, Y. Rogozhin, and S. Verlan. Small size insertion and deletion systems. In C. Martin-Vide, editor, *Scientific Applications of Language Methods*, pages 459–524. Imperial College Press, 2010.

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

[4] J. Almeida and B. Steinberg. Matrix mortality and the Černý-Pin conjecture. In V. Diekert and D. Nowotka, editors, *Developments in Language Theory*, volume 5583 of *Lecture Notes in Computer Science*, pages 67–80. Springer Berlin Heidelberg, 2009.

[5] D. Ananichev. The annulation threshold for partially monotonic automata. *Russian Mathematics*, 54(1):1–9, 2010.

[6] D. Ananichev and M. Volkov. Synchronizing monotonic automata. In *Developments in Language Theory*, volume 2710 of *Lecture Notes in Computer Science*, pages 162–162. Springer Berlin Heidelberg, 2003.

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

[8] D. Ananichev, M. Volkov, and Y. Zaks. Synchronizing automata with a letter of deficiency 2. In O. Ibarra and Z. Dang, editors, *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 433–442. Springer Berlin Heidelberg, 2006.

[9] D. S. Ananichev, M. V. Volkov, and V. V. Gusev. Primitive digraphs with large exponents and slowly synchronizing automata. In *Combinatorics and graph theory, Part IV*, volume 402 of *Zap. Nauchn. Sem. POMI*, pages 9–39. POMI, 2012.

[10] M.-P. Béal. A note on Černý's conjecture and rational series. 2003.

[11] M.-P. Béal, E. Czeizler, J. Kari, and D. Perrin. Unambiguous automata. *Mathematics in Computer Science*, 1(4):625–638, 2008.

[12] M.-P. Béal and D. Perrin. A quadratic algorithm for road coloring. *Discrete Applied Mathematics*, 169(0):15 – 29, 2014.

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

[14] M.-P. Béal and D. Perrin. Symbolic dynamics and finite automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, pages 463–506. Springer Berlin Heidelberg, 1997.

[15] M. Berlinkov. On two algorithmic problems about synchronizing automata. In A. Shur and M. Volkov, editors, *Developments in Language Theory*, volume 8633 of *Lecture Notes in Computer Science*, pages 61–67. Springer International Publishing, 2014.

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

[17] M. V. Berlinkov. Synchronizing automata on quasi-eulerian digraph. In *Proceedings of the 17th international conference on Implementation and Application of Automata*, CIAA'12, pages 90–100, Berlin, Heidelberg, 2012. Springer-Verlag.

[18] M. V. Berlinkov. Synchronizing quasi-eulerian and quasi-one-cluster automata. *International Journal of Foundations of Computer Science*, 24(06):729–745, 2013.

[19] M. V. Berlinkov. Approximating the minimum length of synchronizing words is hard. *Theory of Computing Systems*, 54(2):211–223, 2014.

[20] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423 – 434, 2009.

[21] P. Bonizzoni and N. Jonoska. Regular splicing languages must have a constant. In G. Mauri and A. Leporati, editors, *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 82–92. Springer Berlin Heidelberg, 2011.

[22] G. Budzban. Semigroups and the generalized road coloring problem. *Semigroup Forum*, 69(2):201–208, 2004.

[23] H. Burkhard. Zum Längenproblem homogener Experimente an determinierten und nichtdeterministischen Automaten. *Elektronische Informationsverarbeitung und Kybernetik*, 12(6):301–306, 1976.

[24] P. Černo. Clearing restarting automata and grammatical inference. In T. O. Jeffrey Heinz, Colin de la Higuera, editor, *Proceedings of the Eleventh International Conference on Grammatical Inference*, volume 21 of *JMLR Workshop and Conference Proceedings*, pages 54–68, 2012.

[25] P. Černo and F. Mráz. Clearing restarting automata. *Fundamenta Informaticae*, 104(1):17–54, 2010.

[26] J. Černý. Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis*, 14(3):208–216, 1964.

[27] J. Černý, A. Pirická, and B. Rosenauerová. On directable automata. *Kybernetica*, 7:289–298, 1971.

[28] H. Cho, S.-W. Jeong, F. Somenzi, and C. Pixley. Synchronizing sequences and symbolic traversal techniques in test generation. *J. Electronic Testing*, 4(1):19–31, 1993.

[29] B. Delyon and O. Maler. On the effects of noise and speed on computations. *Theoretical Computer Science*, 129(2):279 – 291, 1994.

[30] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag, 2013. 763 pp.

[31] L. Dubuc. Les automates circulaires biaisés vérifient la conjecture de Černý. *RAIRO - Theoretical Informatics and Applications - Informatique Theorique et Applications*, 30(6):495–505, 1996.

[32] A. Ehrenfeucht, D. Haussler, and G. Rozenberg. On regularity of context-free languages. *Theoretical Computer Science*, 27(3):311 – 332, 1983.

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

[34] H. Fernau, P. Heggernes, and Y. Villanger. A multivariate analysis of some DFA problems. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications*, volume 7810 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin Heidelberg, 2013.

[35] H. Fernau, P. Heggernes, and Y. Villanger. A multi-parameter analysis of hard problems on deterministic finite automata. *Journal of Computer and System Sciences*, 81(4):747 – 765, 2015.

[36] C. Flye Sainte-Marie. Solution to question nr. 48. *L'intermédiaire des Mathématicians*, 1:107–110, 1894.

[37] P. Frankl. An extremal problem fro two families of sets. *Eur. J. Comb.*, 3:125–127, 1982.

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

[39] S. Ginsburg. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine. *J. ACM*, 5(3):266–280, July 1958.

[40] F. Gonze, R. M. Jungers, and A. N. Trahtman. A note on a recent attempt to improve the pin-frankl bound. *CoRR*, abs/1412.0975, 2014.

[41] P. Goralčík, Z. Hedrlín, V. Koubek, and J. Ryšlinková. A game of composing binary relations. *RAIRO - Theoretical Informatics and Applications - Informatique Theorique et Applications*, 16(4):365–369, 1982.

[42] V. Gusev. Lower bounds for the length of reset words in eulerian automata. In G. Delzanno and I. Potapov, editors, *Reachability Problems*, volume 6945 of *Lecture Notes in Computer Science*, pages 180–190. Springer Berlin Heidelberg, 2011.

[43] V. Gusev. Synchronizing automata of bounded rank. In N. Moreira and R. Reis, editors, *Implementation and Application of Automata*, volume 7381 of *Lecture Notes in Computer Science*, pages 171–179. Springer Berlin Heidelberg, 2012.

[44] D. Haussler. Insertion languages. *Information Sciences*, 31(1):77 – 89, 1983.

[45] H. A. Helfgott and Á. Seress. On the diameter of permutation groups. *Annals of Mathematics*, 179:611–658, 2014.

[46] B. Imreh and M. Steinby. Directable nondeterministic automata. *Acta Cybern.*, 14(1):105–115, Feb. 1999.

[47] M. Ito, L. Kari, and G. Thierrin. Insertion and deletion closure of languages. *Theoretical Computer Science*, 183(1):3 – 19, 1997.

[48] M. Ito and K. Shikishima-Tsuji. Some results on directable automata. In J. Karhumäki, H. Maurer, G. Pãun, and G. Rozenberg, editors, *Theory Is Forever*, volume 3113 of *Lecture Notes in Computer Science*, pages 125–133. Springer Berlin Heidelberg, 2004.

[49] M. Ito and K. Shikishima-Tsuji. Some results on directable automata. In J. Karhumäki et al., editors, *Theory Is Forever*, volume 3113 of *Lecture Notes in Computer Science*, pages 125–133. Springer Berlin Heidelberg, 2004.

[50] J. Kari. A counter example to a conjecture concerning synchronizing words in finite automata. *EATCS Bulletin*, 73:146, 2001.

[51] J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theor. Comput. Sci.*, 295(1-3):223–232, Feb. 2003.

[52] J. Kowalski and M. Szykula. The Černý conjecture for small automata: experimental report. *CoRR*, abs/1301.2092, 2013.

[53] D. Kozen. Lower bounds for natural proof systems. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 254–266, 1977.

[54] D. Lee and M. Yannakakis. Testing finite-state machines: state identification and verification. *Computers, IEEE Transactions on*, 43(3):306–320, 1994.

[55] M. Marcus and G. Păun. Regulated Galiukschov semicontextual grammars. *Kybernetika*, 26(4):316–326, 1990.

[56] S. W. Margolis, J.-E. Pin, and M. V. Volkov. Words guaranteeing minimum image. *Int. J. Found. Comput. Sci.*, 15(2):259–276, 2004.

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

[58] P. Martyugin. Complexity of problems concerning reset words for some partial cases of automata. *Acta Cybern.*, 19(2):517–536, 2009.

[59] P. Martyugin. A lower bound for the length of the shortest carefully synchronizing words. *Russian Mathematics*, 54(1):46–54, 2010.

[60] P. Martyugin. Complexity of problems concerning reset words for cyclic and eulerian automata. In B. Bouchou-Markhoff, P. Caron, J.-M. Champarnaud, and D. Maurel, editors, *Implementation and Application of Automata*, volume 6807 of *Lecture Notes in Computer Science*, pages 238–249. Springer Berlin Heidelberg, 2011.

[61] P. Martyugin. Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory of Computing Systems*, 54(2):293–304, 2014.

[62] P. V. Martyugin. Careful synchronization of partial automata with restricted alphabets. In A. A. Bulatov and A. M. Shur, editors, *Computer Science - Theory and Applications*, volume 7913 of *Lecture Notes in Computer Science*, pages 76–87. Springer Berlin Heidelberg, 2013.

[63] R. McNaughton and S. A. Papert. *Counter-Free Automata (M.I.T. Research Monograph No. 65)*. The MIT Press, 1971.

[64] A. Meduna and P. Zemek. Jumping finite automata. *International Journal of Foundations of Computer Science*, 23(7):1555–1578, 2012.

[65] A. Meduna and P. Zemek. *Regulated Grammars and Automata*. Springer US, 2014. Chapter 17: Jumping Finite Automata.

[66] E. F. Moore. Gedanken-experiments on sequential machines. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 129–153. Princeton University Press, Princeton, NJ, 1956.

[67] F. Mráz and F. Otto. Lambda-confluence is undecidable for clearing restarting automata. In S. Konstantinidis, editor, *Implementation and Application of Automata*, volume 7982 of *Lecture Notes in Computer Science*, pages 256–267. Springer Berlin Heidelberg, 2013.

[68] B. K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 132–142, Washington, 1986. IEEE Computer Society.

[69] J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In *Proceedings of the 35th international conference on Mathematical foundations of computer science*, MFCS'10, pages 568–579, Berlin, Heidelberg, 2010. Springer-Verlag.

[70] P. Panteleev. Preset distinguishing sequences and diameter of transformation semigroups. In A.-H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications*, volume 8977 of *Lecture Notes in Computer Science*, pages 353–364. Springer International Publishing, 2015.

[71] J.-E. Pin. Sur les mots synchronisants dans un automate fini. *Elektron. Informationsverarb. Kybernet.*, 14:293–303, 1978.

[72] J.-E. Pin. Sur un cas particulier de la conjecture de Cerny. In *Proceedings of the Fifth Colloquium on Automata, Languages and Programming*, pages 345–352, London, UK, UK, 1978. Springer-Verlag.

[73] J.-E. Pin. On two combinatorial problems arising from automata theory. *Annals of Discrete Mathematics*, 17:535–548, 1983.

[74] C. Pixley. Introduction to a computational theory and implementation of sequential hardware equivalence. In E. Clarke and R. Kurshan, editors, *Computer-Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 54–64. Springer Berlin Heidelberg, 1991.

[75] G. Păun. Two theorems about galiukschov semicontextual languages. *Kybernetika*, 21(5):360–365, 1985.

[76] G. Păun, G. Rozenberg, and A. Salomaa. Insertion-deletion systems. In *DNA Computing: New Computing Paradigms*, pages 187–215. Springer Berlin Heidelberg, 1998.

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

[78] A. Roman. P-NP threshold for synchronizing road coloring. In A.-H. Dediu and C. Martín-Vide, editors, *Language and Automata Theory and Applications*, volume 7183 of *Lecture Notes in Computer Science*, pages 480–489. Springer Berlin Heidelberg, 2012.

[79] A. Roman and M. Drewienkowski. A complete solution to the complexity of synchronizing road coloring for non-binary alphabets. *Information and Computation, in print*, 2015.

[80] I. K. Rystsov. Reset words for commutative and solvable automata. *Theoret. Comput. Sci.*, 172:273–279, 1997.

[81] A. Salomaa. Compositions over a finite domain: From completeness to synchronizable automata. In *A Half-century of Automata Theory*, pages 131–143. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2001.

[82] 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 Computer Science*, pages 5–33. Springer Berlin Heidelberg, 2005.

[83] P. Savický and S.Vaněček. Hledání synchronizačních slov konečných automatů pomocí lineární algebry. 1983.

[84] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. ACM.

[85] M. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190 – 194, 1965.

[86] P. H. Starke. Eine Bemerkung über homogene Experimente. *Elektr. Informationverarbeitung und Kyb.*, 2:257–259, 1966.

[87] P. H. Starke. *Abstrakte Automaten*. V.E.B. Deutscher Verlag der Wissenschaften, Berlin, 1969.

[88] B. Steinberg. The averaging trick and the Černý conjecture. In *Proceedings of the 14th international conference on Developments in language theory*, DLT'10, pages 423–431, Berlin, Heidelberg, 2010. Springer-Verlag.

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

[90] M. Szykula. Checking if an automaton is monotonic is np-complete. *CoRR*, abs/1407.5068, 2014.

[91] A. Trahtman. The road coloring problem. *Israel Journal of Mathematics*, 172(1):51–60, 2009.

[92] A. Trahtman. An algorithm for road coloring. In C. Iliopoulos and W. Smyth, editors, *Combinatorial Algorithms*, volume 7056 of *Lecture Notes in Computer Science*, pages 349–360. Springer Berlin Heidelberg, 2011.

[93] A. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In O. Owe, M. Steffen, and J. Telle, editors, *Fundamentals of Computation Theory*, volume 6914 of *Lecture Notes in Computer Science*, pages 173–180. Springer Berlin Heidelberg, 2011.

[94] A. N. Trahtman. The Cerný conjecture for aperiodic automata. *Discrete Mathematics & Theoretical Computer Science*, 9(2):3–10, 2007.

[95] A. N. Trahtman. The road coloring and Černy conjecture. In J. Holub and J. Žďárek, editors, *Proceedings of the Prague Stringology Conference 2008*, pages 1–12, Czech Technical University in Prague, Czech Republic, 2008.

[96] N. Travers and J. Crutchfield. Exact synchronization for finite-state sources. *Journal of Statistical Physics*, 145(5):1181–1201, 2011.

[97] P. Černo and F. Mráz. δ-clearing restarting automata and cfl. In G. Mauri and A. Leporati, editors, *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 153–164. Springer Berlin Heidelberg, 2011.

[98] S. Verlan. Recent developments on insertion-deletion systems. *Computer Science Journal of Moldova*, 18(2):210–245, 2010.

[99] 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 Computer Science*, pages 11–27. Springer Berlin Heidelberg, 2008.

[100] M. V. Volkov. Synchronizing automata preserving a chain of partial orders. In *Proceedings of the 12th international conference on Implementation and application of automata*, CIAA'07, pages 27–37, Berlin, Heidelberg, 2007. Springer-Verlag.

[101] M. V. Volkov and D. S. Ananichev. Some results on Černý type problems for transformation semigroups. In *Semigroups And Languages*, chapter 2, pages 23–42. World Scientific, 2004.

[102] V. Vorel. Complexity of a problem concerning reset words for eulerian binary automata. In A.-H. Dediu, C. Martín-Vide, J.-L. Sierra-Rodríguez, and B. Truthe, editors, *Language and Automata Theory and Applications*, volume 8370 of *Lecture Notes in Computer Science*, pages 576–587. Springer International Publishing, 2014.

[103] V. Vorel. Subset synchronization of transitive automata. In *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014.*, pages 370–381, 2014.

[104] V. Vorel and A. Roman. Complexity of road coloring with prescribed reset words. In A.-H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications*, Lecture Notes in Computer Science, pages 161–172. Springer International Publishing, 2015.

[105] V. Vorel and A. Roman. Parameterized complexity of synchronization and road coloring. *Discrete Mathematics & Theoretical Computer Science*, 17(1):307–330, 2015.