Comenius University in Bratislava
Faculty of mathematics, physics and informatics

Longest Shortest Words in Regular Languages

Master's Thesis

2021
Bc. Matúš Juran

Comenius University in Bratislava
Faculty of mathematics, physics and informatics

Longest Shortest Words in Regular Languages

Master's Thesis

Study programme:	Computer Science
Field of study:	Computer Science
Department:	Department of Computer Science
Supervisor:	RNDr. Peter Kostolányi, PhD.

Bratislava, 2021
Bc. Matúš Juran

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Matúš Juran
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Longest Shortest Words in Regular Languages

Annotation: Let C be some class of finite automata (such as, e.g., two-way nondeterministic) and A an automaton from this class. Denote by s(A) the length of a shortest word recognised by A and by L(C,n) the maximum value of s(A) over all n-state automata A from C recognising non-empty languages. Asymptotic properties of the function L(C,n) have already been studied for several classes of finite automata and other descriptional mechanisms for regular languages. The thesis will provide a survey of results known in this area and initiate the study of basic properties of L(C,n) for some classes of finite automata yet unexplored in this aspect.

Supervisor:	RNDr. Peter Kostolányi, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	05.11.2019

Approved: 21.11.2019 prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

Student Supervisor

15490825

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matúš Juran
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Longest Shortest Words in Regular Languages
Najdlhšie najkratšie slová v regulárnych jazykoch

Anotácia: Nech C je trieda konečných automatov (ako napríklad dvojsmerné nedeterministické automaty) a A je automat z tejto triedy. Označme ako s(A) dĺžku najkratšieho slova rozoznávaného automatom A a ako L(C,n) maximálnu hodnotu s(A) cez všetky n-stavové automaty A z C rozoznávajúce neprázdny jazyk. Asymptotické vlastnosti funkcie L(C,n) sa už skúmali pre viaceré triedy konečných automatov a ďalších popisných mechanizmov pre regulárne jazyky. V práci bude podaný prehľad známych výsledkov v tejto oblasti a bude v nej započatý výskum základných vlastností funkcie L(C,n) pre niektoré triedy konečných automatov doposiaľ v tomto aspekte nepreskúmané.

Vedúci: RNDr. Peter Kostolányi, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 05.11.2019

Dátum schválenia: 21.11.2019 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgement: I would like to thank all those who directly or indirectly helped me finish this thesis.

Abstract

In this thesis, we study the following problem: given an automaton with a given number of states, how long can the shortest accepted word potentially be? This problem was studied in the past in the case of nondeterministic finite automata and two-way automata. We provide an overview of relevant previous results and define the function lsw which generalizes this problem to sequences of finite sets of languages. Then, we study the problem with rotating finite automata and alternating finite automata as models of choice. We describe lower and upper bounds for accepted languages, their intersections and complements for alphabets of various sizes. Lastly, we compare the values that the function lsw attains for sequences defined by certain models.

Keywords: shortest word, rotating finite automaton, alternating finite automaton

Abstrakt

V tejto práci sa zaoberáme nasledujúcim problémom: aké dlhé môže byť najkratšie slovo akceptované automatom s daným počtom stavov? Tento problém bol už skúmaný pre nedeterministické konečné automaty a dvojsmerné automaty. Poskytujeme prehľad relevantných doterajších výsledkov a definujeme funkciu lsw, ktorá daný problém zovšeobecňuje na postupnosti konečných množín jazykov. Následne daný problém skúmame pre rotujúce konečné automaty a alternujúce konečné automaty. Popisujeme dolné a horné odhady pre akceptované jazyky, ich prieniky a komplementy pre abecedy rôznych veľkostí. Na záver porovnávame hodnoty, ktoré funkcia lsw nadobúda pre postupnosti definované vybranými modelmi.

Kľúčové slová: najkratšie slovo, rotujúci konečný automat, alternujúci konečný automat

Contents

Introduction

When compared to other families studied in the theory of formal languages, regular languages are a relatively simple one. This makes them particularly suitable for studying novel problems which would be much more difficult to tract in more complex cases, such as context-free languages. The results discovered and the techniques developed when studying regular languages may serve as a baseline for further research.

The research in this thesis stems from the following simple problem: how long is the shortest word in a given language? In some cases, the solution is trivial - for example, when the language is finite and all the words are enumerated. It is also trivial to construct a singleton language where the only word is arbitrarily long. The complexity arises from the fact that the language may be infinite, but still has to be described in a finite way. To achieve this, models such as nondeterministic finite-state automata, regular grammars or regular expressions can be used. For these models, we may also measure their descriptional complexity - for example, the number of states of an automaton. This gives rise to a related problem: how long may the shortest word be if the model's descriptional complexity is limited by some parameter? While primarily of theoretical interest, there is also one possible application of the solutions. If we cannot test the emptiness of a language directly (which is possible in the case of regular languages), but it is still possible to test whether the language contains a specific word, knowing how long the shortest word may be allows us to stop as soon as it is known that the language contains no word that is short enough.

The problem of longest shortest words was previously studied in the case of nondeterministic finite automata (see [1]) and in the case of two-way automata (see [6] and [14]). While exact values were obtained in the former case, only lower and upper bounds are known in the latter case. It was also shown that for two-way automata, the length of the shortest word may also depend on the size of the alphabet, since certain lower bounds require alphabets whose size grows with the descriptional complexity. In this thesis, we study the length of the shortest words accepted by two other models: rotating automata and alternating finite automata.

In Chapter 1, we present the definitions of the models we shall be studying, as well as the notion of descriptional complexity and some mathematical preliminaries.

In Chapter 2, we summarize the results from earlier works and define the function

lsw which generalizes the notion of longest shortest word from models of automata to sequences of finite sets of languages.
In Chapter 3, we study the length of the shortest words accepted by rotating automata. Additionally, we examine the shortest words in the intersection of two languages accepted by rotating automata.
In Chapter 4, we study the same problems in the case of alternating finite automata. We also study the shortest words in complements of languages accepted by alternating automata.
In Chapter 5, we compare some bounds obtained by conversions between models with results obtained by studying the individual models.

Chapter 1

Preliminaries

In this chapter, we shall present the necessary definitions, mathematical preliminaries and some of the known results about descriptional complexity of models describing regular languages.

1.1 Models of Finite Automata

The simplest computational model for which the longest shortest words were studied is the deterministic finite automaton.

Definition 1. *A deterministic finite automaton (DFA) is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$ *where* Q *is a finite set of states,* Σ *is a finite, nonempty alphabet,* $\delta : Q \times \Sigma \to Q$ *is a complete transition function,* $q_0 \in Q$ *is the initial state and* $F \subseteq Q$ *is the set of accepting states.*

A configuration of a DFA is a pair (q, w) where $q \in Q$ and $w \in \Sigma^*$. The word w represents the remaining unread input, while the state q represents the current state of the automaton. A computation step is a relation $\vdash$ on the configurations such that $(q_1, cw) \vdash (q_2, w)$ if $\delta(q_1, c) = q_2$. The language accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is the set of words w such that $(q_0, w) \vdash^* (q_F, \varepsilon)$ for some $q_F \in F$. It is denoted by $L(A)$.

Definition 2. *A language is called regular if it is recognized by some deterministic finite automaton.*

We shall denote the family of regular languages as $\mathscr{R}$.

Using this definition of a DFA, the computation always has exactly one possible continuation. If we did not require the δ-function to be complete, there would be at most one possible continuation at any point instead. The modified model is still deterministic, but such a definition is less common. However, allowing multiple continuations as well results in a different model - the nondeterministic finite automaton.

Definition 3. *A nondeterministic finite automaton (NFA) is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$ *where* Q *is a finite set of states,* Σ *is a finite, nonempty alphabet,* $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ *is a transition function,* $q_0 \in Q$ *is the initial state and* $F \subseteq Q$ *is the set of accepting states.*

The configurations can be defined the same way as for DFAs. Since the δ-function returns a set of states, the definition of a computation step is slightly modified; for $q_1, q_2 \in Q, c \in \Sigma \cup \{\varepsilon\}$ and $w \in \Sigma^*$, it holds that $(q_1, cw) \vdash (q_2, w)$ if $q_2 \in \delta(q_1, c)$. Then, we can define the accepted language as $L(A) = \{w \in \Sigma^* \mid \exists q_F \in F : (q_0, w) \vdash^* (q_F, \varepsilon)\}$.

Next, we shall define the models for which the longest shortest words were studied by Dobronravov et al. [6].

Definition 4. *A nondeterministic two-way finite automaton (2NFA) is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$*, where* Q *is the finite set of states,* Σ *is an alphabet which does not contain special symbols* ¢ *and* \$*,* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of accepting states and* $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to 2^{Q \times \{+1,-1\}}$ *is the transition function.*

The automaton starts the computation at the left endmarker ¢ and at every step changes the state and moves the head either to the left or to the right based on the function δ. A configuration of a 2NFA is a triple $(q, ¢w\$, i)$ where $q \in Q$, $w \in \Sigma^*$ and $i \in \{0, \ldots, |w| + 1\}$. A computation step is a relation on the configurations such that $(q_1, ¢w\$, i) \vdash (q_2, ¢w\$, j)$ if either $j = i + 1$ and $(q_2, +1) \in \delta(q_1, c)$ where c is the i-th symbol on the tape (the left endmarker is considered to be the 0-th symbol) or $j = i - 1$ and $(q_2, -1) \in \delta(q_1, c)$ where c is the i-th symbol on the tape. The input $w \in \Sigma^*$ is accepted if $(q_0, ¢w\$, 0) \vdash^* (q_F, ¢w\$, |w| + 1)$ for some $q_F \in F$. Note that the right endmarker must be reached in order to accept the word.

Definition 5. *A 2NFA is called* sweeping *if it only changes its direction on the endmarkers.*

The concept of direction-determinate automata is described in [15].

Definition 6. *Let* q, q_1 *and* q_2 *be some states of a 2NFA, let* c_1 *and* c_2 *be symbols on a tape and let* d_1 *and* d_2 *be* ± 1*. A 2NFA is called direction-determinate if it always holds that if* $(q, d_1) \in \delta(q_1, c_1)$ *and* $(q, d_2) \in \delta(q_2, c_2)$*, then* $d_1 = d_2$*.*

Informally, every state can only be entered from the left or from the right. Every sweeping automaton without redundant states and transitions is also direction-determinate, but not vice versa [6].
Deterministic two-way automata (2DFA) can be defined analogously. The variant of 2DFAs considered in [6] allowed for an incomplete δ-function.

Finally, we shall define the models which we shall study in our research.
Rotating automata can only move the head to the right, but when the right endmarker is reached, the head can return to the left endmarker. The concept was first introduced by Sakoda and Sipser in [21] as *series machines*. The state complexity of rotating automata was studied in [13].

Definition 7. *A rotating nondeterministic finite automaton (RNFA) is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$ *where* Q *is the finite set of states,* Σ *is an alphabet which does not contain special symbols* ¢ *and* \$, $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of accepting states and* $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to 2^Q$ *is the transition function.*

A configuration of a RNFA is a triple $(q, ¢w\$, i)$ where $q \in Q$, $w \in \Sigma^*$ and $i \in \{0, \ldots, |w| + 1\}$. A computation step is a relation on the configurations such that $(q_1, ¢w\$, i) \vdash (q_2, ¢w\$, j)$ if $j \equiv i + 1 \pmod{|w| + 2}$ and $q_2 \in \delta(q_1, c)$ where c is the i-th symbol on the tape. A word $w \in \Sigma^*$ is accepted if $(q_0, ¢w\$, 0) \vdash^* (q_F, ¢w\$, |w| + 1)$ for some $q_F \in F$.

The automaton starts the computation at the left endmarker ¢. When any symbol except for the right endmarker \$ is read, the state changes based on the function δ and the head moves to the right. If the right endmarker is reached in an accepting state, the computation is accepting. If it is reached in a non-accepting state instead, the state shall change according to the δ-function, the head shall move to the left endmarker and the computation shall continue.
As in the 2NFA case, rotating deterministic automata (RDFA) can be defined analogously.

Alternating automata present a way of modelling parallel computations. This model was originally introduced by Chandra, Kozen and Stockmeyer. [4]. The computation can branch out and parallelly continue from several states. Acceptance of the input is then determined by the result of a boolean function which takes the results of all the branches as input. We shall work with a modified version of alternating automata which allows only two types of boolean functions: the logical disjunction of all the branches and the logical conjunction of all the branches. The states to which the disjunction is assigned are referred to as *existential*, while the remaining states are referred to as *universal*.

Definition 8. *An alternating finite automaton (AFA) is a 6-tuple* $A = (Q_\exists, Q_\forall, \Sigma, \delta, q_0, F)$ *where* $Q_\exists$ *is the finite set of existential states,* $Q_\forall$ *is the finite set of universal states,* Σ *is the alphabet,* $q_0 \in Q_\exists \cup Q_\forall$ *is the initial state,* $F \subseteq Q_\exists \cup Q_\forall$ *is the set of accepting states and* $\delta : (Q_\exists \cup Q_\forall) \times (\Sigma \cup \{\varepsilon\}) \to 2^{Q_\exists \cup Q_\forall}$ *is the transition function. There must not exits a sequence of states* $q_1, q_2 \ldots, q_k$ *such that* $q_2 \in \delta(q_1, \varepsilon), \ldots q_k \in \delta(q_{k-1}, \varepsilon)$ *and* $q_1 \in \delta(q_k, \varepsilon)$ *(in other words, there must be no* ε*-cycle).*

Let us denote the set $Q_\exists \cup Q_\forall$ as Q. A configuration of an AFA is a pair (q, w), where $q \in Q$ and $w \in \Sigma^*$. The computation step relation is defined as $(q_1, cw) \vdash (q_2, w)$ if $q_2 \in \delta(q_1, c)$. An accepting computation can be defined recursively; for any state $q_F \in F$, the configuration (q_F, ε) is accepting. If q is an existential state, the computation starting from (q, w) is accepting if there is some $w' \in \Sigma^*$ and some $q' \in Q$ such that $(q, w) \vdash (q', w')$ and the computation starting from (q', w') is accepting. If q is a universal state, every computation starting from (q', w') where $(q, w) \vdash (q', w')$ must be accepting; the computation starting from (q, w) is rejecting otherwise. Note that we do not define a deterministic variant of AFAs since it becomes unnecessary to differentiate between universal and existential states if at most one transition is possible at any point of computation. Therefore, the resulting model would be practically identical with a DFA. Alternation can also be viewed as a combination of nondeterminism and parallelism.

1.2 Descriptional Complexity

For every model, we can define some measure of complexity based on its formal description which can be used to compare the complexity of different instances. In the case of automata, the most usual measure is the number of states. Since this measure is independent from the actual computation and depends solely on the properties of the automaton and its description, it is referred to as *descriptional complexity*. Note that we can also measure the descriptional complexity of other models, such as grammars and regular expressions. For instance, in the case of grammars, one possible measure is the sum of lengths of all the right hand sides, while in the case of regular expressions the complexity can be measured by the number of occurrences of alphabetic symbols.

In our work, we shall focus on variants of automata described in the previous section. For any type of automaton A with a finite set of states Q, we can define its state complexity as the size of the set Q and denote it by $sc(A)$.

We can also define the complexity of a language as the minimal descriptional complexity of a model which recognizes it.

Definition 9. *Let L be a language, let C be some class of models and let dc be a function measuring the descriptional complexity (such as the state complexity of automata). Then the descriptional complexity of the language L with regard to the given class of models and the complexity function is defined as $dc(L) = min\{dc(M) \mid M \in C, L(M) = L\}$, with the minimum of an empty set defined as ∞.*

Next, we shall present the results regarding the equivalence of the models described in the previous section and the effects of conversions on the descriptional complexity.

Proposition 1. *For every NFA A with n states, there exists a DFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant 2^n$.*

Proof. To construct an equivalent DFA, we can use the subset construction from [11]. Thus, 2^n states are always sufficient. □

It is also known that the bound from the previous proposition is tight.

Proposition 2. *For every $n \geqslant 1$, there exists an NFA A such that every equivalent DFA requires 2^n states.*

Proof. The proof is presented as a corollary in [18]. □

Proposition 3. *For every 2NFA A with n states, there exists an NFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant \binom{2n}{n+1}$. For every integer n, there also exists an alphabet Σ with $\Theta(n^n)$ symbols and an n-state 2DFA (and thus also a 2NFA) such that every equivalent NFA requires $\binom{2n}{n+1}$ states.*

Proof. It was proved by Kapoutsis [12] that $\binom{2n}{n+1}$ states are necessary and sufficient to simulate a 2NFA by a one-way nondeterministic automaton. □

Proposition 4. *For every direction-determinate 2NFA A with n states, there exists an NFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant \binom{n}{\lfloor \frac{n}{2} \rfloor}$.*

Proof. The proof can be found in [10]. □

Proposition 5. *For every RNFA A with n states, there exists a 2NFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant 2n$.*

Proof. For every state q of the automaton A, the automaton A' can use two states $(q, \rightarrow)$ and $(q, \leftarrow)$. In the state $(q, \rightarrow)$ A' shall simulate A. However, if the right endmarker is reached, A' can not return to the left endmarker directly. Thus, it shall switch to the state $(q, \leftarrow)$ and move the head to the left until the left endmarker is reached. Then, A' shall move its head one step to the right, change its state to $(q, \rightarrow)$ and continue its computation. □

Proposition 6. *For every AFA A with n states, there exists an NFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant 2^n$.*

Proof. In [9], a construction of an NFA from an AFA is presented. It is also shown that 2^n states are both necessary and sufficient. □

To determine the complexity of a language, we need to show that the complexity of some automaton recognizing it is minimal. A technique attributed to Birget [2] also called the *extended fooling set technique*, can be used to obtain a (not necessarily tight) lower bound on the number of states of an NFA.

Proposition 7. *Let L be a regular language, let $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of pairs of words such that $x_i y_j \in L$ if $i = j$ and either $x_i y_j \notin L$ or $x_j y_i \notin L$ if $i \neq j$. Then any NFA accepting L must have at least n states.*

Proof. For the sake of contradiction, let us suppose that an NFA A accepting L with fewer than n states exists. By pigeonhole principle, there must exist i, j such that A finishes reading both x_i and x_j in the same state q. Without loss of generality, let $x_i y_j \notin L$. Then A must not finish in an accepting state after starting in q and reading y_j. However, since $x_j y_j \in L$, after starting in q and reading y_j, A must finish in an accepting state. Thus, A must not have fewer than n states. □

1.3 Mathematical Preliminaries

In this section, we shall present the definition of the *Landau's function* $g(n)$ and some of its properties.

Definition 10. *Let $lcm(p_1, \ldots, p_k)$ denote the least common multiple of numbers. For every n, the Landau's function is defined as $max\{lcm(p_1, \ldots, p_k) \mid p_1 + \ldots + p_k \leqslant n\}$.*

Without loss of generality, we can assume that $\sum_{i=1}^{k} p_i = n$ - even though the sum can be lower for some choices of n, the least common multiple shall not decrease if some additional number is added to the sequence.
This function is closely related to unary languages (see [5]). The value $g(n)$ is approximately equal to $e^{(1+o(1))\sqrt{n \log n}}$.

The following lemma was proved by Nicolas in [19].

Lemma 1. *Let $g(n)$ denote the Landau's function. It holds that $\lim_{n\to\infty} \frac{g(n+1)}{g(n)} = 1$.*

This immediately gives rise to the following corollary.

Corollary 1. *Let $g(n)$ denote the Landau's function. For any constant $k \in \mathbb{N}$, it holds that $\lim_{n\to\infty} \frac{g(n+k)}{g(n)} = 1$.*

Proof. Let us consider the relation $\sim$ defined as $f(n) \sim g(n) \leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. This relation is commonly used in asymptotic analysis. It follows from Lemma 1 that $g(n) \sim g(n+1), \ldots, g(n+k-1) \sim g(n+k)$. Since the relation $\sim$ is known to be an equivalence relation and thus transitive, it holds that $g(n) \sim g(n+k)$. □

Chapter 2

A Survey of Known Results

In this chapter, we shall formalize the problem of longest shortest words in regular languages and present the key results from [1], [6], [14] and [8], as well as a lemma from [16].

2.1 Formalization of the Problem

For every non-empty language, there exists some integer n such that the language contains no word shorter than n and the language contains a word w of length n. Thus, n is the length of the shortest word in the language. We can define a function which returns such length.

Definition 11. *Let L be a non-empty language. Then, we shall define the* shortest word function *as $sw(L) = min\{|w| \mid w \in L\}$.*

We shall leave the function undefined for the empty language. In other works, this function has also been called the length of the shortest string, or $lss(L)$

Next, we shall present some examples of the sw-function, with an NFA, a context-free grammar and a regular expression as models of choice.

Example 1. *Let A be an NFA given by the following diagram.*

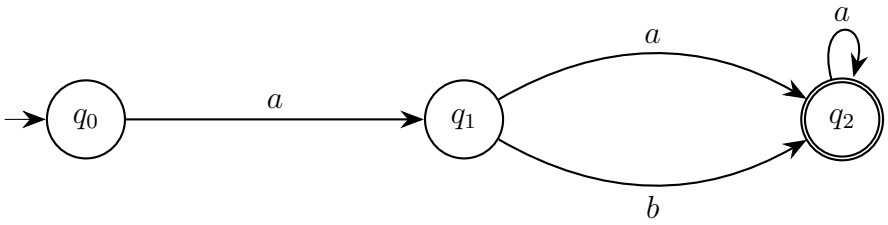

Figure 2.1: The nondeterministic automaton A

Then the shortest words in $L(A)$ are a^2 and ab and thus $sw(L(A)) = 2$.

Example 2. *Let $G = (N, T, P, \sigma)$ be a context-free grammar where $N = \{\sigma\}$, $T = \{a, \overline{a}\}$ and $P = \{\sigma \to a\overline{a}, \sigma \to a\sigma\overline{a}, \sigma \to \sigma\sigma\}$. Then the shortest word in $L(G)$ is $a\overline{a}$ and thus $sw(L(G)) = 2$.*

Example 3. *Let $E = (0^*(01^*)^n0)$ be a regular expression. Then the shortest word in $L(E)$ is 0^{n+1} and thus $sw(L(E)) = n + 1$.*

It can be easily seen that by considering an automaton A which accepts a singleton language $\{w\}$, we can make the value $sw(A)$ arbitrarily large by choosing a long enough word w. The state complexity of such an automaton would be necessarily high.

Therefore, we shall define the following function which computes the maximum possible length of the shortest word among all languages that can be accepted by automata with a given number of states. The function's definition is based on the following idea: if the number of states is bounded by a constant, there exist only finitely many automata and corresponding languages. All of those languages (with the exception of the empty one) contain a shortest word and since the number of languages is finite, it is possible to assign a maximum possible length of the shortest word to every bound on the number of states. A similar measure, called the *rational index*, was introduced by Boasson, Courcelle and Nivat [3]. The definition shall be more general to make it possible to describe more complex settings such as language intersections as well.

Definition 12. *Let $\mathscr{L}$ be a sequence of finite sets of languages over a fixed alphabet and let $\mathscr{I}$ be an index set. For $i \in \mathscr{I}$, we shall denote the i-th set in the sequence by $\mathscr{L}_i$. The* longest shortest word *function $lsw_{\mathscr{L}} : \mathscr{I} \to \mathbb{N}$ is a function such that $lsw_{\mathscr{L}}(i) = max\{sw(L) \mid L \in \mathscr{L}_i, L \neq \varnothing\}$ for all $i \in \mathscr{I}$. In the cases when the language sequence is clear from the context, we shall omit the subscript and simply denote the function by $lsw(i)$.*

In the following, we shall almost exclusively use either $\mathbb{N}\backslash\{0\}$ or $(\mathbb{N}\backslash\{0\}) \times (\mathbb{N}\backslash\{0\})$ as the index set. In the former case, we can choose an alphabet Σ, a class of automata C, a measure of descriptional complexity dc and some unary operation f applied to languages (possibly the identity). Then, we can define $\mathscr{L}_i$ as $\{f(L) \mid L \subseteq \Sigma^*, \exists A \in C : dc(A) \leqslant i, L(A) = L\}$. In the latter case, the sequence $\mathscr{L}$ can be defined analogously using a binary operation: $\mathscr{L}_{i,j} = \{f(L_1, L_2) \mid L_1 \subseteq \Sigma^*, L_2 \subseteq \Sigma^*, \exists A_1 \in C : dc_1(A_1) \leqslant i, L(A_1) = L_1, \exists A_2 \in C : dc_2(A_2) \leqslant j, L(A_2) = L_2\}$.

Note that the descriptional complexity of the two automata may be different.

In the following example, we shall examine the lsw-function in the NFA case.

Example 4. *Let us consider all the languages accepted by some NFA with at most n states. In this case, $\mathscr{L}_n = \{L \mid \exists\ NFA\ A : L(A) = L, sc(A) \leqslant n\}$. The length of the*

shortest word accepted by an NFA is equal to the length of the shortest path (measured by the number of transitions) from the initial state to some of the accepting states and the accepted language is non-empty iff at least one such path exists. Clearly, the length of such path can be at most $n-1$*. To recognize the language* $\{a^{n-1}\}$*,* n *states are clearly sufficient. Thus,* $lsw(n) = n - 1$ *for all* $n \in \mathbb{N}\backslash\{0\}$*.*

The problem of shortest words accepted by NFAs was originally studied by Alpoge et al. [1].

Instead of simply measuring the length of the longest shortest accepted word, we may also be interested in the length of the longest shortest *not* accepted word. This is equivalent to considering the complements of all languages accepted by models with low enough descriptional complexity. We could also apply different unary operations or even binary operations such as intersection. These problems were studied by Alpoge et al. [1] and the key results shall be presented in the following section. However, in the settings that are more complex than the simple NFA case, it may be much more difficult to find the exact values of the lsw-function and we may only achieve some not necessarily tight lower and upper bounds.

2.2 Results for Finite Automata

Alpoge et al. [1] have studied the bounds on the length of the shortest word in the intersection of the languages accepted by finite automata, both in the deterministic and nondeterministic case. We shall now present their findings.

Proposition 8. *Let* A_1 *and* A_2 *be an* n*-state and an* m*-state NFA, respectively. Then* $sw(L(A_1) \cap L(A_2)) < mn$ *(assuming the intersection is non-empty).*

Proof. An mn-state automaton accepting $L(A_1) \cap L(A_2)$ can be constructed using the standard construction for intersection. If the intersection is not empty, the length of the shortest word in it is at most $mn - 1$, as shown in Example 4. □

Proposition 9. *For all coprime integers* n *and* m*, there exists an* n*-state NFA* A_1 *and an* m*-state NFA* A_2 *such that* $sw(L(A_1) \cap L(A_2)) = mn - 1$*.*

Proof. Let $L(A_1) = (a^n)^*a^{n-1}$ and $L(A_2) = (a^m)^*a^{m-1}$. The word a^{mn-1} is in the intersection. For every word in the intersection, there must exist non-negative integers k and k' such that the length of the word equals both $mk+m-1$ and $nk'+n-1$. This can be rewritten as $m(k+1)-1 = n(k'+1)-1$, which implies $m(k+1) = n(k'+1)$. Since $m(k+1)$ must be divisible by n but at the same time m and n are coprime, it must hold that $k+1$ is divisible by n. Thus, $k+1 \geqslant n$ and the length of any word in the intersection must be at least $mn-1$. Since the word a^{mn-1} is in the intersection, $sw(L(A_1) \cap L(A_2)) = mn - 1$. □

The technique can be applied even if m and n have a common factor, but then the shortest word contains only $lcm(n, m) - 1$ symbols. However, the authors have also shown that for binary and larger alphabets, the upper bound is achievable even if m and n are not coprime.

Proposition 10. *For all pairs of integers $n \geqslant 1$ and $m \geqslant 1$, there exists an n-state DFA A_1 and an m-state DFA A_2 such that $sw(L(A_1) \cap L(A_2)) = mn - 1$.*

Proof. Without loss of generality, let $m \geqslant n$. Let $Q_1 = \{p_0, \ldots, p_{n-1}\}$, let $Q_2 = \{q_0, \ldots, q_{m-1}\}$ and let $L(A_1) = \{w \in \{0,1\}^* \mid |w|_1 \equiv 0 \pmod n\}$. The function δ_2 is defined as follows: if $a < n-1$, $\delta_2(q_a, c) = q_{a+c}$, otherwise $\delta_2(q_a, 0) = q_{(a+1) \pmod m}$ and $\delta_2(q_a, 1) = q_0$. Let q_0 be the initial state of A_2 and let q_{m-1} be its only accepting state. An accepting computation of the automaton A_2 always consists of several cycles starting and ending in q_0 and a path from q_0 to q_{m-1}. In order to return to q_0, the automaton must first reach the state q_{n-1}, which requires reading $n-1$ 1's. Then, the initial state can be reached either by reading $(m-n+1)$ consecutive 0's or by reading at most $(m-n)$ consecutive 0's and then an additional symbol 1. Thus, for every word $w \in L(A_2)$, there exist integers $i \geqslant 0$ and $j > 0$ such that $|w|_1 = in + j(n-1)$ and $|w|_0 \geqslant j(m-n+1) - 1$. The choice of i corresponds with number of cycles containing n 1's in the accepting computation, while j corresponds with the number of paths from q_0 to q_0 containing $n-1$ 1's and the additional path which ends in q_{m-1}. Moreover, for all such i, j there exists a word $w \in L(A_2)$ such that $|w|_0 = j(m-n+1) - 1$. This bound can be reached if none of the i cycles with n 1's contains any 0. Thus, there are $j-1$ paths containing $n-1$ 1's and $m-(n-1)$ 0's and a path which ends in q_{m-1} and contains $m-(n-1)-1$ 0's.
For a word w in the intersection, it must hold that $|w|_1 = in + j(n-1), |w|_1 \equiv 0 \pmod n$ and $|w|_0 \leqslant j(m-n+1) - 1$. For the shortest such word w, we can set i to 0, since the choice of i shall not affect any of the condition. The smallest $j \geqslant 1$ such that $j(n-1) \equiv 0 \pmod n$ is n. Therefore $|w|_1 = n(n-1) = n^2 - n$ and $|w|_0 = n(m-n+1) - 1 = mn - n^2 + n - 1$ and after adding the number of 1's and 0's together, we get $mn - 1$. □

Alpoge et al. have also studied the length of the shortest word in the complement of a language accepted by an NFA and reached the following result.

Proposition 11. *Let A be an n-state NFA, let $L = L(A)$. If $L^C \neq \varnothing$, then $sw(L^C) < 2^n$ and there exists a constant c ($0 < c \leqslant 1$) and an infinite NFA family such that all the words shorter than 2^{cn} are accepted by an n-state NFA. The lower bound is achievable even for binary alphabets.*

Proof. An equivalent DFA with at most 2^n states can be constructed using the subset construction. Then, a DFA accepting L^C with the same state complexity can be

constructed by swapping the accepting and non-accepting states. The length of the shortest word in L^C corresponds with the shortest path from the initial state to some of the accepting state and by the pigeonhole principle, its length is less than 2^n. The lower bound follows from the research of regular expressions by Ellul et al. [8]. The full proof is outside of the scope of this text, but its main idea is presented in Lemma 5. □

2.3 Results for Two-Way Automata

The maximal possible length of the shortest words accepted by two-way automata was studied by Dobronravov et al. [6]. In this section, we shall present the key results from the article. First, we shall present the way in which a simple upper bound on the length of the shortest word accepted by a 2NFA was obtained in [6].

Lemma 2. *Let A be an n-state 2NFA. Then $sw(A) \leqslant \binom{2n}{n+1} - 1$.*

Proof. By Proposition 3, an NFA A' such that $L(A) = L(A')$ and $sc(A') \leqslant \binom{2n}{n+1}$ exists. The upper bound in question can be thus obtained by applying the results of the Example 4 to the automaton A'. □

While Kapoutsis [12] has shown that for every n, there exists a language accepted by an n-state 2NFA which can not be accepted by any NFA with fewer than $\binom{2n}{n+1}$ states, it can not be used to obtain a lower bound in a similar way. This is caused by the fact that every word in this language is of length 4.

Dobronravov et al. have also shown that as long as the alphabet size is not bounded, we do not need to distinguish between 2NFA and 2DFA.

Lemma 3. *For every n-state 2NFA A, there exists an n-state 2DFA A' such that $sw(A') = sw(A)$.*

Proof. For every symbol $a \in \Sigma$, there are at most $2n$ possible transitions for A - it can either move its head to the left or to the right and end up in one of the n states. Let Σ' contain the symbols from Σ marked with every possible choice of the transitions. Then $|\Sigma'| \leqslant (2n)^n|\Sigma|$. The 2DFA A' shall process the input deterministically and simulate the 2NFA A based on the marks on the symbols. Thus, the 2NFA A accepts some word over the alphabet Σ iff the 2DFA A' accepts a word of the same length over the alphabet Σ' with appropriate symbol markings. □

As a result, Dobronravov et al. limited themselves to studying deterministic two-way automata. Note that in the cases when the 2NFA A is sweeping, the 2DFA A' obtained by this construction is sweeping as well.
For unary alphabets, they reached the following lower bound.

Proposition 12. *For every $n \geqslant 2$, there exists an n-state 2DFA A such that $sw(L(A)) = g(n-1) - 1$, where g denotes the Landau's function.*

Proof. Let $p_1 + \ldots + p_k = n - 1$, let $lcm(p1, \ldots, p_k) = g(n-1)$ and let w be the input. As shown in [5], a sweeping 2DFA can verify that for every $1 \leqslant i \leqslant k$, $|w| \equiv p_i - 1 \pmod{p_i}$. Then, it holds that $sw(L(A)) = g(n-1) - 1$. If k is even, the head is on the left endmarker after the k-th pass. Because of this, one additional state is needed to reach the right endmarker and accept the input. □

Considering alphabets of size linear in the number of states allows us to achieve the following simple lower bound [6]. Note that the transition function of the 2DFA variant considered by the authors is not necessarily complete and thus the automaton may halt in some configuration.

Proposition 13. *For every odd integer $m \geqslant 1$, there exists a sweeping $2m$-state 2DFA A over an m-symbol alphabet and a word w of length $2^m - 1$ such that $L(A) = \{w\}$.*

Proof. Without loss of generality, let $\Sigma = \{a_1, \ldots, a_m\}$. The automaton A shall perform m passes. During the i-th pass, the 2DFA shall check whether every substring of symbols from $\{a_1, \ldots, a_i\}$ delimited by symbols from $\{¢, \$, a_{i+1}, \ldots, a_m\}$ contains exactly one symbol a_i. The set of states shall be $Q = \{q_j^i \mid i \in \{1, \ldots, m\}, j \in \{0, 1\}\}$ - in every state, the automaton needs to know whether it has encountered a_i in the current substring and which pass it is currently performing. The initial state shall be q_0^1 and the accepting state shall be q_1^m. Let us consider the i-th pass. During this pass, the only available states shall be q_0^i and q_1^i. If the automaton encounters some symbol a_j such that $j < i$, the state shall remain unchanged and the automaton shall continue the scan in the appropriate direction. If it encounters the symbol a_i in the state q_0^i, the state shall change to q_1^i and the computation shall continue, but if the automaton reads this symbol in the state q_1^i, the computation shall halt and the input word shall not be accepted. Finally, if the automaton encounters some symbol a_j such that $j > i$, it shall reset its state to q_0^i if the current state is q_1^i and halt otherwise. Once an endmarker is reached, it is checked once again whether the state is q_1^i. Then (unless the accepting state q_1^m has been reached) the state shall change to q_0^{i+1} and the automaton shall start a new pass in the opposite direction; the computation shall halt otherwise.
Next, it is necessary to prove that the automaton accepts a single word of length $2^m - 1$. Let us define the words $w_0, \ldots, w_m$ as follows: let $w_0 = \varepsilon$ and if $i \geqslant 1$, let $w_i = w_{i-1} a_i w_{i-1}$. Note that it always holds that $|w_i| = 2^i - 1$. We shall now prove that $L(A) = \{w_m\}$.
Let $w \in L(A)$. For $i \in \{0, \ldots, m\}$, let buc be a substring of ¢w\$ such that $b \in \{¢, a_{i+1}, \ldots, a_m\}$, $c \in \{\$, a_{i+1}, \ldots, a_m\}$ and $u \in \{a_1, \ldots, a_i\}^*$. It can be proven by induction that $u = w_i$.

The basis for $i = 0$ trivially holds since in this case the word u must be empty.
During the i-th phase, the automaton must pass through the substring buc (either from left to right or from right to left). When the substring is entered, the state must be q_0^i, since either an endmarker or a symbol with index greater than i is read. As the other end of this substring also contains such symbol, this end must be reached in the state q_1^i. Thus, u must contain the symbol a_i exactly once and can be written as xa_iz where $x, z \in \{a_1, \ldots, a_{i-1}\}$. By induction, it must hold that $x = w_{i-1} = z$ and therefore $u = w_i$.
If we set $i = m$, then it must hold that $w = w_m$ and thus $L(A) = \{w_m\}$. □

Thus, for every n, there exists an n-state 2DFA A such that $sw(L(A)) = 2^{\frac{n}{2}} - 1 \approx 1.414^n$ if $n \equiv 2 \pmod 4$ and $sw(L(A)) \geqslant 2^{\lfloor \frac{n-1}{2} \rfloor} - 1 \approx 1.414^{n-1}$ in general. The authors have also noted that the lower bound can be improved by using the same construction, but counting to 3 rather than to 2 (setting w_{i+1} as $w_i a_{i+1} w_i a_{i+1} w_i$). The detailed proof was not provided in the original article. In that case, the length of the shortest word shall be $3^{\frac{n}{3}} \approx 1.442^n$ if $n \equiv 3 \pmod 6$.
In [6], an improved lower bound was presented as well.

Proposition 14. *For every $n \geqslant 1$, there exists an n-state 2DFA A_n such that $sw(L(A_n)) \approx 1.475^n$.*

Proof. The proof can be found in [6]; we shall only present its key idea. In the article, a subclass of 2DFAs with additional restrictions on the δ-function was considered. Let A be such a 2DFA with n states over an alphabet Σ. The authors constructed a 2DFA A' with mn states and an alphabet of size $m|\Sigma|$, which worked as follows. The new alphabet Σ' consisted of m disjoint copies of the alphabet Σ, denoted by $\Sigma_1, \ldots, \Sigma_m$. In total, m passes over the input were performed. During the i-th pass, the input can be viewed as words $w_1, \ldots, w_k$ from the language $(\Sigma_1 \cup \ldots \cup \Sigma_i)^*$, delimited by ¢, \$ and symbols from $\Sigma' \backslash (\Sigma_1 \cup \ldots \cup \Sigma_i)$. If i is odd, it is possible to simulate a modified version of the 2DFA A on every block $w_1, \ldots, w_k$. This modified version ignores the symbols from $\Sigma_1 \cup \ldots \cup \Sigma_{i-1}$ and treats symbols from Σ_i as the corresponding symbols from Σ. For an even i, the simulation is done on the reverses of $w_1, \ldots, w_k$ since the main pass is performed from right to left. It was also shown that $sw(L(A')) = sw(L(A))^m - 1$. The bound 1.475^n was obtained by using a 2DFA with 5 states and shortest accepted word of length 7 as the base automaton. □

This technique requires the alphabet to be of size $\Theta(n)$. For n-state automata over an m-symbol alphabet, a lower bound approximately equal to $e^{1+o(1)}\sqrt{mn \log \frac{n}{m}}$ was obtained. The lower bound was further improved by Krymski and Okhotin in [14] to 1.626^n if the alphabet's size is exponential in n and to 1.275^n for alphabets with fixed size.

2.4 Results for Regular Expressions

The length of the shortest words described by regular expressions was studied in [8]. In this section, we shall present the results relevant for our research. The first one is concerned with the conversion of regular expression to nondeterministic automata.

Lemma 4. *Let E be a regular expression containing n alphabetical symbols. There exists an NFA A with $n+1$ states such that $L(A) = L(E)$.*

Proof. The proof is inductive and it utilizes NFAs which never return to the initial state. For atomic regular expressions, we can construct such automata with the desired number of states. Otherwise, the NFA can be obtained by applying union, concatenation or closure to the automata corresponding to the subexpressions. The details of the proof can be found in [16]. □

The second result shows that there exist regular expressions such that the length of the shortest word in the complement of the language is exponential.

Lemma 5. *For every $n \geqslant 3$, there exists a regular expression E containing $25n + 110$ symbols over a 5-symbol alphabet such that the length of the shortest word in $L(E)^C = (2^n - 1)(n + 1) + 1$.*

Proof. Let us consider the alphabet $\Sigma = \{\#, \binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}$ and an encoded sequence of numbers $1, 2, \ldots, 2^n - 2$. For $n = 3$, the encoding would be
$\#\binom{0}{0}\binom{0}{0}\binom{0}{1}\#\binom{0}{0}\binom{0}{1}\binom{1}{0}\#\binom{0}{0}\binom{1}{1}\binom{0}{1}\#\binom{0}{1}\binom{1}{0}\binom{1}{0}\#\binom{1}{1}\binom{0}{0}\binom{0}{1}\#\binom{1}{1}\binom{0}{1}\binom{1}{0}\#\binom{1}{1}\binom{1}{1}\binom{0}{1}\#$
We can construct regular expressions which check whether the conditions of syntactic correctness are violated: the first or the last $n + 2$ symbols are incorrect, a block of digits does not contain exactly n symbols, there are two consecutive symbols #, the lower binary number in a block does not equal the upper number plus one or the upper number in a block does not equal the lower number in the previous one. The expression E containing $25n + 110$ symbols can be constructed as a union of those expressions. Then, the encoded sequence is the shortest word which does not violate any of those conditions and its length is $(2^n - 1)(n + 1) + 1$. The details of the proof can be found in [8]. □

As a corollary of Lemmas 4 and 5, we can construct an NFA A with $25n + 111$ states such that $sw(L(A)^C) = (2^n - 1)(n + 1) + 1$. This way, the lower bound from Proposition 11 can be obtained.

Chapter 3

Rotating Automata

In this chapter, we shall present our results for the case of rotating automata. We shall consider unary alphabets, fixed size alphabets and alphabets of size proportional to the automaton's number of states. We shall also examine the shortest words in intersections of languages.

3.1 Basic Results

A trivial upper bound on the length of the shortest word may be obtained by simulating the RNFA by a 2NFA and applying the upper bound found by Dobronravov et al. on the resulting automaton. The transformation of RNFAs into 2NFAs is mentioned in [20].

Theorem 1. *Let A be an n-state RNFA. Then $sw(A) \leqslant \binom{2n}{n} - 1$.*

Proof. The proof is based on the following observation: we can simulate a rotating automaton by a sweeping automaton by returning to the left endmarker instead of utilizing the rotation. This can be achieved by adding a state $q_{\leftarrow}$ for every state q. In this state, the head moves to the left until the endmarker is reached, then the state changes back to q and the computation continues. This construction results in a sweeping (and therefore also direction-determinate) 2NFA with $2n$ states and the upper bound thus follows from Lemma 4. □

A simple lower bound can be obtained the same way as in the case of ordinary finite automata.

Theorem 2. *Let A be an n-state RNFA. Then $sw(A) \geqslant n - 1$.*

Proof. Let $L = \{a^k \mid k \geqslant n - 1\}$. It is trivial to construct an n-state RNFA that recognizes the language L. □

However, this bound is not tight. In the following example, we shall demonstrate that given two sufficiently large coprime numbers n and m (7 and 11 in that particular case), we can construct an RNFA with only $m+n$ states which recognizes a non-empty language L such that $sw(L) = mn - 1$.

Example 5. *Let $L = \{a^k \mid k \equiv 76 \pmod{77}\}$. Clearly, the language L is non-empty and the shortest word is of length 76. The key observation is that L may be obtained as the intersection of two regular languages. Let $L_1 = \{a^k \mid k \equiv 6 \pmod{7}\}$ and let $L_2 = \{a^k \mid k \equiv 10 \pmod{11}\}$. It can be easily seen that $L = L_1 \cap L_2$. In the case of rotating automata, we can use this to our advantage by verifying whether each of the respective languages contains the word in two separate sweeps. This allows us to construct an automaton which requires only $7 + 11 = 18$ states, but $sw(L(A)) = 76$.*

Dobronravov et al. [6] have also studied the length of accepted words over an alphabet consisting of a single symbol. Their result was presented in Proposal 12. We shall show that a similar bound can be achieved in the case of rotating automata by generalizing the technique from Example 5. It was shown by Chrobak [5] that a language similar to the one from the following theorem can be accepted by an n-state sweeping 2DFA. For words over an unary alphabet, it does not matter whether the automaton reads them from the left or from the right. Thus, the same idea can be applied to rotating automata.

Theorem 3. *Let $g(n)$ denote the Landau's function. For every n, there exists an n-state RNFA A over the alphabet $\Sigma = \{a\}$ such that $L(A) = a^{g(n)-1}(a^{g(n)})^*$. Thus, $sw(L(A)) = g(n) - 1$.*

Proof. Let $p_1, \ldots, p_k$ be numbers such that $lcm(p_1, \ldots, p_k) = g(n)$. It follows from the definition of the Landau's function that such numbers have to exist. The key observation is that for every p_i in the partition of n, we can construct a one-way automaton A_i which accepts the language $L_i = \{a^m \mid m \equiv p_i - 1 \pmod{p_i}\}$. To accept such language, p_i states are clearly sufficient. In the $i-$th pass of the rotating automaton, we can verify that the language L_i contains the input word. If the right endmarker is reached in an accepting state of the one-way automaton, the rotating automaton returns to the left endmarker and performs the next pass. Otherwise, the computation is halted and the input is not accepted. Finally, the input is accepted if the right endmarker is reached in an accepting state during the last pass (and therefore during all of the previous passes as well). It follows from the construction of the automaton A that $L(A) = \cap_{i=1}^{k} L_i$.
Now, let w be a word from the language L. Its length is therefore $l = g(n) - 1 + mg(n)$ for some $m \geqslant 0$. For every p_i, it holds that $l \equiv p_i - 1 \pmod{p_i}$, since $l + 1$ is divisible by $g(n)$ and therefore also by p_i. Thus, $w \in L(A)$.

If w is a word from the language $L(A)$ of length l, it must hold that $l \equiv p_i - 1 \pmod{p_i}$ for every i. As a result, $l + 1 \equiv 0 \pmod{p_i}$ for every i as well. Therefore, $l + 1$ must be some multiple of $g(n)$ and $l \equiv g(n) - 1 \pmod{g(n)}$. We have shown that $L(A) = L$. It follows that the length of the shortest word in $L(A)$ is $g(n) - 1$. □

Note that in the case of two-way automata, the proven bound was only $g(n-1)-1$. This is caused by the fact that the accepting states of a two-way automaton are effective only at the right endmarker. In the cases when the partition of n contains an even number of numbers, the computation ends at the left endmarker and an additional state is needed to move the head to the end of the input.

3.2 Results for Larger Alphabets

Let us consider the case when the alphabet is not unary, but still bounded by some constant k independent from the state complexity. In the case of two-way automata, the lower bound was improved in [6] using the technique from Proposal 14. However, we can not apply this exact technique to rotating automata. The reason is that while a 2DFA may treat chosen symbols as endmarkers and simulate a different 2DFA on a part of the input, an RNFA can return to the left endmarker only after reaching the real right endmarker. Instead of attempting to find an analogous technique for RNFAs directly, we shall improve the lower bound in several steps.
We can use the property of Landau's function from Lemma 1 and Corollary 1 to achieve a bound that is better than the one from Theorem 3 for sufficiently large n - it is possible to increase the length of the shortest word k-fold at the cost of k additional states. To our knowledge, this technique was not used by other automata theory researchers before. We shall demonstrate the idea in the following example.

Example 6. *The RNFA shall check whether the input is of the form $(ab^k)^*$ during the first pass and count the number of symbols a during the later passes the same way as in Theorem 3. The diagram of such automaton is given in Figure 3.1.*

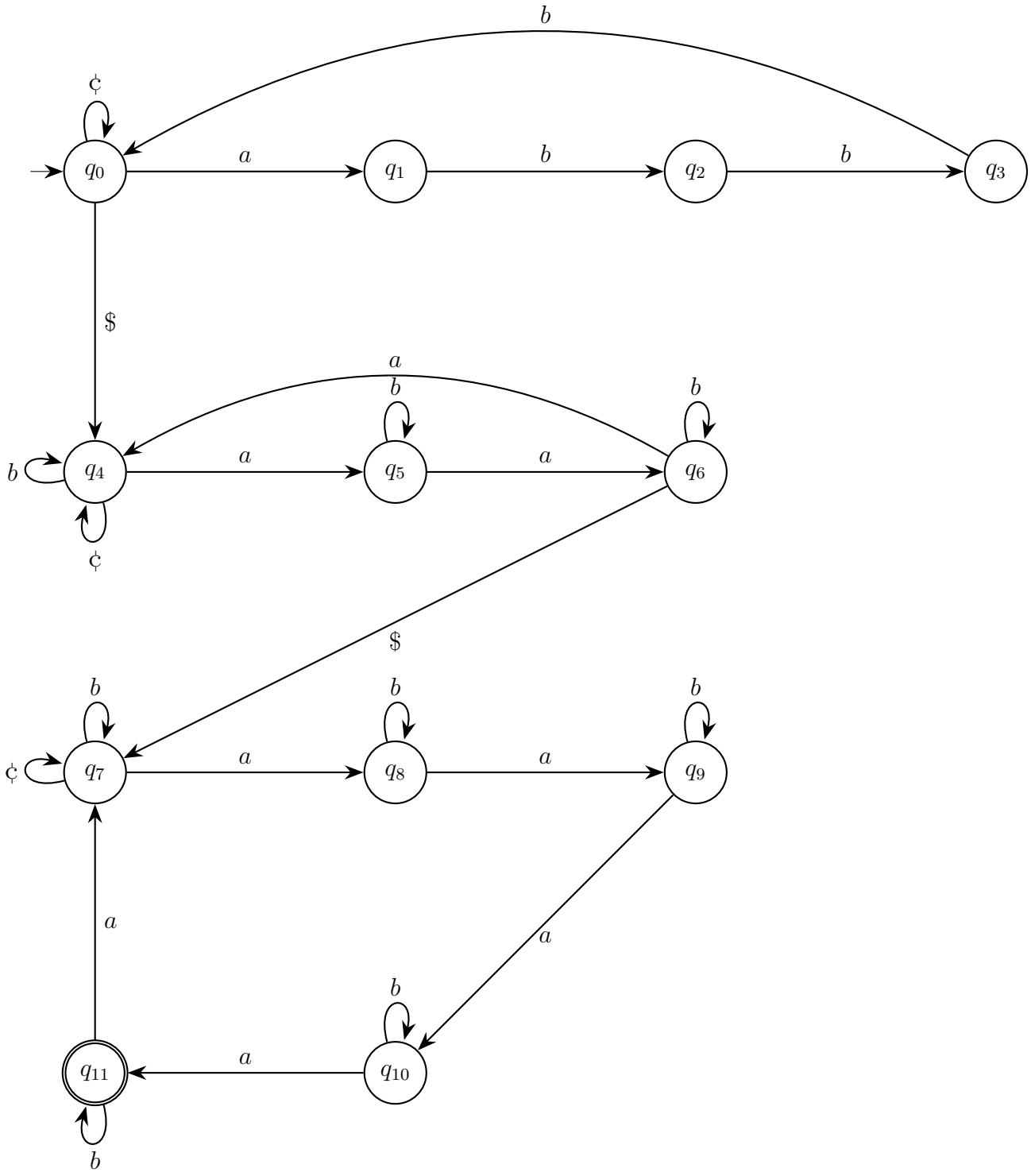

Figure 3.1: The RNFA from Example 6

In this case, the accepted language is $L = \{(ab^3)^*\} \cap \{w \mid |w|_a \equiv 14 \pmod{15}\}$. *The shortest accepted word is therefore* $(ab^3)^{14}$ *and* $sw(L) = 56$.

We can generalize the demonstrated technique in the following theorem.

Theorem 4. *For every pair of integers* n *and* k *such that* $k < n$, *there exists an RNFA*

A such that $sc(A) = n$ *and* $L(A) = (ab^{k-1})^* \cap \{w \mid |w|_a \equiv g(n-k)-1 \pmod{g(n-k)}\}$. *It holds that* $sw(L(A)) = (g(n-k)-1)k$ *and that* $\lim_{n\to\infty} \frac{(g(n-k)-1)k}{(g(n)-1)k} = 1$.

Proof. The construction is similar to the one in Example 6. During the first pass, we can use k states to verify that the input is of the form $(ab^{k-1})^*$. Then, the remaining $n-k$ states can be used to verify the number of symbols a is congruent with $g(n-k)-1$ modulo $g(n-k)$. The shortest accepted word contains $g(n-k)-1$ symbols a, each followed by $k-1$ symbols b. It is therefore of length $(g(n-k)-1)k$. The fact that $\lim_{n\to\infty} \frac{(g(n-k)-1)k}{(g(n)-1)k} = 1$ follows from Corollary 1. □

Note that a binary alphabet is sufficient to achieve an arbitrarily large constant factor. While this technique is applicable to two-way automata as well, it does not result in an improvement over the results from [6] and [14].

This technique can also be viewed in the following way: we are given two languages over disjoint alphabets Σ_1 and Σ_2. In our case, $L_1 = \{a^m \mid m \equiv 14 \pmod{15}\}$ and $L_2 = \{b^3\}$. We can construct a new language which is the same as L_1 if symbols from Σ_2 are omitted, but every symbol from Σ_1 is followed by a word from L_2. Formally, let h be a homomorphism such that $h(c) = c$ if $c \in \Sigma_1$ and $h(c) = \varepsilon$ otherwise. Then the new language is $h^{-1}(L_1) \cap \{(cw)^* \mid c \in \Sigma_1, w \in L_2\}$. Could the bound for binary alphabets be improved further if we chose a different language L_2? For simplicity, let us consider some integer n and languages $L_1 = \{a^m \mid m \equiv g(n)-1 \pmod{g(n)}\}$ and $L_2 = \{b^m \mid m \equiv g(n)-1 \pmod{g(n)}\}$. The new RNFA shall work in three phases:

1. Verify the input is of the form $(ab^+)^*$.

2. Verify the number of symbols a is correct. This can be done by computing the remainder modulo each number in the optimal partition of n and ignoring the symbols b as in Figure 3.2.

3. Verify the number of symbols b after every symbol a is correct. Let $p_1 + \ldots + p_k$ be the optimal partition of n. For each number in this partition, one pass is performed. During the i-th pass, we need to verify that for every block of symbols b, the remainder of its length modulo p_i is $p_i - 1$. If the symbol a or the right endmarker is encountered in an incorrect state, the computation is halted. However, the input is accepted if the length of each block is correct.

The shortest accepted word is of length $g(n)(g(n)-1)$. The automaton A_n in the case when $n = 8$ is shown in the Figure 3.2.

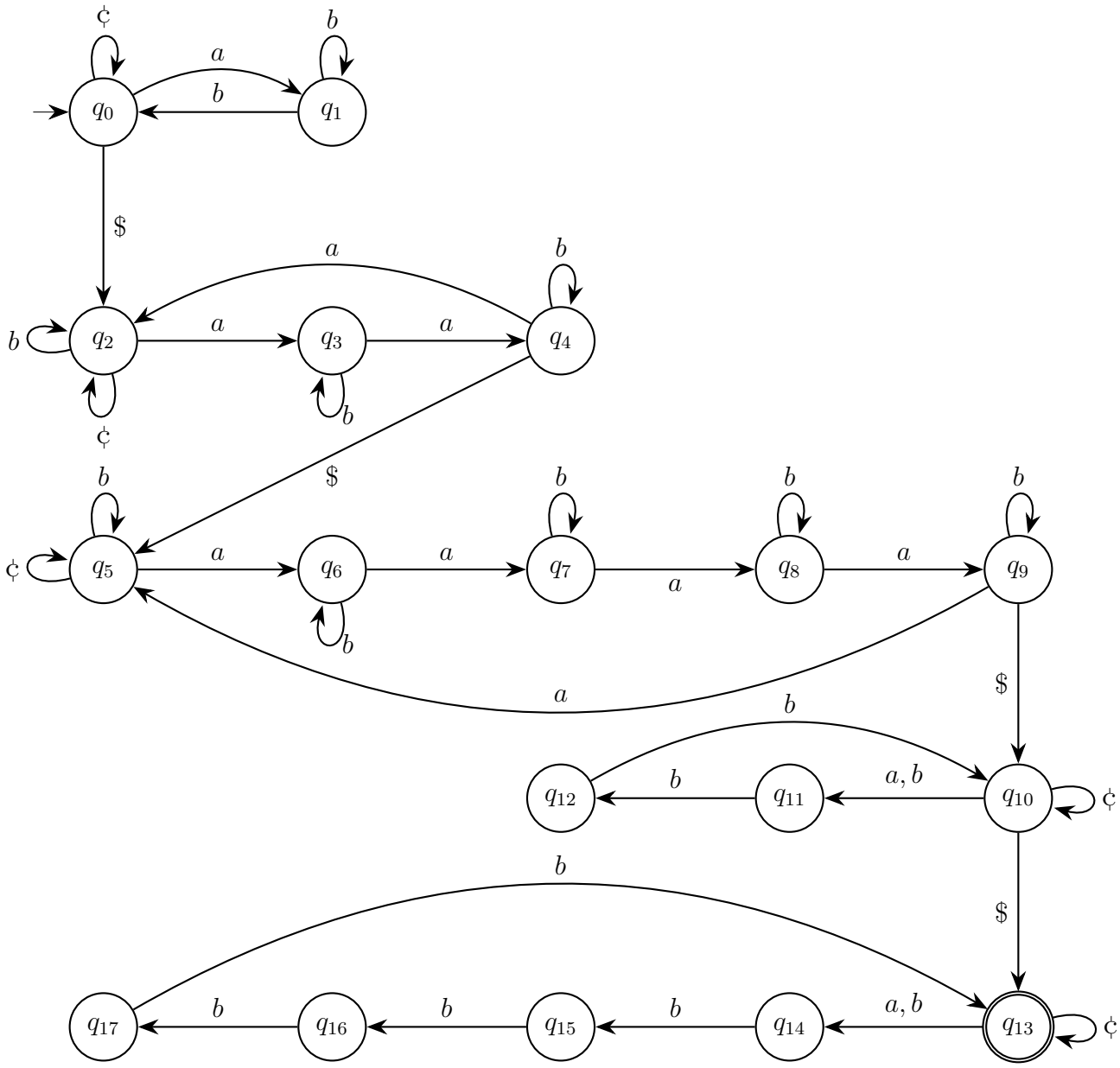

Figure 3.2: The RNFA A_8

However, the given construction requires $2n + 2$ states. Let us consider an RNFA A'_n which only calculates the remainder modulo Landau's function as in Theorem 3. In that case, $sw(L(A'_n)) = g(2n + 2) - 1 \approx e^{\sqrt{(2n+2)\ln(2n+2)}}$. Using the same approximation, it holds that $sw(L(A_n)) \approx e^{2\sqrt{n \ln n}}$. It can be shown that for $n \geqslant 5$, $e^{2\sqrt{n \ln n}} > e^{\sqrt{(2n+2)\ln(2n+2)}}$. The new bound is therefore an improvement indeed.
Let us consider the case where the length of every block of symbols b is congruent with $g(m) - 1$ modulo $g(m)$ for some integer m, not necessarily equal to n. In the following theorem, we shall generalize the previous result.

Theorem 5. *For every pair of integers n and m, there exists a RNFA A over a binary alphabet such that $sc(A) = n + m$ and $sw(L(A)) = g(m)(g(n) - 1)$.*

Proof. Our goal is to accept the language $L = \{w \mid w \in \{a,b\}^*, |w|_a \equiv g(n) - 1 \pmod{g(n)}\} \cap ((a(b^{g(m)})^*b^{g(m)-1})^* \cup (b^{g(m)})^*b^{g(m)-1}(a(b^{g(m)})^*b^{g(m)-1})^*)$. We can accept the languages in the intersection using n and m states, respectively. The shortest word in the resulting language is $(ab^{g(m)-1})^{g(n)-1}$. □

Note that the resulting automaton works correctly due to the fact that during each pass, the state in which each block of symbols b is entered is uniquely determined and the same number of passes is necessary and sufficient to recognize every word from the language L_2. Thus, this technique is not necessarily applicable to any choice of languages. Still, a lower bound approximately equal to $(g(n))^k$ can be achieved for k-symbol alphabets. To prove this, we shall use the following lemma.

Lemma 6. *Let L_1 and L_2 be languages over an alphabet Σ, let c be a symbol such that $c \notin \Sigma$ and let $L_1 \cap L_2 = L$. Then $(Lc)^* = (L_1c)^* \cap (L_2c)^*$.*

Proof. Let w be a word from $(L_1c)^* \cap (L_2c)^*$. Then, w can be rewritten as $v_1cv_2c \dots v_nc$ where for every i, it holds that $v_i \in \Sigma^*$. Since $c \notin \Sigma$, it must also hold that $v_i \in L_1$ and $v_i \in L_2$ and thus $v_i \in L_1 \cap L_2$. Therefore, $(L_1c)^* \cap (L_2c)^* \subseteq (Lc)^*$.
If w is a word from $(Lc)^*$ instead, it can be rewritten as $v_1cv_2c \dots v_nc$. Since $v_i \in L = L_1 \cap L_2$, it also holds that $v_i \in L_1$ and $v_i \in L_2$. Thus, $w \in (L_1c)^*$ and $w \in (L_2c)^*$ and therefore $(Lc)^* \subseteq (L_1c)^* \cap (L_2c)^*$. □

Furthermore, this lemma can be trivially generalized to intersections of more than two languages. Note the importance of the symbol $c \notin \Sigma$. In general, it does not hold that $(L_1 \cap L_2)^* = L_1^* \cap L_2^*$ - consider the case where $L_1 = \{a\}$ and $L_2 = \{aa\}$. Then, $(L_1 \cap L_2) = \varnothing$, but $aa \in L_1^* \cap L_2^*$.

Theorem 6. *Let n be a fixed integer. For every integer i such that $1 \leqslant i \leqslant n$, let $\Sigma_i = \{a_1, \dots, a_i\}$. Let $L_1 = \{a_1^k \mid k \equiv g(n) - 1 \pmod{g(n)}\}$ and for every i such that $2 \leqslant i \leqslant n$, let $L_{i+1} = (L_ia_{i+1})^*L_i \cap \{w \mid w \in \Sigma_{i+1}^*, |w|_{a_{i+1}} \equiv g(n) - 1 \pmod{g(n)}\}$. Let us set $x = g(n)$. Then for every i such that $1 \leqslant i \leqslant n$, $sw(L_i) = x^i - 1$ and there exists an RNFA with in states accepting L_i.*

Proof. We shall use mathematical induction to prove the first part. Clearly, $sw(L_1) = g(n) - 1 = x - 1$ and thus the basis step holds. For the induction step, we may assume that $sw(L_i) = x^i - 1$. By the definition of L_{i+1}, we can split any accepted word w into words from L_i separated by individual symbols a_{i+1}. Assuming w is the shortest word in L_{i+1}, it must contain exactly $x - 1$ symbols a_{i+1}, since a shorter accepted word would exist otherwise. Thus, w contains exactly x words from L_i. Each of those words must be of length $x^i - 1$, or else we could replace it by a shorter word from L_i and obtain a word from L_{i+1} that is shorter than w. Therefore, the word w contains $x(x^i - 1) + x - 1 = x^{i+1} - 1$ symbols.

In the second part of the proof, we need to show that each language L_i can be accepted by an RNFA with no more than in states. We shall prove this by mathematical induction as well. In the case of the language L_1, n states are clearly sufficient. Thus, the basis holds. Moreover, we can represent the language L_1 as the intersection of several simpler regular languages which can be accepted by NFAs with n states in total, with each NFA computing the remainder modulo one of the numbers in the optimal partition of n. For the induction step, we may assume that $L_i = L'_1 \cap \ldots \cap L'_m$ and that the languages $L'_1, \ldots, L'_m$ can be accepted by NFAs with in states in total. For each j such that $1 \leqslant j \leqslant m$, let $A_j = (Q, \Sigma_i, \delta, q_0, F)$ be the NFA accepting the language L'_j. We can modify this automaton by changing its alphabet to Σ_{i+1} and adding the transitions $q_0 \in \delta(q_F, a_{i+1})$ for every $q_F \in F$. The modified automaton shall accept the language $(L'_j a_{i+1})^* L'_j$. This holds since the symbol a_{i+1} can only be read in an accepting state and the initial state has to be reached immediately afterwards. Note that the automaton's number of states remained the same.
It follows from Lemma 6 that $(L_i a_{i+1})^* L_i = (L'_1 a_{i+1})^* L'_1 \cap \ldots \cap (L'_m a_{i+1})^* L'_m$. The language L_{i+1} was defined as $(L_i a_{i+1})^* L_i \cap \{w \mid w \in \Sigma_{i+1}^*, |w|_{a_{i+1}} \equiv g(n) - 1 \pmod{g(n)}\}$. We have already shown that the language $(L_i a_{i+1})^* L_i$ is the intersection of languages accepted by NFAs with in states in total. The language $\{w \mid w \in \Sigma_{i+1}^*, |w|_{a_{i+1}} \equiv g(n) - 1 \pmod{g(n)}\}$ can be represented as the intersection of several regular languages, similarly to L_1. The difference is that the NFAs count the symbols a_{i+1} and ignore the remaining ones. In total, these NFAs require n states. Thus, we can represent the language L_{i+1} as the intersection of languages accepted by NFAs $A_1, \ldots A_k$ with $in + n = (i+1)n$ states in total. We can simulate these NFAs by an RNFA A. During the j-th pass, we shall simulate the NFA $A_j = (Q_j, \Sigma_{i+1}, \delta, q_0, F_j)$. If the right endmarker is reached in a state from F_j, the RNFA shall accept the input if $j = k$ and return to the left endmarker and simulate the next NFA if $j < k$. If the right endmarker is reached in a state from $Q_j \backslash F_j$, the RNFA shall halt and the input is rejected. To construct the RNFA A, $(i+1)n$ states are therefore sufficient. □

We have ultimately reached a bound similar to the improved one from [6]. The key difference was that we could not simulate a different RNFA on a part of the input tape. However, we were able to represent the used languages as the intersections of simpler regular languages instead. This allowed us to process the blocks "simultaneously".

Next, we shall study the length of words accepted by rotating automata without a strict bound on the alphabet size. It turns out that the idea used for 2DFAs in [6] and presented as Proposition 13 is applicable to RNFAs as well.

Theorem 7. *For every $m \geqslant 1$, there exists an $2m-$state RNFA A such that $L(A) = \{w\}$ and $|w| = 2^m - 1$.*

Proof. The proof is almost identical to the proof for such lower bound in the two-way automata case. Let $\Sigma = \{a_1, \ldots, a_m\}$. Let $Q = \{q_j^{(i)} \mid i \in \{1, \ldots, m\}, j \in \{1, 2\}\}$. The transition function is defined as follows:

$$
\begin{array}{ll}
\delta(q, ¢) = \{q\} & \text{for every } q \in Q \\
\delta(q_0^{(i)}, a_i) = \{q_1^{(i)}\} & \\
\delta(q_1^{(i)}, a_{i+k}) = \{q_0^{(i)}\} & \text{for every } k \text{ such that } k > 0 \text{ and } i + k \leqslant m \\
\delta(q_1^{(i)}, \$) = \{q_0^{(i+1)}\} & \text{for every } i < m \\
\delta(q_j^{(i)}, a_{i-k}) = \{q_j^{(i)}\} & \text{for every } k \text{ such that } k > 0 \text{ and } i - k \geqslant 1 \text{ and for every } j \in \{1, 2\}
\end{array}
$$

The transition function returns an empty set for all the remaining combinations of states and symbols. The only accepting state shall be $q_1^{(m)}$. During the i-th pass, the automaton checks that whenever a symbol a_{i+k} for some $k > 0$ or the right endmarker is encountered, exactly one symbol a_i was read since the last greater symbol was read. Let $w_0 = \varepsilon$ and let $w_i = w_{i-1} a_i w_{i-1}$ for all $1 \leqslant i \leqslant m$. We can use induction to show that $|w_i| = 2^i - 1$. The base case for w_0 clearly holds. Now, if $|w_k| = 2^k - 1$, then $|w_{k+1}| = 2|w_k| + 1 = 2^{k+1} - 2 + 1 = 2^{k+1} - 1$. Next, we shall prove that $L(A) = \{w_m\}$. First, we shall show that either $L(A) = \varnothing$ or $L(A) = \{w_m\}$.
Let $w = cud$ be a substring of some word from $L(A)$ such that $c \in \{¢, a_{i+1}, \ldots, a_m\}$, $d \in \{\$, a_{i+1}, \ldots, a_m\}$ and $u \in \{a_1, \ldots, a_i\}^*$. It can be shown by induction that $u = w_i$. This clearly holds when $i = 0$. If the computation is accepting, both the symbol c and the symbol d must be entered in the state $q_1^{(i)}$ during the i-th pass - otherwise, the transition function would return an empty set of states and the computation would halt without accepting. This means that u must contain exactly one symbol a_i, which allows us to split u into two substrings over $\{a_1, \ldots, a_{i-1}\}$ separated by the symbol a_i and apply the induction hypothesis to them. Since $u = w_{i-1} a_i w_{i-1}$, it is equal to w_i by its definition. If we set $i = m$, then c and d must be the left and right endmarker, respectively. This implies that w_m is the only accepted word. Now, we need to show that $w_m \in L(A)$. For any i, we can use induction to show that w_m can be split into several words w_i, separated by symbols greater than a_i. Since we w_i contains exactly one symbol a_i, the word w_m shall be accepted. Thus, $L(A) = \{w_m\}$ and $sw(L(A)) = 2^m - 1$. □

Note that the analogous theorem in [6] additionally required m to be odd. The reason for this is technical: a two-way automaton may accept the input only if its head is on the right endmarker. In total, m passes are performed and after an even number of passes, the head is on the left endmarker instead. Since rotating automata perform each of the passes from left to right, m may be even in our case. As in the 2NFA case,

this yields a lower bound $2^{\lfloor \frac{n}{2} \rfloor} - 1 \approx 1.41^n$ for an n-state RNFA. As in [6], this bound can be slightly improved.

Theorem 8. *For every $m \geqslant 1$, there exists an $3m-$state RNFA A such that $L(A) = \{w\}$ and $|w| = 3^m - 1$.*

Proof. The idea of the proof is the same as in Theorem 7, but we set w_i as $w_{i-1}a_iw_{i-1}a_iw_{i-1}$ and modify the δ-function accordingly. □

For n-state automata, the new bound is $3^{\lfloor \frac{n}{3} \rfloor} - 1 \approx 1.44^n$.

3.3 Intersection

Up to this point, we were interested in the length of the shortest word accepted by a single automaton. In this section, we shall study a related problem: if A_1 is some n-state RNFA and A_2 is some m-state RNFA, how long may the shortest word in $L(A_1) \cap L(A_2)$ be? Note that if the two automata use the same alphabet Σ and $L(A_2) = \Sigma^*$, then $L(A_1) \cap L(A_2) = L(A_1)$ and $sw(L(A_1) \cap L(A_2)) = sw(L(A_1)$. Is it possible to achieve a lower bound that grows relative to both n and m?
Let us consider the languages $L_1 = (a_1^{g(n)})^* a_1^{g(n)-1}$ and $L_2 = (L_1a_2)^*L_1 \cap \{w \in \{a_1, a_2\}^* \mid |w|_{a_2} \equiv g(n) - 1 \pmod{g(n)}$ from Theorem 6. As shown in the theorem's proof, we can construct an n-state RNFA A_1 such that $L(A_1) = (L_1a_2)^*L_1$ and an n-state RNFA A_2 such that $L(A_2) = \{w \in \{a_1, a_2\}^* \mid |w|_{a_1} \equiv g(n) - 1 \pmod{g(n)}$. Thus, $L_2 = L(A_1) \cap L(A_2)$ and $sw(L_2) = g(n)^2 - 1$. We can generalize this result to the cases where the automata have a different number of states.

Theorem 9. *For every pair of integers n and m, there exists an n-state RNFA A_1 and an m-state RNFA A_2 such that $sw(L(A_1) \cap L(A_2)) = g(m)g(n) - 1$.*

Proof. Let $L_1 = (a_1^{g(n)})^* a_1^{g(n)-1}$ and let $L_2 = (L_1a_2)^*L_1 \cap \{w \in \{a_1, a_2\}^* \mid |w|_{a_2} \equiv g(m) - 1 \pmod{g(m)}$. Then, $sw(L_1) = g(n) - 1$ and $sw(L_2) = g(m) - 1 + g(m)(g(n) - 1) = g(m)g(n) - 1$. The language $\{w \in \{a_1, a_2\}^* \mid |w|_{a_2} \equiv g(m) - 1 \pmod{g(m)}$ can be accepted by an m-state RNFA. Since the language L_1 can be represented as the intersection of languages accepted by NFAs with n states in total, the language $(L_1a_2)^*L_1$ can be accepted by an n-state RNFA. □

If the size of the alphabet is not restricted, this technique can also be applied to the languages from the Theorem 7 and their modifications from Theorem 8.

Theorem 10. *For every pair of integers n and m, there exists an $3n$-state RNFA A_1 and an RNFA A_2 with $3m$ states such that $sw(L(A_1) \cap L(A_2)) = 3^{n+m} - 1$.*

Proof. Two disjoint alphabets Σ_1 and Σ_2 are considered. We shall define a homomorphism h_2 such that $h_2(c) = \varepsilon$ for every $c \in \Sigma_1$ and $h_2(c) = c$ for every $c \in \Sigma_2$. Due to Theorem 8, there exists a singleton language $L_1 \in \Sigma_1^*$ such that $sw(L_1) = 3^n - 1$, accepted by an RNFA with $3n$ states. We can accept the language $h_2^{-1}(L_1)$ by an RNFA A_1 with $3n$ states as well.
There also exists a singleton language $L_2 \in \Sigma_2^*$ such that $sw(L_2) = 3^m - 1$, also due to Theorem 8. Moreover, we can represent L_2 as the intersection of languages accepted by NFAs with $3m$ states in total. Therefore, we can accept the language $(L_2\Sigma_1)^*L_2$ by an RNFA A_2 with $3m$ states. Then, $sw(L(A_1) \cap L(A_2)) = (3^n)(3^m - 1) + 3^n - 1 = 3^{n+m} - 1$. □

If we set $3n = n'$ and $3m = m'$, we can also construct a single RNFA $A_{n,m}$ with $n' + m'$ states such that $L(A_{n.m}) = L(A_1) \cap L(A_2)$. Then, $sw(L(A_{n,m})) \approx 1.44^{n'+m'}$, which is not an improvement compared to Theorem 8.

Chapter 4

Alternating Finite Automata

In this chapter, we shall present our results for the case of alternating automata. As in the previous chapter, we shall consider both unary and larger alphabets. Afterwards, we shall compare these results with those achieved for rotating automata. Lastly, we shall study the length of the shortest words in intersections and complements of languages recognized by alternating automata.

4.1 Basic Results

First, we shall show a simple way to construct an AFA that accepts the intersection of two languages accepted by two AFAs.

Lemma 7. *Let A_1 and A_2 be alternating automata with n and m states, respectively. There exists an alternating automaton A_3 with $n + m + 1$ states such that $L(A_3) = L(A_1) \cap L(A_2)$.*

Proof. The automaton A_3 shall consist of copies of A_1 and A_2 and an initial non-accepting universal state. From this state, transitions on ε shall lead to the initial states of both A_1 and A_2. Since the new initial state is universal, it follows that A_3 accepts the input iff both A_1 and A_2 accept it. □

It can be easily seen that the same construction can be used to describe the intersection of an arbitrary number of languages with the use of only one additional state.

The Landau's function appeared in the lower bounds for two-way and rotating automata over unary alphabets. We shall show that an analogous bound can be achieved in the case of alternating automata.

Theorem 11. *Let $n \geqslant 6$. There exists an AFA A over a unary alphabet with n states such that $sw(A) = g(n-1) - 1$, where g denotes the Landau's function.*

Proof. The idea of the construction is similar to the ones in the settings of two-way and rotating automata. However, instead of verifying that the input belongs to the intersection by reading it multiple times, an AFA can do so parallelly. In Figure 4.1, we show the construction for $n = 9$. It holds that $g(8) = 15$, which can be achieved as the least common multiple of 3 and 5. In general, let $p_1 + \ldots + p_k$ be the partition of $n - 1$ for which the maximum possible least common multiple is achieved and let $L_i = \{a^k \mid k \equiv p_i - 1 \pmod{p_i}\}$. It follows from Lemma 7 that we can construct an AFA accepting the language $\cap_{i=1}^{k} L_i$ with n states.

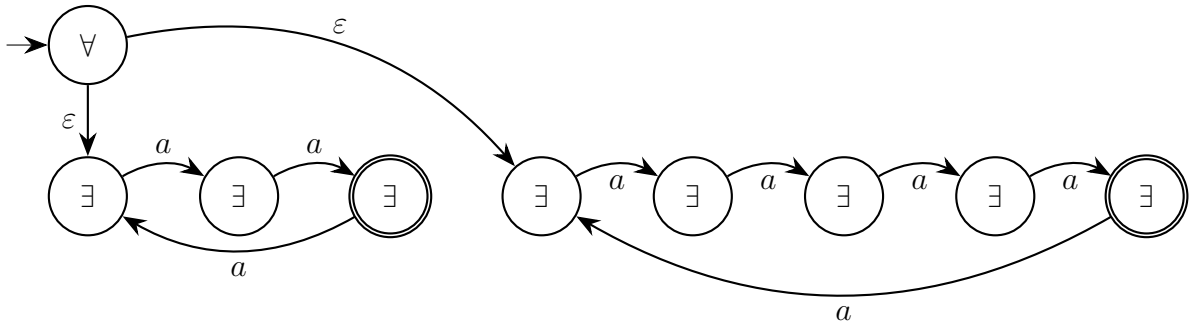

Figure 4.1: The AFA accepting the language $L = \{a^{3n-1} \mid n \geqslant 1\} \cap \{a^{5n-1} \mid n \geqslant 1\}$

□

Note that the Landau's function is asymptoticaly equal to $e^{(1+o(1))\sqrt{n \ln n}}$.

4.2 Results for Larger Alphabets

A lower bound similar to those achieved for two-way and rotating automata can be obtained over alphabets of linear size as well. The technique is based on the simple lower bound described in [6] and Proposition 14.

Theorem 12. *For every $m \geqslant 1$, there exists an AFA A with $3m + 3$ states such that $L(A) = \{w\}$ and $|w| = 3^m$.*

Proof. Let $\Sigma = \{a_1, \ldots, a_m\} \cup \{b\}$. The symbol b serves as an endmarker. Let $w_0 = \varepsilon$ and let $w_i = w_{i-1}a_i w_{i-1} a_i w_{i-1}$. The AFA A shall accept the intersection of m languages. The i-th sub-automaton checks whether whenever a symbol greater than a_i or b is encountered, exactly two symbols a_i were read since the last such symbol. Each sub-automaton needs 3 states to count the symbols. Apart from those $3m$ states and the initial state, we also require one accepting state which can be entered if the "endmarker" b is read in a correct state and a trash state which is entered if any symbol is read after b. The accepting and trash states can be shared and therefore $3m + 3$ states are used

in total, including the initial one. The proof that $L(A) = \{w_m b\}$ is analogous to the ones in [6] and Theorem 7. It can be proved by induction that the length of w_i is equal to $3^i - 1$, since it holds that the length of w_0 is $0 = 3^0 - 1$ and then the length of w_i is $3(3^{i-1} - 1) - 1 = 3^i - 1$. The length of the shortest and only accepted word $w_m b$ is thus exactly 3^m. □

For an n-state AFA, Theorem 12 yields a bound equal to $3^{\lfloor \frac{n-3}{3} \rfloor}$, which is approximately 1.44^{n-3}. Note that this technique can not be generalized for counting to numbers larger than 3, since the increased state complexity would overweigh the greater length of the only and thus shortest accepted word. In general, the bound would be $x^{\lfloor \frac{n-3}{x} \rfloor}$ if each word contained x copies of the previous one. However, the maximum of the function $x^{\frac{1}{x}}$ is reached when $x = e$ and the function is decreasing on the interval (e, ∞).

We are also interested in the following problem: how long can the shortest accepted word be if the alphabet is bounded by some constant, but not unary? In the next theorem, we shall show that we can achieve a better lower bound than in the case of unary alphabets. However, this result does not depend on alternation between universal and existential states; it suffices that succint AFAs can accept the complements of languages accepted by NFAs.

Theorem 13. *Let Σ be an alphabet with 5 symbols. There exists an AFA A over the alphabet Σ with $25n + 112$ states such that the shortest word accepted by A is of length $(2^n - 1)(n + 1) + 1$.*

Proof. Based on Lemma 5, there exists a regular expression E with $25n + 110$ symbols over the alphabet Σ such that the shortest word not recognized by E is of length $(2^n - 1)(n + 1) + 1$. We can construct an NFA which accepts the language $L(E)$ with $25n + 111$ states [17]. Then, we can convert the NFA into an AFA by making every of its states existential. Finally, we can convert the AFA into another AFA with state complexity equal to $25n + 112$ which accepts the complement of $L(E)$. The final conversion can be achieved by adding an additional trash state to assure the AFA always reads the entire input, exchaning existential and universal states and exchanging accepting and non-accepting states. □

For an n-state AFA, the bound is thus approximately $\frac{(n-112)1.028^n}{25}$. For sufficiently large n, the bound is higher than the one achieved for unary alphabets. We attempted to improve the lower bound for larger (but still bounded by some constant) alphabets by generalizing the technique from [8]. The idea was to use k-ary encoding instead of binary encoding. The resulting alphabet would therefore be of size $k^2 + 1$. However, we found out that the number of states of the resulting AFA was too high and the

shortest word would be longer if the additional symbols were ignored. The reason is that the regular expression on which the AFA is based on is in fact a union of seven other expressions and the descriptional complexity of some of them grows for larger alphabets. If the alphabet $\mathbb{Z}_k^2 \cup \{\#\}$ was used instead of $\mathbb{Z}_2^2 \cup \{\#\}$, the complexity of the resulting expression would be $\frac{1}{2}(k^3(2n+2) + k^2(4n+39) + k(2n+3) + 14(n+3))$ and the length of the shortest word in the complement would be $(k^n - 1)(n+1) + 1$.

To obtain a simple upper bound on the length of the shortest word, we can convert the AFA into an NFA and apply the bound from [1] to the resulting automaton.

Theorem 14. *For every AFA A with n states accepting a non-empty language, it holds that $sw(A) \leqslant 2^n - 1$.*

Proof. This upper bound follows from the fact that an equivalent NFA with 2^n states exists. If the accepted language is non-empty, the shortest word can not be longer than $2^n - 1$. □

In the case of nondeterministic finite automata, the length of the shortest accepted word had to be linear, but the length of the shortest word not accepted by the automaton could be exponential with regard to the state complexity. We have already shown that in the AFA case even the shortest word can be of length $e^{(1+o(1))\sqrt{n \ln n}}$. However, it turns out that if the complement is non-empty, the length of its shortest word is still only exponential in the worst case.

Theorem 15. *Let A be an AFA with n states. If $L(A)^C$ is non-empty, then $sw(L(A)^C) \leqslant 2^n - 1$.*

Proof. We can construct an n-state AFA accepting the language $L(A)^C$ by making every existential state of the automaton A universal and vice versa and by making its accepting states non-accepting and vice versa. The upper bound then follows from the previous theorem. □

4.3 Similarity with RNFAs

We have shown that similar bounds can be achieved for both rotating and alternating automata, despite the differences between these models. The reason is that both of these models can read the input multiple times, either sequentially or parallelly. This results in efficient (compared to NFAs) constructions for intersections of multiple languages and allows us to accept certain languages with relatively few states. The following two theorems describe the sufficient conditions for both unary and larger alphabets.

Theorem 16. *Let M be a formal model and let dc be its measure of descriptional complexity. Let us assume that for every DFA A_{DFA}, there exists an instance A_M from M such that $L(A_M) = L(A_{DFA})$ and $dc(A_M) \leqslant sc(A_{DFA})$. Let $A_1, \ldots, A_k$ be a sequence of the model's instances and let us set $\sum_{i=1}^{k} dc(A_i) = n$. Let us assume that there exists a constant c such that for any choice of $A_1, \ldots, A_k$, there exists an instance A' such that $L(A') = \bigcap_{i=1}^{k} L(A_i)$ and $dc(A') = n + c$. Then for every integer m, there also exists an instance A such that $dc(A) = m + c$ and $L(A) = (a^{g(m)})^* a^{g(m)-1}$.*

Proof. Let $p_1 + \ldots + p_k = m$ be the optimal partition of m from Landau's function. For every $1 \leqslant i \leqslant k$, we can construct an instance A_i such that $dc(A_i) = p_i$ and $L(A_i) = (a^{p_i})^* a^{p_i - 1}$. Then, we can construct the instance A such that $L(A) = \bigcap_{i=1}^{k} L(A_i) = (a^{g(m)})^* a^{g(m)-1}$ and $dc(A) = m + c$. □

Theorem 17. *Let M be a formal model and let dc be its measure of descriptional complexity. Let us assume that for every DFA A_{DFA}, there exists an instance A_M from M such that $L(A_M) = L(A_{DFA})$ and $dc(A_M) \leqslant sc(A_{DFA})$. Let $A_1, \ldots, A_k$ be a sequence of the model's instances and let us set $\sum_{i=1}^{k} dc(A_i) = n$. Let us assume that there exists a constant c such that for any choice of $A_1, \ldots, A_k$, there exists an instance A' such that $L(A') = \bigcap_{i=1}^{k} L(A_i)$ and $dc(A') = n + c$. Then for every integer m, there also exists an instance A such that $dc(A) = m + c$ and $sw(L(A)) = 3^{\lfloor \frac{m-c}{3} \rfloor}$, assuming the size of the alphabet is not bounded.*

Proof. If the size of the alphabet is not bounded, the technique from Theorem 12 can be applied. □

4.4 Intersection

In the previous chapter, we studied the potential length of the shortest word in the intersection of two languages recognized by rotating automata. In this section, we shall study this problem in the alternating automata case.

We have obtained some basic bounds on the length of the shortest word earlier in this chapter and presented a way to construct an AFA that accepts the intersection of two languages in Lemma 7. In combination, these results can be used to obtain a simple upper bound on the length of the shortest word in the intersection of two languages.

Theorem 18. *Let A_1 and A_2 be alternating automata with n and m states, respectively. If their intersection is non-empty, then $sw(L(A_1) \cap L(A_2)) \leqslant 2^{m+n+1} - 1$.*

Proof. It is possible to construct an AFA with $m+n+1$ states accepting the language $L(A_1) \cap L(A_2)$. This automaton can be subsequently converted into an NFA with 2^{m+n+1} states. □

However, our approach was quite straightforward and several questions remain open. For example, we could still attempt to find pairs of languages such that the shortest words in their intersections are close to this bound. The following theorem holds even if we consider binary alphabets only.

Theorem 19. *For every pair of integers $n \geqslant 5$ and $m \geqslant 5$, there exists an AFA A_1 with $n+1$ states and an AFA A_2 with $m+1$ states such that $sw(L(A_1) \cap L(A_2)) = g(m)(g(n)-1)$.*

Proof. We can construct an AFA A_1 with $n+1$ states such that $L(A_1) = \{w \in \{a,b\}^* \mid |w|_a \equiv g(n)-1 \pmod{g(n)}\}$ as in Theorem 11. Similarly, we can construct an AFA A_2 accepting the language $(a(b^{g(m)})^*b^{g(m)-1})^+$ with $m+1$ states, as shown in the figure for $m = 8$.

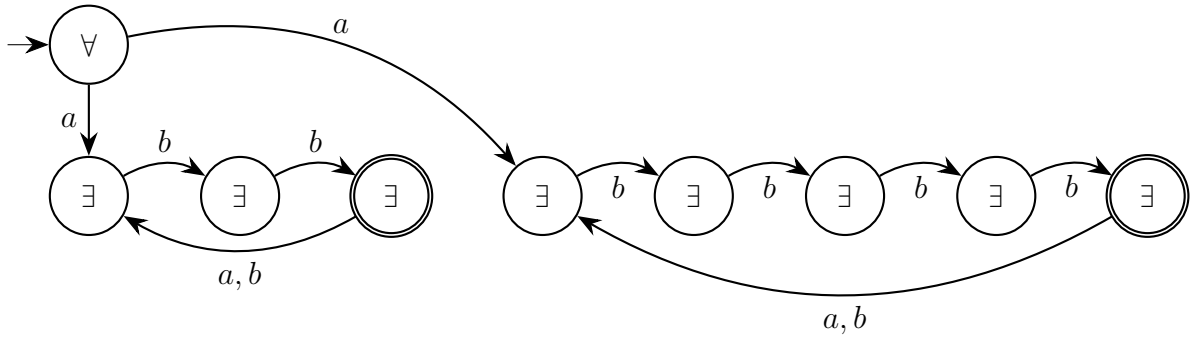

Figure 4.2: The AFA accepting the language $L = (a(b^{15})^*b^{14})^+$

□

This result is similar to the one achieved for rotating automata. If we consider the results of Theorem 13 and use an alphabet of size 6, this bound can be improved.

Theorem 20. *For every pair of integers $n \geqslant 5$ and $m \geqslant 5$, there exists an AFA A_1 with $25n+112$ states and an AFA A_2 with $m+3$ states such that $sw(L(A_1) \cap L(A_2)) = g(m)((2^n-1)(n+1)+1)$.*

Proof. Let us consider the alphabets $\Sigma_1 = \{a,b,c,d,e\}$ and $\Sigma_2 = \{f\}$. The automaton A_1 can be constructed the same way as the automaton from Theorem 13 over the alphabet Σ_1, with additional transitions $\delta(q,f) = q$ for every state q. The automaton A_2 can be constructed the same way as the automaton A_2 from theorem 19, but with different alphabets. Then, the shortest word in the intersection contains $(2^n-1)(n+1)+1$ symbols from Σ_1, each of them followed by $g(m)-1$ symbols from Σ_2. □

By encoding the alphabet Σ_1 as in [8], a weaker bound can be obtained for ternary alphabets as well.

A higher bound can be achieved if we consider alphabets of linear sizes.

Theorem 21. *For every pair of integers n and m, there exists an AFA A_1 with $3n+3$ states and an AFA A_2 with $3m+5$ states such that $sw(L(A_1) \cap L(A_2)) = 3^n(1+3^m)$.*

Proof. By combining the technique from Theorem 19 with the languages from Theorem 12, an intersection with the shortest word of length $3^n(1+3^m)$ can be obtained. □

If we consider an n-state and an m-state automaton instead, the bound is approximately 1.44^{n+m}.
Naturally, the following question arises: is it possible to achieve a bound that is exponential with regard to both parameters for alphabets of constant size? It can be shown that this is possible indeed; the key observation is that the AFA obtained by complementing an NFA uses universal states exclusively.

Theorem 22. *For every pair of integers n and m and a 10-symbol alphabet Σ, there exists an AFA A_1 with $25n+112$ states and an AFA A_2 with $25m+113$ states such that $sw(L(A_1) \cap L(A_2)) = ((2^n-1)(n+1)+1)((2^m-1)(m+1)+2)$.*

Proof. There must exist two alphabets Σ_1 and Σ_2 such that $|\Sigma_1| = |\Sigma_2| = 5$ and $\Sigma = \Sigma_1 \cup \Sigma_2$. Let h be a homomorphism such that $h(c) = c$ for every $c \in \Sigma_1$ and $h(c) = \varepsilon$ for every $c \in \Sigma_2$.
It follows from Theorem 13 that there exists an AFA with $25n+112$ states accepting a language over Σ_1 such that its shortest word is of length $(2^n-1)(n+1)+1$. Let us denote this language by L_1. There must also exist a language $L_2 \subseteq \Sigma_2^*$ such that $sw(L_2) = (2^m-1)(m+1)+1$ which can be accepted by an AFA with $25m+112$ states. Then, we can construct an AFA A_1 with $25n+112$ states such that $L(A_1) = h^{-1}(L_1)$ by taking the AFA accepting the language L_1 and adding transitions $q \in \delta(q,c)$ for every state q and every symbol $c \in \Sigma_2$. The automaton A_2 shall accept the language $(\Sigma_1 L_2)^*$. Let $A_2' = (Q_\exists', Q_\forall', \Sigma_2, \delta', q_0', F')$ be the AFA accepting the language L_2. In Theorem 13, it was shown that $25m+112$ states are sufficient. Since it was obtained as the complement of some NFA, it holds that all of its states are universal and thus $Q_\exists' = \varnothing$. We can obtain the automaton A_2 by certain modifications. First, a new initial state q_0 is added. This state is universal and also accepting. Then, the new transition function δ is defined based on the function δ'. All existing transitions are preserved, but new ones are added. For every symbol $c \in \Sigma_1$ and every accepting state $q \in F'$, it holds that $q_0' \in \delta(q_0, c)$ and that $q_0' \in \delta(q, c)$. Thus, the new AFA A_2 can be formally defined as $(\varnothing, Q_\forall' \cup \{q_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, F)$ where $F = F' \cup \{q_0\}$. We need to prove that $L(A_2) = (\Sigma_1 L_2)^*$. By induction, we can show that every run of the AFA A_2 on any

word w from the language $(\Sigma_1 L_2)^*$ finishes succesfully in some accepting state. Since $q_0 \in F$, the basis $\varepsilon \in L(A_2)$ holds. Let $w = w_1 w_2$, where $w_1 \in (\Sigma_1 L_2)^*$ and $w_2 \in \Sigma_1 L_2$. Suppose that every possible run on the input w_1 ends in some accepting state q_F. The first symbol of the word w_2 must be a symbol from Σ_1. The function δ was constructed in such a way that the automaton must read the first symbol and change its state to q_0'. (While ε-transitions may be possible as well, they may only lead to other accepting states, otherwise some run on w_1 would not be accepting.) The computation therefore continues from the configuration (q_0', w_2'), where $w_2' \in L_2$. It follows from the definition of the automaton A_2' that every possible run ends in an accepting state.

We must also prove that $L(A_2) \subseteq (\Sigma_1 L_2)^*$. Let us consider a word $w \in L(A_2)$. If $w \neq \varepsilon$, its first symbol must be from Σ_1, since no other transitions from q_0 are defined. Thus, if w contains exactly one symbol from Σ_1, it must be the first symbol and the remaining input must be a word from the language L_2. Otherwise, a non-accepting run would exist due to the properties of the AFA A_2'. We have shown earlier that if $w = w_1 w_2$ and $w_1 \in (\Sigma_1 L_2)^*$, the AFA A_2 must finish each run on the word w_1 in an accepting state. For the sake of contradiction, let us assume that $w_2 \notin (\Sigma_1 L_2)^*$. Without loss of generality, we may also assume that w_1 is the longest prefix of w from the language $(\Sigma_{12})^*$. If the first symbol of the suffix w_2 is not from Σ_1, some possible run shall not be accepting - otherwise, the prefix w_1 would not be the longest one with the given property. After reading the symbol from Σ_1, the state q_0' must be reached. Consider the part of input between the read symbol from Σ_1 and either the next such symbol or the end of the input if the remaining input contains no more symbols from Σ_1. Let us denote this part of input by v. It holds that $v \in \Sigma_2^*$. However, it must also hold that $v \notin L_2$, otherwise we could extend the prefix w_1. Then, some run on v must finish in a non-accepting state. If no symbol from Σ_1 follows, this means that it is possible to read the entire input without finishing in an accepting state. Otherwise, it holds that $\delta(q, c) = \varnothing$ and the computation halts unsuccessfuly. In either case, it is impossible that $w \in L(A_2)$ and thus a contradiction has been reached. The AFA A_2 therefore accepts the language $(\Sigma_1 L_2)^*$.

In the intersection $L(A_1) \cap L(A_2)$, there must be at least $(2^n - 1)(n + 1) + 1$ symbols from Σ_1 and each of those symbols must be followed by a word from L_2 of length $(2^m - 1)(m + 1) + 1$. Thus, it holds that $sw(L(A_1) \cap L(A_2)) = ((2^n - 1)(n + 1) + 1)((2^m - 1)(m + 1) + 2)$. □

Chapter 5

Comparison of Models

In this chapter, we shall compare the values that the lsw-function from Definition 12 attains for some sequences based on models. Up to this point, we were primarily concerned with finding some lower and upper bounds on the length of the shortest accepted word. However, it is also possible to study the relationships between the (not necessarily known) optimal values using the various model conversions. Consider the following example:

Example 7. *Let us consider the index set $\mathscr{I} = \mathbb{N}$, an alphabet Σ and two sequences of finite sets of languages $\mathscr{L}_1$ and $\mathscr{L}_2$. The n-th set of $\mathscr{L}_1$ shall contain exactly the languages which can be accepted by some RNFA over the alphabet Σ with n states and the n-th set of $\mathscr{L}_2$ shall contain exactly the languages which can be accepted by some 2NFA over the alphabet Σ with n states. Due to Lemma 5, it must hold that $lsw_{\mathscr{L}_1}(n) \leqslant lsw_{\mathscr{L}_2}(2n)$.*

Note that this result does not depend on the size of the alphabet Σ - the size can be unary, bounded by some constant or proportional to the number of states.
We have shown that we can use an upper bound for 2NFAs to obtain an upper bound for RNFAs and that we can also use a lower bound for RNFAs to obtain a lower bound for 2NFAs. For this pair of models, the conversion-based bounds are not tighter than those obtained in Chapters 2 and 3. For unary alphabets, it is known that bounds based on Landau's function are achievable for both RNFAs and 2NFAs (see Theorem 3 and [6]). Given n states, the respective lower bounds are $g(n) - 1$ and $g(n-1) - 1$. Although it follows from the conversion-based bound that there exists a $2n$-state 2NFA A_1 such that $sw(L(A_1)) \geqslant g(n) - 1$, there also exists a $2n$-state 2NFA A_2 such that $sw(L(A_1)) \geqslant g(2n-1) - 1$. For alphabets of linear sizes, the approximate lower bound 1.44^n is achievable for both types of automata. Thus, the conversion-based bound does not lead to an improvement either. In the case when the size of the alphabet is bounded by a constant k, we have shown that an RNFA with kn states can achieve a lower bound which is a polynomial of degree k with regard to $g(n)$ (see Theorem 6).

The conversion-based bound implies that a 2NFA with $2kn$ states can accept the same language with the same shortest word. However, it was shown in [6] that kn-state 2DFA can accept the same language and in [14] that an exponential lower bound is achievable with fixed alphabets.

Next, we shall compare RNFAs and NFAs. The index set $\mathscr{I} = \mathbb{N}$ shall be used throughout the chapter.

Proposition 15. *Let $\mathscr{L}_{RNFA}$ and $\mathscr{L}_{NFA}$ be sequences of finite sets of languages over an alphabet Σ, with n-th set of $\mathscr{L}_{RNFA}$ containing languages accepted by n-state RNFAs and with n-th set of $\mathscr{L}_{NFA}$ containing languages accepted by n-state NFAs. Then $lsw_{\mathscr{L}_{RNFA}}(n) \leqslant lsw_{\mathscr{L}_{NFA}}(\binom{2n}{n})$.*

Since it is known that $lsw_{\mathscr{L}_{NFA}}(n) = n - 1$ regardless of the choice of Σ, this yields the upper bound for RNFAs from Theorem 1. The function $\binom{2n}{n} - 1$ also grows faster than the function 1.44^n.

Similarly, 2NFAs and NFAs can be compared.

Proposition 16. *Let $\mathscr{L}_{2NFA}$ and $\mathscr{L}_{NFA}$ be sequences of finite sets of languages over an alphabet Σ, with n-th set of $\mathscr{L}_{2NFA}$ containing languages accepted by n-state 2NFAs and with n-th set of $\mathscr{L}_{NFA}$ containing languages accepted by n-state NFAs. Then $lsw_{\mathscr{L}_{2NFA}}(n) \leqslant lsw_{\mathscr{L}_{NFA}}(\binom{n}{\lfloor \frac{n}{2} \rfloor})$.*

The obtained lower bound for NFAs is weaker than the known one for any choice of alphabet Σ, while the upper bound for 2NFAs is the one from [6].

The AFAs can be compared to NFAs as well.

Proposition 17. *Let $\mathscr{L}_{AFA}$ and $\mathscr{L}_{NFA}$ be sequences of finite sets of languages over an alphabet Σ, with n-th set of $\mathscr{L}_{AFA}$ containing languages accepted by n-state AFAs and with n-th set of $\mathscr{L}_{NFA}$ containing languages accepted by n-state NFAs. Then $lsw_{\mathscr{L}_{AFA}}(n) \leqslant lsw_{\mathscr{L}_{NFA}}(2^n)$.*

Similarly to previous cases, the upper bound is a known one (see Theorem 14).

It is also possible to compare non-equivalent models, such as NFAs and context-free grammars. Although context-free grammars are strictly more powerful than finite automata, it is possible to construct a grammar equivalent with a given NFA. This way, we can obtain a lower bound on the length of the longest shortest word, although not necessarily a tight one. We shall make use of the following lemma.

Lemma 8. *For every n-state NFA A over a unary alphabet, there exists a context-free grammar in Chomsky normal form (CNF) with $O(n^{\frac{2}{3}})$ nonterminals such that $L(G) = L(A) \backslash \{\varepsilon\}$.*

Proof. The proof can be found in [7]. □

The definition of CNF in [7] does not allow grammars to produce the empty word. We may also consider the definition where the initial nonterminal σ does not appear on the right-hand side of any rule, but the rule $\sigma \to \varepsilon$ may be included. Then, we can construct a grammar G in CNF with $(On^{\frac{2}{3}})$ nonterminals such that $L(G) = L(A)$

Proposition 18. *Let $\mathscr{L}_{NFA}$ and $\mathscr{L}_{CFG}$ be sequences of finite sets of languages over an alphabet $\Sigma = \{a\}$, with n-th set of $\mathscr{L}_{NFA}$ containing languages accepted by n-state NFAs and with n-th set of $\mathscr{L}_{CFG}$ containing languages described by context-free grammars in CNF with n nonterminals. Then there exists an integer N and a constant c such that for every $n \geqslant N$, $lsw_{\mathscr{L}_{NFA}}(n) \leqslant lsw_{\mathscr{L}_{CFG}}(\lceil cn^{\frac{2}{3}} \rceil)$.*

Thus, for every sufficiently large n, there exists a grammar G in CNF with $\lceil cn^{\frac{2}{3}} \rceil$ nonterminals such that $L(G) = \{a^{n-1}\}$ and therefore $sw(L(G)) = n - 1$.

Conclusion

In this thesis, we studied how long may the shortest word in a regular language be with regard to the model's descriptional complexity. Other researchers have studied this problem in the cases of nondeterministic finite automata and two-way automata. Our models of choice were rotating nondeterministic finite automata and alternating finite automata.

First, we presented the necessary definitions and other preliminaries. Then, we summarized the known results and defined the function lsw. The said function generalizes the problem to all sequences of finite sets of languages, while still allowing the original approach based on automata and their number of states. The results of our research were presented in the latter chapters.

For rotating automata, we achieved lower bounds similar to those known for two-way automata. If the chosen alphabet is unary, the shortest word may be of length $g(n)-1$, where g denotes Landau's function known from number theory and n denotes the automaton's number of states. If an alphabet's size grows proportionally with the number of states, the length of the shortest word can be approximately 1.44^n. For alphabets of size k, we achieved a lower bound approximately equal to $(g(n))^k$ by modifying a technique used for two-way automata in [6]. We have also extended these lower bounds to the intersection of two languages. We were only able to obtain a simple upper bound by converting the RNFA into an NFA.

For alternating automata, we largely studied the same problems. We were able to obtain an exponential lower bound even for alphabets with limited size by constructing an AFA which accepts the complement of a specific regular expression described in [8]. However, the lower bounds for unary and unbounded alphabets were quite similar to those obtained for two-way and rotating automata despite the seeming differences between these models. We were also able to find the explanation for this: these models are able to process the input multiple times, either sequentially or parallelly. This makes it possible to accept the languages which can be expressed as the intersection of several simpler languages with relatively low descriptional complexity.

In the last chapter, we compared several models using the lsw function. In the last chapter, we obtain some lower bounds using transformations between models. The resulting bounds were generally weaker than those obtained by studying the individual

models. This suggests that using the models' specific properties is necessary in order to reach optimal results.

Bibliography

[1] Levent Alpoge, Thomas Ang, Luke Schaeffer, and Jeffrey Shallit. Decidability and shortest strings in formal languages. In *Descriptional Complexity of Formal Systems*, 2011.

[2] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43:185–190, 1992.

[3] Luc Boasson, Bruno Courcelle, and Maurice Nivat. The rational index: A complexity measure for languages. *SIAM Journal on Computing*, 10:284–296, 1981.

[4] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[5] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.

[6] Egor Dobronravov, Nikita Dobronravov, and Alexander Okhotin. On the length of shortest strings accepted by two-way finite automata. In *Developments in Language Theory*, 2019.

[7] Michael Domaratzki, Giovanni Pighizzini, and Jeffrey Shallit. Simulating finite automata with context-free grammars. *Information Processing Letters*, 84:339–344, 2002.

[8] Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Ming-Wei Wang. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics*, 10:407–437, 2005.

[9] A. Fellah, H. Jürgensen, and S. Yu. Constructions for alternating finite automata. *International Journal of Computer Mathematics*, 35(1-4):117–132, 1990.

[10] Viliam Geffert and Alexander Okhotin. One-way simulation of two-way finite automata over small alphabets. In *Non-Classical Models of Automata and Applications*, 2013.

[11] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation (2nd ed.)*. Addison-Wesley, 2000.

[12] Christos Kapoutsis. Removing bidirectionality from nondeterministic finite automata. In *Mathematical Foundations of Computer Science*, 2005.

[13] Christos Kapoutsis, Richard Královič, and Tobias Mömke. Size complexity of rotating and sweeping automata. *Journal of Computer and System Sciences*, 78:537–558, 2012.

[14] Stanislav Krymski and Alexander Okhotin. Longer shortest strings in two-way finite automata. In *Descriptional Complexity of Formal Systems*, 2020.

[15] Michal Kunc and Alexander Okhotin. Reversibility of computations in graph-walking automata. In *Mathematical Foundations of Computer Science*, 2013.

[16] Ernst Leiss. The complexity of restricted regular expressions and the synthesis problem for finite automata. *Journal of Computer and System Sciences*, 23:348–354, 1980.

[17] Ernst Leiss. Constructing a finite automaton for a given regular expression. *SIGACT News*, 12:81–87, 1980.

[18] Ernst Leiss. Succint representation of regular languages by boolean automata. *Theoretical Computer Science*, 13:323–330, 1981.

[19] Jean-Louis Nicolas. Ordre maximal d'un élément d'un groupe de permutations. *Comptes rendus de l'Académie des Sciences*, 270:1473–1476, 1970.

[20] Giovanni Pighizzini. Two-way finite automata: Old and recent results. *Fundamenta Informaticae*, 126:225–246, 2013.

[21] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *Tenth Annual ACM Symposium on Theory of Computing*, 1978.