Commenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Finite Automata with Input Transformations

Bachelor's Thesis

2019

Júlia Froncová

Commenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Finite Automata with Input Transformations

Bachelor's Thesis

Study programme	Computer Science (Single degree study, bachelor I. deg. full time form)
Study field:	Computer Science, Informatics
Department:	Department of Computer Science
Supervisor:	RNDr. Peter Kostolányi, PhD.

Bratislava, 2019
Júlia Froncová

94555116

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Júlia Froncová
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Finite Automata with Input Transformations
Konečné automaty s transformáciami vstupu

Anotácia: Práca sa bude zaoberať triedou automatov, ktoré sú rozšírením klasických nedeterministických konečných automatov o možnosť vykonávať určité transformácie vstupu. Takéto automaty sa už v literatúre skúmali pre niekoľko konkrétnych transformácií; tieto špeciálne prípady budú v práci zasadené do spoločného matematického rámca a na tejto všeobecnejšej úrovni bude učinených zopár základných pozorovaní o popisnej sile rozšírených konečných automatov.

Vedúci:	RNDr. Peter Kostolányi, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.
Dátum zadania:	17.10.2018

Dátum schválenia: 24.10.2018

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

....
študent

....
vedúci práce

94555116

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Júlia Froncová
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Finite Automata with Input Transformations

Annotation: The thesis will focus on a class of automata extending classical nondeterministic finite automata by a possibility of performing certain input transformations. Such automata have already been studied in literature for several particular transformations; these concrete instances will be placed into a common mathematical framework and some basic observations regarding descriptive power of extended finite automata will be made at this more general level.

Supervisor:	RNDr. Peter Kostolányi, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	17.10.2018
Approved:	24.10.2018

doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

....
Student

....
Supervisor

Acknowledgement: I would like to thank my supervisor RNDr. Peter Kostolányi, PhD. for his great help and advice.

Abstrakt

V práci sa budeme zaoberať konečnými automatmi, ktoré majú pridanú možnosť nejakým spôsobom transformovať svoj vstup. Bordihn, Hozler a Kutrib skúmali automaty s niekoľkými konkrétnymi transformáciami, ako reverz a otáčanie. Odrážajúc sa od tohto predošlého výskumu, vytvoríme všeobecný matematický rámec pre skúmanie automatov s transformáciami vstupu a zameriame sa najmä na výpočtovú silu automatov s fixným konštantným ohraničením počtu aplikácií transformácie na vstup. Ukážeme zopár príkladov transformácií, ktoré umožňujú konečným automatom akceptovať jazyky spadajúce do rôznych levelov Chomského hierarchie (resp. túto hierarchiu prevyšujúce). Navyše dokážeme nutnú a postačujúcu podmienku, ktorej splnením transformácia vstupu zachováva regularitu pri konštantnom počte jej použití konečným automatom.

Kľúčové slová: konečný automat, transformácia vstupu, výpočtová sila

Abstract

Finite automata with an additional ability to perform some kind of input transformations shall be considered in the thesis. Such automata have already been studied by Bordihn, Holzer, and Kutrib for several particular transformations, such as input reversal or revolving. Building upon this line of research, we shall provide a general mathematical basis for the study of automata with input transformations and mainly focus on computational power of automata that apply their input transformations at most some fixed constant number of times. We shall describe several examples of input transformations, which enable finite automata to accept languages from different levels of the Chomsky hierarchy (and beyond). Moreover, we shall prove a necessary and sufficient condition, under which an input transformation preserves regularity when applied at most a constant number of times by a finite automaton.

Keywords: finite automaton, input transformation, computational power

Contents

Introduction

Henning Bordihn, Markus Holzer, and Martin Kutrib have introduced automata with an additional possibility to perform some kind of transformation on the input for the first time in [1], where they have examined pushdown automata which can reverse their input during the computation. The question they have raised was: how does the computational power of pushdown automata change if they may reverse the remaining input k times for some natural constant k, or an arbitrary number of times. On the side, they have posed the same question about finite automata.

The research regarding the change in computational power has been later extended to finite automata with other types of input transformations such as revolving [2], or transformations inspired by biology such as hairpin transformation [4]. These automata have been collectively termed *extended finite automata*. Next in [3] hybrid extended finite automata have been introduced. These models use a set of transformations, which they may apply on the input during the computation. The research on these automata has been focused on a question, whether adding a specific input transformation to this set changes the computational power of the automaton. Finally, in [6], a brief summary of the results known about extended finite automata has been presented.

For the purpose of unified presentation of their results, Bordihn, Holzer, and Kutrib [6] have presented a semi-formal definition of a general extended automaton, which they have used as a unifying framework for examining the automata with specific input transformations. Since this definition lacks sufficient mathematical formality, it is not suitable for a research focused on answering questions about automata with *general* input transformations.

The main objective of this thesis is to initiate a study of automata with input transformations at such general level. As a result, we shall provide a new common mathematical basis for a consistent study of extended automata. To do so we shall define *input permutations* as those transformations which preserve the number of occurrences of each symbol in the input. As a specific family we shall set apart oblivious input permutations, which can be written as a permutation of positions of symbols in a word. We shall then define automata with a possibility to use these input permutations.

We shall demonstrate this denition on the transformations from the past research and add some examples of our own input permutations. We shall focus more on the family of oblivious input permutations and we shall show how some of them change the computational power of finite automata, when used a constant number of times.

Our aim in this thesis is to show what can be said about the computational power of the automata with input transformations, but not only for some specific transformations, but in general. As the main result of our research we shall identify a necessary and sufficient condition under which an input transformation preserves regularity when applied at most a constant number of times by a finite automaton.

Chapter 1

Extended Finite Automata

Automata with an additional possibility of performing some kind of transformation on the remaining input were considered for the first time by Henning Bordihn, Markus Holzer, and Martin Kutrib [1], who introduced pushdown automata with the possibility of reversing their remaining input. The question raised in [1] was whether the computational power of these pushdown automata was increased or remained the same if the automata had the possibility of reversing their input either k times for some fixed natural k, or an arbitrary number of times.

Inspired by the developments on pushdown automata extended in this way, Bordihn, Holzer, and Kutrib [1] have also considered finite automata with input reversal. The main question was the same as for pushdown automata: (how) does the computational power of finite automata change when they are allowed to reverse their input.

This question was later extended to finite automata with other types of input transformations, such as shifting to the left and right [2], or transformations inspired by biology such as hairpin transformation [4]. Such automata have been collectively termed *extended finite automata* by Bordihn, Holzer, and Kutrib [3]. In this chapter we shall give a summary of the results on these extended automata obtained in [1, 2, 3, 6, 4].

In the rest of the thesis, we shall incorporate the particular models surveyed in this chapter into a more general framework: finite automata with suitably defined input permutations. This shall provide a common mathematical basis for the consistent study of extended finite automata. We shall also take a look at the computational power of these input permutation automata.

This chapter is organized as follows: firstly, we shall present some definitions of the concepts used in [1, 2, 3, 6, 4]. Then we shall summarize the known results on the computational power of extended finite automata and consolidate them in the Table 1.1 at the end of the chapter. The most important results of this kind will be summarized in Chapter 3.

1.1 Notation

We shall use the following notation: an empty word shall be denoted by ε. Families of regular, context-free, linear context-free, and extended context-sensitive languages shall be denoted by $\mathscr{R}$, $\mathscr{L}_{CF}$, $\mathscr{L}_{LIN}$, $\mathscr{L}_{CS}$, respectively. The reversal of a word w shall be denoted by w^R and $L^R = \{w^R \mid w \in L\}$. For the length of w we shall use the notation $|w|$ and for the number of occurrences of a symbol a in w we shall use the notation $|w|_a$. We shall write $L_1 \setminus L_2$ for the difference between the languages L_1 and L_2. We shall denote by $(s)_2$, for each s in $\{0,1\}^*$, the natural number with binary representation s. The formal definitions of basic concepts of formal language theory can be found, e.g., in [7, 5, 8].

1.2 Extended Finite Automata

We shall now review the definitions of particular families of extended finite automata introduced in [1, 2, 3, 6, 4]. We shall confine ourselves to semi formal descriptions, as we shall only use them to present the results from the past research. We shall later (in Chapter 2) present our formal definition of input permutation automata, which we shall use while presenting our original results. Most of the models described in this section can be viewed as specializations of input permutation automata – the formal definition of input permutation automata will thus serve as formal definition of the automata described in this section as well.

All types of (nondeterministic) extended finite automata studied in [1, 2, 3, 6, 4] can be defined to be a 6-tuple $A = (Q, \Sigma, \delta, \Delta, q_0, F)$, where Q is a non-empty finite set of states, Σ is the input alphabet, δ is the ordinary transition function, a mapping from $Q \times (\Sigma \cup \{\varepsilon\})$ to 2^Q, Δ is an input-transforming transition function, a mapping from $Q \times \Sigma$ to 2^Q, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states.

Configurations of an extended finite automaton $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ are pairs (q, w), where $q \in Q$ is the current state, and $w \in \Sigma^*$ is the remaining part of the input. A step from one configuration to a successor configuration can be induced either by δ as an ordinary step of a finite automaton, or by Δ, which means that a transformation will be applied to the remaining input.

In particular, for the input transformations examined in [1, 2, 3, 6], the computational step can be defined as follows. Let us consider $a, b \in \Sigma$, $w \in \Sigma^*$, and p in $\Delta(q, a)$:

(a) An input-reversal step (later *ir*): $(q, a) \vdash_A (p, a)$ and $(q, aw) \vdash_A (p, w^R a)$.

(b) A left-revolving step (later *lr*): $(q, a) \vdash_A (p, a)$ and $(q, awb) \vdash_A (p, baw)$.

(c) A right-revolving step (later rr): $(q, a) \vdash_A (p, a)$ and $(q, aw) \vdash_A (p, wa)$.

(d) A circular-interchanging step (later ci): $(q, a) \vdash_A (p, a)$ and $(q, awb) \vdash_A (p, bwa)$.

(e) A circular-shift transition (later cs): $(q, a) \vdash_A (p, a)$ and $(q, aw) \vdash_A (p, vau)$, for all u and v with $w = uv$.

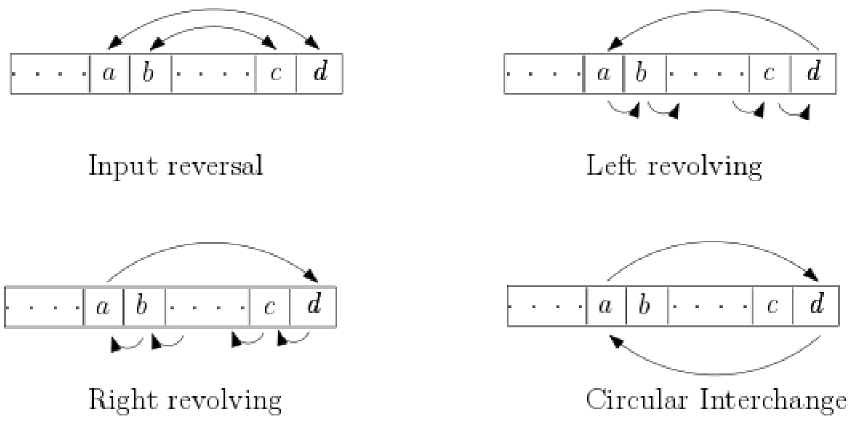


Figure 1.1: Input transformations: input-reversal, left-revolving, right-revolving, and circular-interchange

In all cases described above, the language accepted by an extended finite automaton A is defined as $L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash_A^* (q, \varepsilon), q \in F\}$.

Before we proceed to the results, there are some more useful terms and notations used in the articles [1, 2, 3, 6]:

The language accepted by a finite extended automaton A with at most k non-ordinary steps, for some $k \in \mathbb{N}$, is denoted by $L_k(A)$. The family of languages $\mathscr{L}$(o-NFA), is a family of languages accepted by nondeterministic extended finite automata with a possibility to use the transformation o. For example $\mathscr{L}$(lr-NFA) is a family of languages accepted by left-revolving finite automata.

A hybrid extended finite automaton is an interesting concept studied specifically in the article [3]. This automaton can also transform the yet unread input, but this time it may choose from a set of input transformations. The family of languages accepted by hybrid extended automata with a set of input transformations O is denoted by $\mathscr{L}$(O-NFA).

1.3 Automata with a Single Input Transformation

In 2004, Bordihn, Holzer, and Kutrib examined automata with a possibility of reversing their input [1]. Although the main concern of the article were pushdown automata, they slightly touched on finite automata and came to these conclusions:

Theorem 1. *Let k be some natural number. A language L is accepted by some input-reversal finite automaton $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ with exactly k (at most k) input-reversals, if and only if L is regular.*

Theorem 2. *A language L is accepted by some input-reversal finite automaton (with an unbounded number of input-reversals) if and only if L is a linear context-free language.*

Later in 2005, extended finite automata with shifting transformations, namely left-revolving, right-revolving, and circular interchange, were examined [2]. The outcome of the examination was that the latter transformation does not increase the computational power of finite automata, even if the transformation is used an unbounded number of times. This is true about the former two transformations as well, but only if they are used a constant number of times during the computation. Families of languages accepted by finite automata allowing an unbounded number of these input transformations are strict subfamilies of the family of context-sensitive languages and superfamilies of the regular languages. These families are also incomparable with the family of context-free languages.

Moreover, it is shown in [2] that if we take a reversal of a language accepted by a right revolving finite automaton, we can simulate its computation by left revolving. Finally, it was observed that the family of languages accepted by left-revolving finite automata and the family of languages accepted by right-revolving finite automata are incomparable.

The above described results are summarized in the following theorems.

Theorem 3. *A language L is accepted by some circular-interchanging finite automaton (with an unbounded number of input-reversals) if and only if L is regular.*

Theorem 4. *Let k be a non-negative integer. A language L is accepted by a revolving finite automaton A with at most k revolving steps, i.e., $L_k(A) = L$, if and only if L is regular.*

Theorem 5. *The family $\mathscr{L}(lr\text{-}NFA)$ is incomparable with $\mathscr{L}(rr\text{-}NFA)$.*

Theorem 6. *Every language accepted by a revolving finite automaton is context sensitive.*

Theorem 7. *The family of linear context free languages is a strict subfamily of languages accepted by left-revolving finite automata.*

Theorem 8. *Let L be accepted by a right-revolving finite automaton A, i.e., $L = L(A)$. Then the reversal of L can be accepted by some left-revolving finite automaton B, i.e., $L(B) = L^R$.*

Here is a summary of relations between the families of languages accepted by extended finite automata with shifting transformations:

1. $\mathscr{R} \subsetneq \mathscr{L}(rr\text{-}NFA) \subsetneq \mathscr{L}(\{lr, rr\}\text{-}NFA) \subsetneq \mathscr{L}_{CS}$.
2. $\mathscr{L}_{LIN} \subsetneq \mathscr{L}(lr\text{-}NFA) \subsetneq \mathscr{L}(\{lr, rr\}\text{-}NFA) \subsetneq \mathscr{L}_{CS}$.
3. The families $\mathscr{L}(lr\text{-}NFA)$ and $\mathscr{L}(rr\text{-}NFA)$ are incomparable.
4. Each of the families $\mathscr{L}(lr\text{-}NFA)$, $\mathscr{L}(rr\text{-}NFA)$, and $\mathscr{L}(\{lr, rr\}\text{-}NFA)$ are incomparable with $\mathscr{L}_{CF}$. Furthermore, $\mathscr{L}(rr\text{-}NFA)$ is incomparable with $\mathscr{L}_{LIN}$.

$$\begin{tikzcd}
 & \mathscr{L}_{CS} & \\
\mathscr{L}_{CF} \arrow[ur] & \mathscr{L}(\{rr, lr\} - NFA) \arrow[u] & \\
 & & \mathscr{L}(rr - NFA) \arrow[uul] \arrow[ul] \\
\mathscr{L}_{LIN} = \mathscr{L}(ir - NFA) \arrow[uu] \arrow[r] & \mathscr{L}(lr - NFA) \arrow[uu] & \\
 & \mathscr{L}^R(rr - NFA) \arrow[u] & \\
 & \mathscr{R} = \mathscr{L}(ci - NFA) \arrow[u] \arrow[uul] \arrow[uuur] &
\end{tikzcd}$$

Figure 1.2: Inclusion structure of families of languages accepted by finite automata with shifting transformations. All inclusions depicted are strict, and families that are not connected with an arrow are pairwise incomparable.

To illustrate the theorems above, we shall present some examples of languages accepted by extended automata with a possibility to use the above mentioned input transformations an arbitrary number of times [2].

Example 1. *Let $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ be a left(right)-revolving finite automaton, where $Q = \{q_0, q_a, q_b\}$, $F = \{q_0\}$, $\Sigma = \{a, b\}$, and*

$$\begin{aligned}
\delta(q_0, c) &= \{q_c\} \ (\forall c \in \Sigma), \\
\delta(q_c, d) &= \{q_0\} \ (\forall c, d \in \Sigma \textit{ such that } c \neq d), \\
\Delta(q_c, c) &= \{q_c\} \ (\forall c \in \Sigma).
\end{aligned}$$

This automaton starts to read the input in q_0. It reads the first symbol and remembers it. If the next symbol is the same as the remembered, the automaton revolves *the input to left (right) and it keeps revolving it until it finds the other symbol. After reading the other symbol, it starts form q_0 anew. It is easy to see that the automaton accepts the language $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$. This example shows that there exists a non-regular language which can be accepted by a revolving automaton.*

Example 2. *Let $A = (Q, \Sigma, \delta, \Delta, q_0, F)$ be a left-revolving finite automaton, where $Q = \{q_0, q_a, q_b, q'_a, q'_b\}$, $F = \{q_0\}$, $\Sigma = \{a, b\}$, and*

$$\begin{aligned}\delta(q_0, c) &= \{q_c\} \ (\forall c \in \Sigma),\\ \Delta(q_c, d) &= \{q'_c\} \ (\forall c, d \in \Sigma),\\ \delta(q'_c, c) &= \{q_0\} \ (\forall c \in \Sigma).\end{aligned}$$

This automaton starts to read the input in q_0. It reads the first symbol and remembers it. Next it revolves the input to left still remembering the read symbol. If the following symbol is the same as the remembered, the automaton starts again from q_0. It is easy to see that the automaton accepts the language $\{ww^R \mid w \in \{a,b\}^*\}$. This example shows that there exists a language which can be accepted by a left-revolving automaton, but (as can be shown [2]) cannot be accepted by any right-revolving automaton.

1.4 Hybrid Finite Automata

The research of Bordihn, Holzer, and Kutrib [3] on hybrid extended finite automata dealt with the set of transformations $O = \{ir, lr, rr, ci, cs\}$. Their results can be summarized by the following theorems.

Theorem 9. *Let $X \subseteq \{ir, lr, rr, ci, cs\}$ and k be a non-negative integer. A language L is accepted by a X-NFA A with at most k non-ordinary steps, if and only if L is regular.*

Theorem 10. *Let $X \subseteq \{ir, lr, rr, ci, cs\}$. Then $\mathscr{L}(X\text{-}NFA) \subsetneq \mathscr{L}_{CS}$*

Theorem 11. *For any $X \subseteq \{ir, lr, rr, ci, cs\}$ with at least one of the transformations $\{lr, rr, cs\}$, the family $\mathscr{L}(X\text{-}NFA)$ is incomparable with $\mathscr{L}_{CF}$.*

Lemma 1. *(a) $\mathscr{L}(ir\text{-}NFA) = \mathscr{L}(\{ir, ci\}\text{-}NFA)$.*

(b) $\mathscr{L}(rr\text{-}NFA) \subsetneq \mathscr{L}(\{rr, ci\}\text{-}NFA)$.

(c) $\mathscr{L}(cs\text{-}NFA) \subsetneq \mathscr{L}(\{cs, ci\}\text{-}NFA)$.

Theorem 12. *For $x \in \{lr, rr\}$, we have $\mathscr{R} \subsetneq \mathscr{L}(cs\text{-}NFA) \subsetneq \mathscr{L}(x\text{-}NFA) = \mathscr{L}(\{x, cs\}\text{-}NFA)$.*

Theorem 13. *For any $X \subseteq \{ir, lr, rr, cs\}$ with $|X \cap \{ir, lr, rr\}| > 1$, we have $\mathscr{L}(X$-$NFA) = \mathscr{L}(\{lr, rr\}$-$NFA)$.*

To sum up, we see that any combination of transformations used during the computation of extended finite automata results in a family of languages that falls into the family of context-sensitive languages. Any combination containing at least one of the shifting transformations results in a family of languages which is incomparable with the family of context-free languages. Circular interchange increases the power of extended finite automata originally containing either right-revolving or circular-shift but does not increase the power of the extended automata originally containing input-reversal. The family of languages accepted by automata with the possibility to use both right-revolving and left-revolving is equal to the family of languages accepted by automata containing a combination of shifting transformations and an input-reversal transformation if the combination contains more than one of these three transformations: left-revolving, right-revolving, and input-reversal.

1.5 Hairpin Finite Automata

In 2007, Bordihn, Holzer, and Kutrib [4] initiated a study of a new kind of extended automata with a possibility to use different types of hairpin input transformations. The hairpin input transformation was inspired by the way how the DNA can fold. It works as follows: it sets one pointer on the first symbol of a given word. Then it finds another occurrence of this symbol in the word and sets on it a second pointer. Then it reverses the word in between the two pointers.

There are three different types of the hairpin transformation: a left-most, a right-most, and a general hairpin input transformation.

Let us consider an input-transforming function Δ, $a \in \Sigma$, $v, w \in \Sigma^*$, $u \in (\Sigma \setminus \{a\})^*$ and p in $\Delta(q, a)$. We then have:

1. A left-most hairpin step (later lh): $(q, auaw) \vdash_A (p, au^R aw)$.

2. A general hairpin step (later h): $(q, avaw) \vdash_A (p, av^R aw)$.

3. A right-most hairpin step (later rh): $(q, awau) \vdash_A (p, aw^R au)$.

It was shown that these transformations do not increase the computational power of the finite automata, when they are used only a constant number of times during the computation. If an unbounded number of these transformations is used, these finite automata accept languages that are properly contained in the family of context-sensitive languages and are superfamilies of the family of regular languages. The following theorems summarize the results from [4].

Theorem 14. *Let k be a non-negative integer. A language L is accepted by any type of hairpin automaton A with at most k hairpin steps, if and only if L is regular.*

Theorem 15. *There is a language accepted by a general hairpin automaton, which is accepted neither by a left-most, nor by a right-most hairpin automaton.*

Theorem 16. *There is a language accepted by a right-most hairpin automaton, which is accepted neither by a left-most, nor by a general hairpin automaton language.*

Theorem 17. *There is a non-context-free right-most hairpin automaton language.*

1.6 Summary of Results

To sum up the results, we can see that if we have extended automata using any of the examined input transformations at most k times, for some natural k, the family of languages accepted by these automata is equal to the family of regular languages. If we can use the transformation an unbounded number of times, the computational power of such automata increases to the ability of accepting languages from a strict subfamily of the context sensitive languages. In the case of input-reversal the automata can accept languages belonging to the linear context-free language family and in the case of circular-interchange, the computational power of the automata with the possibility to use this transformation even an unbounded number of times does not increase beyond accepting regular languages. For better picture see the table below.

	At most k transformations	Unbounded number of transformations
ci	$\mathscr{R}$	$\mathscr{R}$
lr	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$
rr	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$
ir	$\mathscr{R}$	$\mathscr{L}_{LIN}$
cs	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$
lh	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$
rh	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$
h	$\mathscr{R}$	$\subsetneq \mathscr{L}_{CS}$

Table 1.1: Computational power of particular families of extended finite automata.

Chapter 2

Finite Input Permutation Automata

We shall now introduce our general framework of finite automata extended with input transformations, which we shall call *finite input permutation automata*. This will provide common mathematical framework for models described in the previous chapter.

This chapter is divided into two sections. In the first one, we shall define *input permutations*, which shall generalize most of the particular input transformations from the previous chapter. By an input permutation we shall understand a mapping on words which satisfies a condition that when it is applied on a word over a set of symbols, the number of occurrences of each symbol in the permuted word does not differ from the number in the original word. As a specialization of this definition, we shall also introduce a family of *oblivious input permutations*. Such permutations do not depend on the specific symbols but only on their positions in the word. We shall be later working with the second definition, because it is simpler and so more can be said about it.

In the second section, we shall define input permutation automata. These are similar to the particular models described in the previous chapter, only this time we shall use our general input permutations instead of some particular input transformation.

We shall define two variants of input permutation automata: a general one and a blind one. Each of them has a possibility to apply a specific input permutation. The difference between them is that the latter one can only nondeterministically decide to apply the input permutation on the remaining input. The former one can also peek at the first symbol of the remaining input and decide according to that if it applies the input permutation or continues in the computation without applying it.

2.1 Input Permutations

Let us fix an infinite universe of symbols Ω for the rest of the thesis. From now on, we shall always suppose that $\Sigma \subseteq \Omega$ holds for each alphabet Σ.

Definition 1. *An input permutation is a sequence $\Phi = (\Phi_n)_{n\in\mathbb{N}}$, where $\Phi_n : \Omega^n \longrightarrow \Omega^n$ is a mapping between words of length n for each n in $\mathbb{N}$, such that $|w|_c = |\Phi(w)|_c$ is satisfied for all w in Ω^n and each c in Ω. For w in Ω^* and in particular for $w \in \Sigma^*$ for each $\Sigma \subseteq \Omega$, we shall write $\Phi(w)$ to denote $\Phi_{|w|}(w)$. For each language $L \subseteq \Omega^*$, $\Phi(L) = \{\Phi(w) \mid w \in L\}$.*

Definition 2. *An oblivious input permutation is a sequence $\varphi = (\varphi_n)_{n\in\mathbb{N}}$, where $\varphi_n : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ is a bijection for each n in $\mathbb{N}$, which we call an index permutation. We shall identify φ with an input permutation $\Phi = (\Phi_n)_{n\in\mathbb{N}}$ defined for all n in $\mathbb{N}$ and all $a_0, a_1, ..., a_{n-1}$ in Ω by $\Phi_n(a_0...a_{n-1}) = a_{\varphi_n(0)}...a_{\varphi_n(n-1)}$.*

We shall thus write $\varphi_n(w) = \Phi_n(w)$ for each n in $\mathbb{N}$ and w in Σ^n for some alphabet Σ, as well as $\varphi(w) = \Phi(w)$ for each w in Σ^*. For each language $L \subseteq \Sigma^*$, $\varphi(L) = \{\varphi(w) \mid w \in L\}$.

We shall use the following notation: let $\Phi = (\Phi_n)_{n\in\mathbb{N}}$ be an input permutation (or in particular oblivious input permutation). Then we shall write $\Phi[n]$ for the mapping Φ_n.

Remark 1. *The permutations defined above are non-uniform in general, as the mappings may be defined differently for each word length. The oblivious permutations depend on word length only, whereas the general permutations can also depend on particular symbols in the word.*

The following examples show that the input transformations considered in [1, 2, 3, 6] can be modeled using oblivious input permutations.

Example 3. *Circular interchange is an oblivious input permutation $\varphi = (\varphi_n)_{n\in\mathbb{N}}$, such that for each $n \in \mathbb{N}$, $\varphi_n(0) = n-1$, $\varphi_n(n-1) = 0$, and $\varphi_n(i) = i$ for $i = 1, ..., n-2$.*

Example 4. *Right revolving is an oblivious input permutation $\varphi = (\varphi_n)_{n\in\mathbb{N}}$, such that for each $n \in \mathbb{N}$, $\varphi_n(i) = (i+1) \pmod{n}$ for $i = 0, ..., n-1$.*

Example 5. *Left revolving is an oblivious input permutation $\varphi = (\varphi_n)_{n\in\mathbb{N}}$, such that for each $n \in \mathbb{N}$, $\varphi_n(i) = (i-1) \pmod{n}$ for $i = 0, ..., n-1$.*

Example 6. *Input reversal is an oblivious input permutation $\varphi = (\varphi_n)_{n\in\mathbb{N}}$, such that for each $n \in \mathbb{N}$, $\varphi_n(i) = n-1-i$ for $i = 0, ..., n-1$.*

It is easy to see that left-most hairpin and right-most hairpin mentioned in the article [4] cannot be modeled via oblivious input permutations. The following examples show that they can be captured by non-oblivious input permutations.

Example 7. *Left-most hairpin is an input permutation $\Phi = (\Phi_n)_{n\in\mathbb{N}}$, such that for each $w = avau$, where $a \in \Omega$, $v \in (\Omega \setminus \{a\})^*$, and $u \in \Omega^*$, $\Phi_{|w|}(w) = av^Rau$.*

Example 8. *Right-most hairpin is an input permutation $\Phi = (\Phi_n)_{n \in \mathbb{N}}$, such that for each $w = avau$, where $a \in \Omega$, $v \in \Omega^*$, and $u \in (\Omega \setminus \{a\})^*$, $\Phi_{|w|}(w) = av^R au$.*

One of the input transformations described in Chapter 1 was circular shift. This can be modeled via neither of our input permutation definitions. The oblivious one has a fixed index permutation for each $n \in \mathbb{N}$. Similarly, non-oblivious input permutation maps a specific word of a length $n \in \mathbb{N}$ to a specific word of the same length. On the other hand, circular shift can have multiple outputs for a single word. For this reason some kind of "nondeterministic" input permutations would be needed. These would certainly be possible to define, however we shall focus on "deterministic" input permutations only.

2.2 Input Permutation Automata

In this section we shall present two models of automata with a possibility to apply an input permutation on the remaining input of the automaton. One just nondeterministically decides when to apply the permutation and the other one reads the first symbol of the remaining input and decides according to that, if it applies the permutation or not.

Definition 3. *A blind input permutation automaton is a tuple $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$, where Q is a non-empty finite set of states, Σ is an input alphabet, δ is an ordinary transition function, a mapping from $Q \times (\Sigma \cup \{\varepsilon\})$ to 2^Q, Δ is an input-transforming transition function, a mapping from Q to 2^Q, φ is an input permutation, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of accepting states.*

Definition 4. *Let $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ be a blind input permutation automaton. A configuration of A is a pair $(q, w) \in Q \times \Sigma^*$.*

The q in the definition of the configuration represents the current state of the automaton A and w represents the remaining input.

Definition 5. *Let $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ be a blind input permutation automaton. A transition step of A is a binary relation $\vdash_A$ on the set of configurations of A, such that*

(i) If $p \in \delta(q, a)$ for some $p, q \in Q$, and $a \in \Sigma \cup \{\varepsilon\}$, then $(q, aw) \vdash_A (p, w)$ for all $w \in \Sigma^*$.

(ii) If $p \in \Delta(q)$ for some $p, q \in Q$, then $(q, w) \vdash_A (p, \varphi(w))$ for all $w \in \Sigma^*$.

(iii) No other pairs of configurations are in the relation $\vdash_A$.

Definition 6. *The language accepted by a blind input permutation automaton* $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ *is defined by* $L(A) = \{w \in \Sigma^* \mid \exists q_f \in F : (q_0, w) \vdash_A^* (q_f, \varepsilon)\}$.

Definition 7. *An input permutation automaton is a 7-tuple* $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$*, where* Q *is a non-empty finite set of states,* Σ *is an input alphabet,* δ *is an ordinary transition function, a mapping from* $Q \times (\Sigma \cup \{\varepsilon\})$ *to* 2^Q*,* Δ *is an input-transforming transition function, a mapping from* $Q \times (\Sigma \cup \{\varepsilon\})$ *to* 2^Q*,* φ *is an input permutation,* $q_0 \in Q$ *is an initial state, and* $F \subseteq Q$ *is a set of accepting states.*

The definition of a configuration of an input permutation automaton is the same as for the blind input permutation automaton.

Definition 8. *Let* $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ *be an input permutation automaton. A transition step of* A *is a binary relation* $\vdash_A$ *on the set of configurations of* A*, such that*

(i) If $p \in \delta(q, a)$ *for some* $p, q \in Q$*, and* $a \in \Sigma \cup \{\varepsilon\}$*, then* $(q, aw) \vdash_A (p, w)$ *for all* $w \in \Sigma^*$.

(ii) If $p \in \Delta(q, a)$ *for some* $p, q \in Q$*, and* $a \in \Sigma \cup \{\varepsilon\}$*, then* $(q, aw) \vdash_A (p, \varphi(aw))$ *for all* $w \in \Sigma^*$.

(iii) No other pairs of configurations are in the relation $\vdash_A$.

The definition of a language accepted by an input permutation automaton is analogical to the one for the blind input permutation automaton in Definition 6.

Let $k \in \mathbb{N}$. We shall denote by $L^{(k)}(A)$ a language of all $w \in \Sigma^*$, such that there exists an accepting computation of a (blind) input permutation automaton $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ on w, during which the input permutation is applied at most k times.

Definition 9. *A (blind) input permutation automaton* A *is* k*-permuting for some* $k \in \mathbb{N}$*, if it applies its input permutation at most* k*-times on each input. In other words, a* k*-permuting automaton uses the input-transforming function at most* k *times during the computation on any input.*

For any $k \in \mathbb{N}$ and an arbitrary input permutation automaton A, we may construct an automaton simulating A while maintaining a counter of applications of the input permutation. In this way we can filter out the computations during which the input permutation is applied at most k times. We shall use this construction in the proof of the following lemma.

Lemma 2. *For every blind input permutation automaton* A *and* $k \in \mathbb{N}$*, there exists a* k*-permuting blind input permutation automaton* A_k*, such that* $L(A_k) = L^{(k)}(A)$.

Proof. Let $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ be a blind input permutation automaton. Let us construct a blind input permutation automaton $A_k = (Q_k, \Sigma, \delta_k, \Delta_k, \varphi, (q_0, 0), F_k)$, where

$$\begin{aligned}
Q_k &= Q \times \{0, 1, ..., k\},\\
F_k &= F \times \{0, 1, ..., k\},\\
\delta_k((q, i), a) &= \{(p, i) \mid p \in \delta(q, a)\}\ (\forall q \in Q,\ a \in \Sigma \cup \{\varepsilon\},\ i \in \{0, 1, ..., k\}),\\
\Delta_k((q, i)) &= \{(p, i + 1) \mid p \in \Delta(q)\}\ (\forall q \in Q,\ i \in \{0, 1, ..., k - 1\}\}),\\
\Delta_k((q, k)) &= \varnothing\ (\forall q \in Q).
\end{aligned}$$

Clearly A_k is k-permuting and $L(A_k) = L^{(k)}(A)$. □

Lemma 3. *For every input permutation automaton A and $k \in \mathbb{N}$, there exists a k-permuting input permutation automaton A_k, such that $L(A_k) = L^{(k)}(A)$.*

Proof. The proof is analogical to the one of Lemma 2. The only difference is in the definition of Δ_k, where we put

$$\begin{aligned}
\Delta_k((q, i), a) &= \{(p, i + 1) \mid p \in \Delta(q, a)\}\ (\forall q \in Q,\ a \in \Sigma \cup \{\varepsilon\},\ i \in \{0, 1, ..., k - 1\}),\\
\Delta_k((q, k), a) &= \varnothing\ (\forall q \in Q,\ a \in \Sigma \cup \{\varepsilon\}).
\end{aligned}$$

□

Definition 10. *Let φ be an input permutation and k some natural number. We define $\mathscr{L}_{BL}(\varphi)$ as a family of languages accepted by blind input permutation automata with the input permutation φ, and $\mathscr{L}_{BL}(\varphi, k)$ as a family of languages accepted by blind k-permuting input permutation automata with the input permutation φ.*

Definition 11. *Let φ be an input permutation and k some natural number. We define $\mathscr{L}(\varphi)$ as a family of languages accepted by input permutation automata with the input permutation φ, and $\mathscr{L}(\varphi, k)$ as a family of languages accepted by k-permuting input permutation automata with the input permutation φ.*

Proposition 1. *Let φ be an input permutation and $i, j \in \mathbb{N}$, such that $i \leq j$. Then $\mathscr{L}_{BL}(\varphi, i) \subseteq \mathscr{L}_{BL}(\varphi, j)$ and $\mathscr{L}(\varphi, i) \subseteq \mathscr{L}(\varphi, j)$.*

Proof. Evident. □

Proposition 2. *Let φ be an input permutation and $k \in \mathbb{N}$. Then $\mathscr{L}_{BL}(\varphi) \subseteq \mathscr{L}(\varphi)$ and $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{L}(\varphi, k)$.*

Proof. We shall show that for each blind input permutation automaton A with an input permutation φ, there exists an input permutation automaton A', such that $L(A') = L(A)$.

Let $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ be a blind input permutation automaton and $A' = (Q, \Sigma, \delta, \Delta', \varphi, q_0, F)$ be an input permutation automaton that has all the components of the 7-tuple identical with the components of the tuple A with the only diffenrence of the input-transforming function Δ'. We shall put $\Delta'(q, \varepsilon)$ for all $\Delta(q)$. It is easy to see that $L(A') = L(A)$. $\square$

Chapter 3

Power of Input Permutation Automata

This chapter shall be about the computational power of the input permutation automata defined in the previous chapter. For simplicity of analysis we shall focus on oblivious permutations only. *All permutations are understood to be oblivious in what follows.*

In the first section, we shall present some examples of oblivious input permutations that allow the input permutation automata to accept languages that are not regular, not context-free, or not even recursively enumerable, even when the automata have a possibility to use the input permutation only once during the computation.

In the second section we shall deal with oblivious input permutations, such that automata which use them always accept regular languages. After a series of lemmas we shall prove a theorem characterizing oblivious permutations φ, such that $\mathscr{L}(\varphi, k) = \mathscr{R}$ and $\mathscr{L}_{BL}(\varphi, k) = \mathscr{R}$, for $k \in \mathbb{N}$.

3.1 Permutations Not Preserving Regularity

The input permutations examined in the previous research, namely left- and right-revolving, circular interchange and reversal can be described as oblivious input permutations, as we have shown in the previous chapter. Languages accepted by input permutation automata with the possibility to apply the above mentioned input permutations at most k times for some k in $\mathbb{N}$ fall into the family of regular languages. The following examples show that some oblivious input permutations might empower the automata to accept some languages that are not regular, not context-free, or not even recursively enumerable.

The first example describes an input permutation that empowers finite automata to accept languages that are context-free but not regular.

Example 9. *Let us define an oblivious input permutation $\varphi = (\varphi_n)_{i\in\mathbb{N}}$ for each $n \in \mathbb{N}$ as follows: if $n \equiv 0 \pmod 2$, for all $i \in \mathbb{Z}_n$*

$$\varphi_n(i) = \begin{cases} i & \text{if } i \equiv 0 \pmod n, \\ n-i & \text{if } i \equiv 1 \pmod n. \end{cases}$$

If $n \equiv 1 \pmod 2$ and $\lfloor n/2 \rfloor \equiv 0 \pmod 2$, for all $i \in \mathbb{Z}_n$

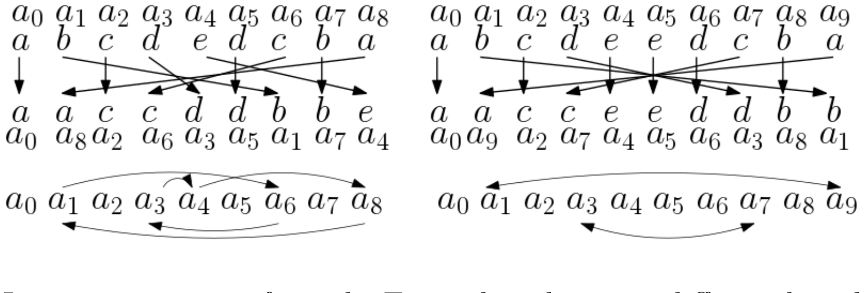
$$\varphi_n(i) = \begin{cases} \lfloor n/2 \rfloor & \text{if } i = n-1, \\ n-i & \text{if } i \equiv 1 \pmod 2 \ \wedge \ i < \lfloor n/2 \rfloor, \\ n-i-2 & \text{if } i \equiv 0 \pmod 2 \ \wedge \ i \geq \lfloor n/2 \rfloor, \\ i & \text{else.} \end{cases}$$

If $n \equiv 1 \pmod 2$ and $\lfloor n/2 \rfloor \equiv 1 \pmod 2$, for all $i \in \mathbb{Z}_n$

$$\varphi_n(i) = \begin{cases} \lfloor n/2 \rfloor & \text{if } i = n-1, \\ n-i & \text{if } i \equiv 1 \pmod 2 \ \wedge \ i \leq \lfloor n/2 \rfloor, \\ n-i-2 & \text{if } i \equiv 0 \pmod 2 \ \wedge \ i > \lfloor n/2 \rfloor, \\ i & \text{else.} \end{cases}$$

It is possible to construct a blind input permutation automaton with the input permutation φ, that would apply this input permutation in the beginning of the computation, and accept the language $\{a^n b^n \mid n \in \mathbb{N}\}$, which does not belong to the family of regular languages. The index permutation transforms an input $w = a^n b^n$, where $n \in \mathbb{N}$, to $(ab)^n$. The language of all such words is obviously regular.

It is also possible to construct an input permutation automaton with φ, which applies it on the input in the beginning of the computation that accepts the language $L = \{w \in \Sigma^* \mid w = w^R\}$, which is also not regular. This index permutation shuffles the indices in a way that the first and the last symbols of the input are next to each other, the second and the second last as well and so on. The automaton can then easily check if the permuted input is made up of a sequence of pairs of the same symbols. In case the length of the input is odd, there will be one extra symbol at the end of the permuted input, which was originally on the $\lfloor n/2 \rfloor$-th position.



a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
a b c d e d c b a
a a c c d d b b e
a_0 a_8 a_2 a_6 a_3 a_5 a_1 a_7 a_4
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9
a b c d e e d c b a
a a c c e e d d b b
a_0 a_9 a_2 a_7 a_4 a_5 a_6 a_3 a_8 a_1
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9

Figure 3.1: Input permutation from the Example 9 shown on different lengths of inputs.

Next we shall show an example of a context-sensitive language, which does not belong to the $\mathscr{L}_{CF}$ family, that can be accepted by an automaton with an input permutation.

Example 10. *Let us define an oblivious input permutation $\varphi = (\varphi_n)_{i\in\mathbb{N}}$ as follows: let $n \in \mathbb{N}$. If $n \equiv 0 \pmod 2$ and $n/2 \equiv 0 \pmod 2$, then for all $i \in \mathbb{Z}_n$*

$$\varphi_n(i) = \begin{cases} n/2 + i - 1 & \textit{if } i \equiv 1 \pmod 2 \ \wedge \ i < n/2, \\ n/2 + i + 1 & \textit{if } i \equiv 0 \pmod 2 \ \wedge \ i \geq n/2, \\ i & \textit{else.} \end{cases}$$

If $n \equiv 0 \pmod 2$ and $n/2 \equiv 1 \pmod 2$, then for all $i \in \mathbb{Z}_n$

$$\varphi_n(i) = \begin{cases} n/2 + i - 1 & \textit{if } i \equiv 1 \pmod 2, \\ i & \textit{else.} \end{cases}$$

This permutation shuffles the symbols similarly to the example $\{w \in \Sigma^* \mid w = w^R\}$, but this time it does not take the symbol from the end of the word, but from the middle. If n is an even number and if we divide an input word w in half into w_1 and w_2, then applying φ on w results in a word composed of a sequence of pairs of symbols, in which one symbol is from w_1 and one from w_2, both from the same position in their words. This can be easily checked by a finite automaton. It is easy to see that this automaton accepts the language $\{ww \mid w \in \Sigma^*\}$.

If $n \equiv 1 \pmod 2$, it does not matter how the index permutation φ_n is defined, we can for example put $\varphi_n(i) = i$ for all $i \in \mathbb{Z}_n$, because the input then can never belong to the language $L = \{ww \mid w \in \Sigma^*\}$. This can also be checked by a finite automaton.

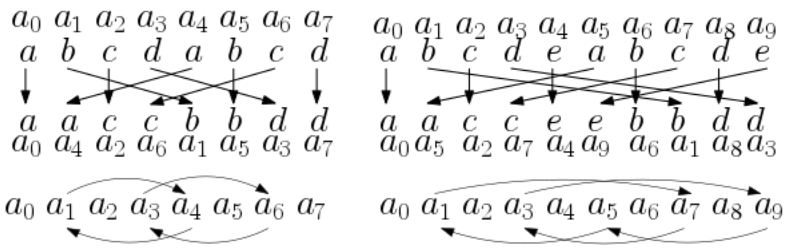

a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
a b c d a b c d
a a c c b b d d
a_0 a_4 a_2 a_6 a_1 a_5 a_3 a_7
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9
a b c d e a b c d e
a a c c e e b b d d
a_0 a_5 a_2 a_7 a_4 a_9 a_6 a_1 a_8 a_3

Figure 3.2: Input permutation from the Example 10 shown on different lengths of inputs.

We shall show that input permutation automata with an oblivious input permutation can even accept languages that do not fall into the family of recursively enumerable languages. We shall do this by reduction, where we shall show that with the help of such automaton we can construct a Turing machine accepting a language of binary codes of Turing machines which accept an empty language, which is clearly not recursively enumerable. In what follows $\langle M\rangle$ denotes any effective encoding of Turing machines into binary strings.

Example 11. *Let $L_\varnothing = \{\langle M\rangle \mid M \text{ is a Turing machine } \wedge\ L(M) = \varnothing\}$. We can map these binary codes of Turing machines from the language $L_\varnothing$ into a set of natural numbers S, where each number would represent one code of a Turing machine from $L_\varnothing$. Let convert : $\{0,1\}^* \longrightarrow \mathbb{N}$ be a mapping from binary strings to natural numbers, such that for $s \in \{0,1\}^*$: $convert(s) = (1s)_2$. In other words a binary string s is mapped on a natural number, represented in a binary form, by putting 1 in front of the binary string s.*

Let $S = \{convert(\langle M\rangle) \mid \langle M\rangle \in L_\varnothing\}$ be our set of natural numbers representing $L_\varnothing$. Let $L_S = \{a^n b \mid n \in S\}$. This language clearly is not recursively enumerable. If it was, we could construct a Turing machine $M^\varnothing$ accepting $L_\varnothing$ using a reduction depicted in Figure 3.3. This $M^\varnothing$ would work this way: Let A_S be a Turing machine accepting L_S. Our $M^\varnothing$ takes its input $\langle M\rangle$ and transforms it to $w = a^n b$, where $n = convert(\langle M\rangle)$. This is possible to be carried out algorithmically. This w is then presented to A_S. If A_S accepts w, $M^\varnothing$ accepts $\langle M\rangle$, if it rejects or ends in a cycle, $M^\varnothing$ does the same on $\langle M\rangle$. This way $M^\varnothing$ accepts its input $\langle M\rangle$ only if $\langle M\rangle$ is a binary code of a Turing machine accepting an empty language. L_S thus cannot be recursively enumerable.

Let us now take an input permutation φ with an index permutation $\varphi_n = \varphi[n]$ for all $n \in \mathbb{N}$. For all $n \notin S$, we put $\varphi_{n+1}(i) = i$, for all $i \in \mathbb{Z}_n$. For all $n \in S$ we put $\varphi_{n+1}(i) = i - 1 \pmod{n}$, for all $i \in \mathbb{Z}_n$. φ_0 may be defined arbitrarily. A non-blind automaton with φ that applies this input permutation on its input in the beginning of the computation in case a is the first symbol of the input, and then checks if the input is of the form ba^k, for some $k \in \mathbb{N}$, can accept our L_S. This shows that some input permutation automata can even accept languages that do not belong to the recursively enumerable language family.

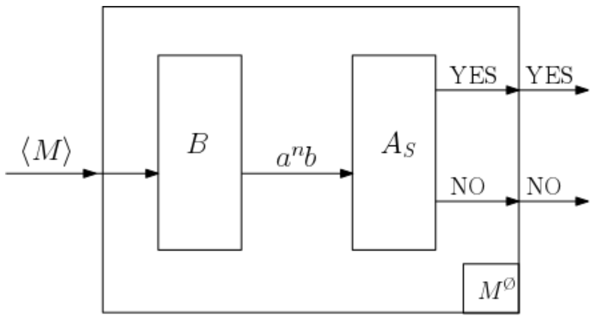


Figure 3.3: Turing machine $M^\varnothing$ from the Example 3.3, which gets an input $\langle M\rangle$, then uses an algorithm B, which transforms the input with the mapping *convert* into a suitable input for A_S. A_S accepts or declines the transformed input and so does $M^\varnothing$.

3.2 Regularity Preserving Permutations

In this section we shall identify a necessary and sufficient condition under which a family of languages accepted by k-permuting input permutation automata with a specific oblivious input permutation falls into the family of regular languages.

First, we shall prove some basic properties of oblivious input permutations. Next we shall define a normal form of input permutation automata that will serve us in the proof of our main results. Finally we shall prove that k-permuting input permutation automata equipped with an oblivious permutation φ always accept regular languages if and only if $\mathscr{R}$ is closed under φ^{-1}.

Definition 12. *Let φ be an oblivious input permutation and $\varphi_n = \varphi[n]$ for all $n \in \mathbb{N}$. Then we shall denote by φ^{-1} the inverse oblivious input permutation defined for all $n \in \mathbb{N}$ by $\varphi^{-1}[n] = \varphi_n^{-1}$.*

Thus if $w = a_0...a_{n-1}$ for $a_0, ..., a_{n-1} \in \Sigma$ and $\varphi = (\varphi_n)_{n\in\mathbb{N}}$ is an oblivious input permutation, then $\varphi^{-1}(w) = a_{\varphi_n^{-1}(0)}...a_{\varphi_n^{-1}(n-1)}$ and $\varphi^{-1}(L) = \{\varphi^{-1}(w) \mid w \in L\}$.

Lemma 4. *Let φ be an oblivious input permutation and L a language. Then $\varphi^{-1}(\varphi(L)) = \varphi(\varphi^{-1}(L)) = L$.*

Proof. We shall only prove the identity $\varphi^{-1}(\varphi(L)) = L$. The identity $\varphi(\varphi^{-1}(L)) = L$ shall follow by the latter via substituting φ^{-1} for φ, as it is clear that $(\varphi^{-1})^{-1} = \varphi$

"$\subseteq$": Let $w = a_0...a_{n-1} \in \varphi^{-1}(\varphi(L))$. From the definition of φ^{-1} we know that there exists some $v \in \varphi(L)$, such that $w = \varphi^{-1}(v)$. Let $v = c_0...c_{n-1}$. Similarly from the definition of φ, there exists $u \in L$ such that $v = \varphi(u)$. Let $u = b_0...b_{n-1}$. Then $v = \varphi(u) = \varphi(b_0...b_{n-1}) = b_{\varphi_n(0)}...b_{\varphi_n(n-1)}$, so $c_i = b_{\varphi_n(i)}$ for all $i \in \mathbb{Z}_n$. Moreover $w = \varphi^{-1}(v) = \varphi^{-1}(c_0...c_{n-1}) = c_{\varphi_n^{-1}(0)}...c_{\varphi_n^{-1}(n-1)} = b_{\varphi_n(\varphi_n^{-1}(0))}...b_{\varphi_n(\varphi_n^{-1}(n-1))}$, and so $a_i = c_{\varphi^{-1}(i)} = b_{\varphi_n(\varphi_n^{-1}(i))}$, for all $i \in \mathbb{Z}_n$. As a result, we get $a_i = b_{\varphi_n(\varphi_n^{-1}(i))} = b_i$, for all $i \in \mathbb{Z}_n$, and so $w = u$ and $w \in L$.

"$\supseteq$": Let $w = a_0...a_{n-1} \in L$. If we apply φ on w, we get $\varphi(w) = \varphi(a_0...a_{n-1}) = a_{\varphi_n(0)}...a_{\varphi_n(n-1)}$. Next we apply φ^{-1} on $\varphi(w)$, we get $\varphi^{-1}(\varphi(w)) = \varphi^{-1}(a_{\varphi_n(0)}...a_{\varphi_n(n-1)}) = a_{\varphi_n(\varphi_n^{-1}(0))}...a_{\varphi_n(\varphi_n^{-1}(n-1))}$ and as a result, we get $a_{\varphi_n^{-1}(\varphi_n(0))}...a_{\varphi_n^{-1}(\varphi_n(n-1))} = a_0...a_{n-1}$, which means that $\varphi^{-1}(\varphi(w)) = w$ and so w is in $\varphi^{-1}(\varphi(L))$. □

Lemma 5. *Let φ be an oblivious input permutation. Then for all $L \subseteq \Sigma^*$:*

$$\{w \in \Sigma^* \mid \varphi(w) \in L\} = \varphi^{-1}(L).$$

Proof. Let φ be an oblivious input permutation and $L \subseteq \Sigma^*$.

"$\subseteq$": Let $w = a_0...a_{n-1} \in \Sigma^*$ be such that $\varphi(w) \in L$. Thus there exists $u = b_0...b_{n-1} \in L$ such that $u = \varphi(w)$. From the last equality we get $b_0...b_{n-1} = \varphi(a_0...a_{n-1}) =$

$a_{\varphi_n(0)}...a_{\varphi_n(n-1)}$. If we apply φ^{-1} on u we get $\varphi^{-1}(b_0...b_{n-1}) = b_{\varphi_n^{-1}(0)}...b_{\varphi_n^{-1}(n-1)} = a_{\varphi_n(\varphi_n^{-1}(0))}...a_{\varphi_n(\varphi_n^{-1}(n-1))} = a_0...a_{n-1}$, which means that $w \in \varphi^{-1}(L)$.

"$\supseteq$": Let $w \in \varphi^{-1}(L)$. This means that there exists $u \in L$ such that $w = \varphi^{-1}(u)$. If we apply φ on both sides of the last equality, we get $\varphi(w) = \varphi(\varphi^{-1}(u))$. We need to show that $\varphi(\varphi^{-1}(u)) = u$. Let $u = a_0...a_{n-1}$ and $v = b_0...b_{n-1}$ so that $v = \varphi^{-1}(u) = a_{\varphi_n^{-1}(0)}...a_{\varphi_n^{-1}(n-1)}$. If we apply φ on v, we get $\varphi(v) = \varphi(b_0...b_{n-1}) = b_{\varphi_n(0)}...b_{\varphi_n(n-1)} = a_{\varphi_n^{-1}(\varphi_n(0))}...a_{\varphi_n^{-1}(\varphi_n(n-1))} = a_0...a_{n-1}$. We proved that $\varphi(\varphi^{-1}(u)) = u$ and so $\varphi(w) \in L$. $\square$

Lemma 6. *$\mathscr{L}_{BL}(\varphi, 0) \subseteq \mathscr{R}$ and $\mathscr{L}(\varphi, 0) \subseteq \mathscr{R}$.*

Proof. The proof is trivial since $\mathscr{L}(\varphi, 0)$ ($\mathscr{L}_{BL}(\varphi, 0)$) is a family of languages for which there exists a (blind) automaton with an oblivious input permutation, such that it does not apply the input permutation at all, which is the same as a family of languages accepted by finite automata, in other words $\mathscr{R}$. $\square$

The following definition describes a normal form of input permutation automata, which allows the automata to divide their states into two sets. This partition assures that once the input-transforming transition function is defined from one state, the ordinary transition function may not be defined from this state, which will later help us in the proof of our theorems.

Definition 13. *Let $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ be a (blind) input permutation automaton. Let $Q = Q_1 \cup Q_2$, where $Q_1 = \{q \mid \exists a \in \Sigma : \Delta(q, a) \neq \varnothing\}$ ($Q_1 = \{q \mid \Delta(q) \neq \varnothing\}$ for a blind automaton) and $Q_2 = \{q \mid \exists a \in \Sigma : \delta(q, a) \neq \varnothing\}$. The automaton A is said to be in state-disjoint normal form if $Q_1 \cap Q_2 = \varnothing$, in other words, once the input-transforming transition function is defined from some state q, the ordinary transition function may not be defined from this state and vice versa. We shall write Q^{Δ} for Q_1 and Q^{δ} for Q_2.*

We shall now prove that this is indeed a normal form.

Lemma 7. *For each blind input permutation automaton $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ there exists a blind input permutation automaton A' in state-disjoint normal form, such that $L(A') = L(A)$. Clearly, if A is a k-permuting automaton for some $k \in \mathbb{N}$, then A' is a k-permuting automaton as well.*

Proof. Let $A = (Q, \Sigma, \delta, \Delta, \varphi, i, F)$ be a blind input permutation automaton. Let

$A' = (Q', \Sigma, \delta', \Delta', \varphi, i_\delta, F')$ be such that:

$$\begin{aligned}
Q' &= \{q_\Delta, q_\delta \mid q \in Q\},\\
F' &= \{q_\delta \mid q \in F\},\\
\Delta'(q_\Delta) &= \{p_\delta \mid p \in \Delta(q)\} \ (\forall q \in Q),\\
\delta'(q_\delta, a) &= \{p_\delta \mid p \in \delta(q, a)\} \ (\forall q \in Q,\ a \in \Sigma),\\
\delta'(q_\delta, \varepsilon) &= \{q_\Delta\} \cup \{p_\delta \mid p \in \delta(q, \varepsilon)\} \ (\forall q \in Q).
\end{aligned}$$

It is easy to see that A' is in state-disjoint normal form and that $L(A') = L(A)$. Clearly, if A is k-permuting $k \in \mathbb{N}$, then A' is k-permuting as well. □

Lemma 8. *For each input permutation automaton $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ there exists an input permutation automaton A' in state-disjoint normal form, such that $L(A') = L(A)$. Moreover, if A is a k-permuting automaton for some $k \in \mathbb{N}$, then A' is a k-permuting automaton as well.*

Proof. Let $A = (Q, \Sigma, \delta, \Delta, \varphi, i, F)$ be our input permutation automaton. Let $A' = (Q', \Sigma, \delta', \Delta', \varphi, i_\delta, F')$ be such that:

$$\begin{aligned}
Q' &= \{q_\Delta, q_\delta \mid q \in Q\}\\
F' &= \{q_\delta \mid q \in F\}\\
\Delta'(q_\Delta, a) &= \{p_\delta \mid p \in \Delta(q, a)\} \ (\forall q \in Q,\ a \in \Sigma \cup \{\varepsilon\})\\
\delta'(q_\delta, a) &= \{p_\delta \mid p \in \delta(q, a)\} \ (\forall q \in Q,\ a \in \Sigma)\\
\delta'(q_\delta, \varepsilon) &= \{q_\Delta\} \cup \{p_\delta \mid p \in \delta(q, \varepsilon)\} \ (\forall q \in Q)
\end{aligned}$$

It is easy to see that A' is in state-disjoint normal form and that $L(A') = L(A)$. Moreover, if A is k-permuting $k \in \mathbb{N}$, then A' is k-permuting as well. □

The following theorems tell us what condition the input permutation has to satisfy in order to preserve regularity when applied at most k times during each computation for some k in $\mathbb{N}$.

In what follows, we shall use the notation $L_{p,q}(A) = \{w \in \Sigma^* \mid (p, w) \vdash_A^* (q, \varepsilon)\}$, where $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$ is an input permutation automaton and $q, p \in Q$.

Theorem 18. *Let φ be an oblivious input permutation and $k \in \mathbb{N}$. If $\mathscr{R}$ is closed under φ^{-1}, then $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{R}$. Conversely, if $k \geq 1$ and $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{R}$, then $\mathscr{R}$ is closed under φ^{-1}.*

Proof. We shall prove the first part of this statement by induction. The statement is trivial for $k = 0$, as we have shown in Lemma 6.

Let us now suppose that the statement holds for some $k \in \mathbb{N}$. We want to show that it is also true for $k+1$. Let $L \in \mathscr{L}_{BL}(\varphi, k+1)$. From the definition of $\mathscr{L}_{BL}(\varphi, k+1)$

we know that there exists a $(k+1)$-permuting blind input permutation automaton $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$, such that $L(A) = L$. By Lemma 7 we can assume that A is in state-disjoint normal form.

As A is k-permuting, it is clear that a number $\#(q)$ in $\{0, ..., k+1\}$ can be associated with each q in Q so that $\#(q)$ is the maximum number of applications of φ over all computations of A starting in q. Without loss of generality, it can be assumed that $\#(q_0) = k+1$ (otherwise $L(A)$ belongs to $\mathscr{L}_{BL}(\varphi, k)$, and it follows by the induction hypothesis that $L(A)$ is in $\mathscr{R}$).

The automaton is in state-disjoint normal form by assumption, thus Q is partitioned to two disjoint subsets Q^Δ and Q^δ as in Definition 13. For the purpose of this proof we shall use the following notation:

$$\begin{aligned} Q[i,\delta] &= \{q \in Q^\delta \mid \#(q) = i\} \ (\forall i \in \{0, ..., k+1\}) \\ Q[i,\Delta] &= \{q \in Q^\Delta \mid \#(q) = i\} \ (\forall i \in \{0, ..., k+1\}) \\ Q[i] &= Q[i,\Delta] \cup Q[i,\delta] \ (\forall i \in \{0, ..., k+1\}) \\ F[i] &= \{q \in F \mid \#(q) = i\} \ (\forall i \in \{0, ..., k+1\}) \end{aligned}$$

If we analyze the accepting computations of the automaton A, it follows by Lemma 5 that

$$L = \bigcup_{f \in F[0]} L_{q_0,f}(A) \cup \bigcup_{p \in Q[0,\Delta]} L_{q_0,p}(A)\varphi^{-1}(L'[p])$$

where

$$L'[p] = \bigcup_{\substack{r \in \Delta(p) \\ f \in F}} L_{r,f}(A).$$

We can construct a blind input permutation automaton A'_p for each $p \in Q[0,\Delta]$, such that $L(A'_p) = L'[p]$. We define $A'_p = (Q', \Sigma, \delta', \Delta', \varphi, q'_0, F')$, where q'_0 is a new state and

$$\begin{aligned} Q' &= \{q'_0\} \cup Q \setminus Q[0] \\ F' &= F \setminus F[0] \\ \Delta'(q) &= \Delta(q) \ (\forall q \in Q \setminus Q[0]) \\ \delta'(q,a) &= \delta(q,a) \ (\forall q \in Q \setminus Q[0], \ a \in \Sigma \cup \{\varepsilon\}) \\ \delta'(q'_0, \varepsilon) &= \Delta(p) \\ \delta'(q'_0, a) &= \varnothing \ (\forall a \in \Sigma) \\ \Delta'(q'_0) &= \varnothing \end{aligned}$$

This automaton is clearly well defined and it can apply the input permutation at maximum k times. It begins each computation in q'_0 and transitions on ε to one of the states $r \in \Delta(p)$. Clearly $r \in Q[1,\delta]$. From there the automaton continues the computation according to δ and Δ functions. Clearly $L(A'_p) = L'[p]$ and so $L'[p]' \in \mathscr{L}_{BL}(\varphi, k)$. From the induction hypothesis we know that $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{R}$. Since the languages $L_{q_0,f}(A)$ for all $f \in F[0]$ are clearly regular and $\mathscr{R}$ is closed under union and concatenation, the language L belongs to $\mathscr{R}$ if $\mathscr{R}$ is closed under φ^{-1}.

We shall prove by contradiction the second part of the statement. Let φ be an oblivious input permutation. Suppose that $k \geq 1$, $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{R}$ and $\mathscr{R}$ is not closed under φ^{-1}. Let $L \in \mathscr{R}$ be such that $\varphi^{-1}(L) \notin \mathscr{R}$. Since $L \in \mathscr{R}$, there exists a finite automaton $A = (Q, \Sigma, \delta, q_0, F)$, such that $L(A) = L$. We can construct a blind input permutation automaton $A' = (Q \cup \{q'_0\}, \Sigma, \delta, \Delta, \varphi, q'_0, F)$ with our input permutation φ, which applies φ as the first thing in the beginning of the computation and moves to q_0. From there it follows the ordinary transition function δ. It follows by Lemma 5 that $L(A') = \varphi^{-1}(L)$. Moreover $L(A') \in \mathscr{L}_{BL}(\varphi, 1) \subseteq \mathscr{L}_{BL}(\varphi, k)$. Since $\varphi^{-1}(L) \notin \mathscr{R}$ also $\mathscr{L}_{BL}(\varphi, k) \nsubseteq \mathscr{R}$, which is in contradiction with our assumptions. From this we get that if $\mathscr{L}_{BL}(\varphi, k) \subseteq \mathscr{R}$, then $\mathscr{R}$ is closed under φ^{-1}. □

Theorem 19. *Let φ be an oblivious input permutation and $k \in \mathbb{N}$. If $\mathscr{R}$ is closed under φ^{-1}, then $\mathscr{L}(\varphi, k) \subseteq \mathscr{R}$. Conversely, if $k \geq 1$ and $\mathscr{L}(\varphi, k) \subseteq \mathscr{R}$, then $\mathscr{R}$ is closed under φ^{-1}.*

Proof. The proof of this theorem will be similar to the one of the Theorem 18. We shall use induction to prove the first part of the statement. For $k = 0$ is the statement trivial, as we have shown in Lemma 6.

Let us now suppose that the statement holds for some $k \in \mathbb{N}$. We want to show that it is also true for $k+1$. Let $L \in \mathscr{L}(\varphi, k+1)$. From the definition of $\mathscr{L}(\varphi, k+1)$ we know that there exists a $(k+1)$-permuting input permutation automaton $A = (Q, \Sigma, \delta, \Delta, \varphi, q_0, F)$, such that $L(A) = L$. By Lemma 8 we can assume that A is in state-disjoint normal form.

As A is k-permuting, it is clear that a number $\#(q)$ in $\{0, ..., k+1\}$ can be associated with each q in Q so that $\#(q)$ is the maximum number of applications of φ over all computations of A starting in q. Without loss of generality, it can be assumed that $\#(q_0) = k+1$ (otherwise $L(A)$ belongs to $\mathscr{L}(\varphi, k)$, and it follows by the induction hypothesis that $L(A)$ is in $\mathscr{R}$).

The automaton is in state-disjoint normal form by assumption, thus Q is partitioned to two disjoint subsets Q^{Δ} and Q^{δ} as in Definition 13. For the purpose of this proof we shall use the notation $Q[i,\delta]$, $Q[i,\Delta]$, $Q[i]$, $F[i]$ defined in the same way as in the proof of the Theorem 18.

If we analyze the accepting computations of the automaton A, it follows by Lemma 5 that

$$L = \bigcup_{f \in F[0]} L_{q_0,f}(A) \cup \bigcup_{p \in Q[0,\Delta]} L_{q_0,p}(A) \bigcup_{a \in \Sigma \cup \{\varepsilon\}} \left(\varphi^{-1}(L'[p,a]) \cap a\Sigma^*\right)$$

where

$$L'[p,a] = \bigcup_{\substack{r\in\Delta(p,a)\\ f\in F}} L_{r,f}(A).$$

We can construct an input permutation automaton $A'_{p,a}$ for each $p \in Q[0,\Delta]$ and $a \in \Sigma \cup \{\varepsilon\}$, such that $L(A'_{p,a}) = L'[p,a]$. We define $A'_{p,a} = (Q', \Sigma, \delta', \Delta', \varphi, q'_0, F')$, where q'_0 is a new state and

$$\begin{aligned}
Q' &= \{q'_0\} \cup Q \setminus Q[0] \\
F' &= F \setminus F[0] \\
\Delta'(q,b) &= \Delta(q,b)\ (\forall q \in Q \setminus Q[0], b \in \Sigma \cup \{\varepsilon\}) \\
\delta'(q,b) &= \delta(q,b)\ (\forall q \in Q \setminus Q[0],\ b \in \Sigma \cup \{\varepsilon\}) \\
\delta'(q'_0,\varepsilon) &= \Delta(p,a) \\
\delta'(q'_0,b) &= \varnothing\ (\forall b \in \Sigma) \\
\Delta'(q'_0,b) &= \varnothing\ (\forall b \in \Sigma \cup \{\varepsilon\})
\end{aligned}$$

This automaton is clearly well defined and it can apply the input permutation at maximum k times. It begins each computation in q'_0 and transitions on ε to one of the states $r \in \Delta(p,a)$. Clearly $r \in Q[1,\delta]$. From there the automaton continues the computation according to δ and Δ functions. Clearly $L(A'_{p,a}) = L'[p,a]$ and so $L'[p,a] \in \mathscr{L}(\varphi,k)$. From the induction hypothesis we know that $\mathscr{L}(\varphi,k) \subseteq \mathscr{R}$. Since the languages $a\Sigma^*$ and $L_{q_0,f}(A)$ for all $f \in F[0]$ are clearly regular and $\mathscr{R}$ is closed under union, intersection and concatenation, the language L belongs to $\mathscr{R}$ if $\mathscr{R}$ is closed under φ^{-1}.

In the proof of the second part of the statement of the Theorem 18, we have shown that if $k \geq 1$ and $\mathscr{L}_{BL}(\varphi,k) \subseteq \mathscr{R}$, then $\mathscr{R}$ is closed under φ^{-1}. By Proposition 2 we know that $\mathscr{L}_{BL}(\varphi,k) \subseteq \mathscr{L}(\varphi,k)$. This way, if the statement applies to $\mathscr{L}_{BL}(\varphi,k)$, it also applies to $\mathscr{L}(\varphi,k)$. □

Theorem 20. *Let φ be an oblivious input permutation. Then the following statements are equivalent:*

(i) $\mathscr{L}_{BL}(\varphi,k) = \mathscr{R}$ for all $k \in \mathbb{N}$.

(ii) $\mathscr{L}(\varphi,k) = \mathscr{R}$ for all $k \in \mathbb{N}$.

(iii) $\mathscr{R}$ is closed under φ^{-1}.

Proof. We have proved in Theorems 18 and 19 that the inclusion "$\subseteq$" in (i) and (ii) is equivalent to (iii). The reversed inclusion is trivial, we would just take $k = 0$ and apply Proposition 1. Thus we have proved the equivalences $(i) \Leftrightarrow (iii)$ and $(ii) \Leftrightarrow (iii)$. From $(i) \Leftrightarrow (iii) \Leftrightarrow (ii)$ we get $(i) \Leftrightarrow (ii)$. □

Conclusion

This thesis has focused on automata with an additional possibility to use an input transformation. A question on the computational power of these models has been raised: do they still accept regular languages or can they also accept languages from different levels of the Chomsky hierarchy?

In Chapter 1 we have summarized the past research on languages accepted by automata with some specific input transformations. In Chapter 2 we have provided mathematical basis for a study of input transformation automata at a general level by defining an input permutation and a specialization of this concept – an oblivious input permutation. The former is a mapping of words which preserves the number of occurrences of each symbol. The latter, in addition, does not depend on specific symbols but only on their positions in the word. Next we have defined two variants of input permutation automata, a general one and a blind one. Both have a possibility to use a specific input permutation during their computation. The difference between them is that the former may peek at the first symbol of the input and decide according to that if it uses the input permutation, of continues without using it, and the latter may perform the input permutation only according to the current state.

Later we have defined what we have called k-permuting input permutation automata, which are automata that use the input permutation at most k times during the computation on any input. We have showed, that for each input permutation automaton A, we can construct a k-permuting automaton A_k, such that $L(A_k) = L^{(k)}(A)$. Finally, we have defined a family of languages accepted by blind input permutation automata with an input permutation φ denoted by $\mathscr{L}_{BL}(\varphi)$ and a family of languages accepted by blind k-permuting input permutation automata with an input permutation φ and some natural k denoted by $\mathscr{L}_{BL}(\varphi, k)$. Similarly we have defined $\mathscr{L}(\varphi)$ and $\mathscr{L}(\varphi, k)$ for non-blind input permutation automata.

In the last chapter, we have presented some examples of oblivious input permutations, which can transform some of the languages which are not regular, not context-free, or not even recursively enumerable into languages recognizable by finite automata. This way we have showed how powerful the input permutation automata can be. Next, for the purpose of our future proofs, we have showed that $\varphi^{-1}(\varphi(L)) = \varphi(\varphi^{-1}(L)) = L$ and $\{w \in \Sigma^* \mid \varphi(w) \in L\} = \varphi^{-1}(L)$ for an oblivious input permutation φ and all

$L \subseteq \Sigma^*$. Later we have defined a state-disjoint normal form, which ensures that the set of states, from which the ordinary transition function of an automaton is defined is disjoint from the set of states, from which the the input-transforming function is defined. This has helped us in the proof of our final theorems.

The main results of this thesis are the final theorems. They tell us about the necessary and sufficient condition under which the families $\mathscr{L}_{BL}(\varphi, k)$ and $\mathscr{L}(\varphi, k)$, where φ is an oblivious input permutation and k a natural number, are equal to $\mathscr{R}$. The condition states that it must be satisfied that $\mathscr{R}$ is closed under φ^{-1}.

Some questions for further research can be raised: are the families $\mathscr{L}_{BL}(\varphi, k)$ and $\mathscr{L}(\varphi, k)$ equal if φ is a non-oblivious input permutation? What is the relationship between the families $\mathscr{L}_{BL}(\varphi)$ and $\mathscr{L}(\varphi)$?

Bibliography

[1] Henning Bordihn, Markus Holzer, and Martin Kutrib. Input reversals and iterated pushdown automata: a new characterization of Khabbaz geometric hierarchy of languages. In *International Conference on Developments in Language Theory (DLT 2004)*, pages 102–113. Springer, 2004.

[2] Henning Bordihn, Markus Holzer, and Martin Kutrib. Revolving-input finite automata. In *International Conference on Developments in Language Theory (DLT 2005)*, pages 168–179. Springer, 2005.

[3] Henning Bordihn, Markus Holzer, and Martin Kutrib. Hybrid extended finite automata. In *Implementation and Application of Automata (CIAA 2006)*, pages 34–45. Springer, 2006.

[4] Henning Bordihn, Markus Holzer, and Martin Kutrib. Hairpin finite automata. In *International Conference on Developments in Language Theory (DLT 2007)*, pages 108–119, 2007.

[5] Michael A Harrison. *Introduction to formal language theory.* Addison-Wesley Longman Publishing Co., Inc., 1978.

[6] Markus Holzer and Martin Kutrib. Gaining power by input operations: finite automata and beyond. In *International Conference on Implementation and Application of Automata (CIAA 2011)*, pages 16–29. Springer, 2011.

[7] John E Hopcroft. *Introduction to automata theory, languages, and computation.* Pearson Education India, 2008.

[8] Dexter C Kozen. *Automata and computability.* Springer Science & Business Media, 2012.