

COMENIUS UNIVERSITY, BRATISLAVA  
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

# COMPLEXITY OF LANGUAGE TRANSFORMATIONS

DIPLOMA THESIS

NOVEMBER 19, 2014

2014

Boris Vida

COMENIUS UNIVERSITY, BRATISLAVA  
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

# COMPLEXITY OF LANGUAGE TRANSFORMATIONS

DIPLOMA THESIS

Study programme: Computer Science  
Study field: 2508 Computer Science, Informatics  
Department: Department of Computer Science  
Supervisor: Prof. RNDr. Branislav Rován, PhD.

Bratislava, 2014

Boris Vida



Comenius University in Bratislava  
Faculty of Mathematics, Physics and Informatics

---

## THESIS ASSIGNMENT

**Name and Surname:**

**Study programme:**

**Field of Study:**

**Type of Thesis:**

**Language of Thesis:**

**Title:**

**Aim:**

**Supervisor:**

**Department:**

**Assigned:**

**Approved:**

Guarantor of Study Programme

.....  
Student

.....  
Supervisor



Univerzita Komenského v Bratislave  
Fakulta matematiky, fyziky a informatiky

---

## ZADANIE ZÁVEEEČNEJ PEÁCE

**Meno a priezvisko študenta:**

**Študijný program:**

**Študijný odbor:**

**Typ záverečnej práce:**

**Jazyk záverečnej práce:**

**Názov:**

**Cieľ:**

**Vedúci:**

**Katedra:**

**Vedúci katedry:**

**Dátum zadania:**

**Dátum schválenia:**

garant študijného programu

.....  
študent

.....  
vedúci práce

# Acknowledgement

I would like to express gratitude to my supervisor Prof. RNDr. Branislav Rován, PhD. for his help and advices by writing of this thesis. I would also like to thank my family and friends for their support during my studies.

# Abstrakt

ToDo: Abstrakt

**KEYWORDS:** transformácie jazykov, popisná zložitosť, a-prekladač

# Abstract

ToDo: Abstract

**KEYWORDS:** language transformations, descriptive complexity, a-transducer

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Preliminaries</b>	<b>2</b>
1.1 Basic concepts and notation . . . . .	2
1.2 Transformation models . . . . .	3
1.3 Complexity, advisors and decomposition . . . . .	6
<b>2 Current State of Research</b>	<b>8</b>
2.1 Basic Properties of A-transducers . . . . .	8
2.2 State Complexity of Finite State Devices . . . . .	10
2.2.1 Finite State Automata . . . . .	11
2.2.2 Sequential Transducers . . . . .	12
2.3 Decompositions of finite automata . . . . .	13
<b>3 Complexity of A-transducers</b>	<b>14</b>
3.1 "Modular-counting" languages . . . . .	14
3.2 Common transformations . . . . .	17
<b>4 Foreign advisors</b>	<b>18</b>
4.1 Description of the framework . . . . .	18
4.2 T-decomposable and T-undecomposable languages . . . . .	19
4.3 Closure properties . . . . .	21
<b>Conclusion</b>	<b>22</b>



# Introduction

<sup>the past</sup>  
In ~~last~~ fifty years, <sup>A</sup> a number of models for realization of language transformations were introduced. Most of them can be described as an extension of a finite automaton, which has been ~~added~~ <sup>augmented by</sup> an output tape. ~~and the right side of the transition function consists not only of states, but also of substrings of an output alphabet.~~ Such models are, e. g., Moore and Mealy machines, sequential transducers, general sequential machines, <sup>A</sup> ~~A~~-transducers, etc.

In our thesis, we would like to investigate some complexity properties of these transformation models, with emphasis on <sup>A</sup> ~~A~~-transducers.

We assume, that the reader is <sup>familiar</sup> ~~acknowledged~~ with <sup>the</sup> ~~basic~~ concepts of formal languages. If this is not the case, we recommend to obtain this understanding from [1].

# Chapter 1

## Preliminaries

In this section, <sup>we present</sup> ~~we would like to clarify~~ some basic notations <sup>and terminology</sup> used in our thesis.

### 1.1 Basic concepts and notation

**Notation.** We denote

- by  $\epsilon$  an empty string,
- by  $|w|$  the length of a word  $w$  ( $|\epsilon| = 0$ ),
- by  $|A|$  the number of elements of a finite set (or a finite language)  $A$ ,
- by  $\#_a(w)$  the number of <sup>occurrences of the</sup> symbols  $a$  in a word  $w$ ,
- if  $u \equiv a_1a_2\dots a_m$ ,  $v \equiv b_1b_2\dots b_n$ , then by  $u.v$  or simply  $uv$  we denote a word  $a_1a_2\dots a_mb_1b_2\dots b_n$ ,
- by  $A^+$  we denote a transitive closure of  $A$ , by  $A^*$  a reflexive-transitive closure of  $A$ .

**Definition.** A family of languages is an ordered pair  $(\Sigma, \mathcal{L})$ , such that

1.  $\Sigma$  is an infinite set of symbols
2. every  $L \in \mathcal{L}$  is a language over some finite set  $\Sigma^* \subset \Sigma$
3.  $L \neq \emptyset$  for some  $L \in \mathcal{L}$

**Definition.** A homomorphism is a function  $h : \Sigma_1^* \rightarrow \Sigma_2^*$ , such that

$$h(uv) = h(u)h(v)$$

**Notation.** If  $\forall w \neq \epsilon : h(w) \neq \epsilon$ , we call  $h$  an  $\epsilon$ -free homomorphism and denote it by  $h_\epsilon$ .

**Notation.** For a set  $A$ ,  $2^A$  is the set of all subsets of  $A$ .

**Definition.** An *inverse homomorphism* is a function  $h^{-1} : \Sigma_1^* \rightarrow 2^{\Sigma_2^*}$ , such that  $h$  is a homomorphism and

$$h^{-1}(u) = \{v | h(v) = u\}$$

**Definition.** A family of languages is called a (*full*) *trio*, if it is closed under  $\epsilon$ -free (arbitrary) homomorphism, inverse homomorphism and intersection with a regular set.

**Definition.** A (full) trio is called a (*full*) *semi-AFL*, if it is closed under union.

**Definition.** A (full) semi-AFL is called a (*full*) *AFL*, if it is closed under concatenation and  $+$ .

We now characterize two common<sup>ly</sup> used families of language, included in<sup>the</sup> a Chomsky hierarchy, to make clear the notation in this thesis.

**Notation.** The family of *regular languages*, denoted by  $\mathcal{R}$ , is the family of all languages generated by a type 3 grammar or accepted by a deterministic finite automaton (see e. g. [1] for full definition).

**Notation.** The family of *context free languages*, denoted by  $\mathcal{L}_{CF}$ , is the family of all languages generated by a type 2 grammar or accepted by a pushdown finite automaton (see e. g. [1] for full definition).

## 1.2 Transformation models

~~We shall now~~  
~~Now we would like to~~ define some of the models mentioned in the Introduction. Although the central point of our interest is an  $\mathcal{A}$ -transducer, we also introduce the definitions of other models, which will be used in the next chapters, because they can give us an insight of language transformations in general and many of the concepts used in results involving them can be put to use by examination of  $\mathcal{A}$ -transducers.

Since A-transducer is the most general type of transducer, we define it first and then we only specify the differences between A-transducers and other models.

**Definition.** An *A-transducer* is a 6-tuple  $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ , where

- $K$  is a finite set of states,
- $\Sigma_1$  and  $\Sigma_2$  are the input and output alphabet, respectively,
- $H \subseteq K \times \Sigma_1^* \times \Sigma_2^* \times K$  is the transition function, where  $H$  is finite,
- $q_0 \in K$  is the initial state,
- $F \subseteq K$  is a set of accepting states.

If  $H \subseteq K \times \Sigma_1^* \times \Sigma_2^+ \times K$ , we call  $M$  an  $\epsilon$ -free A-transducer.

**Definition.** If  $H \subseteq K \times \Sigma_1 \cup \{\epsilon\} \times \Sigma_2 \cup \{\epsilon\} \times K$ , the corresponding A-transducer is called *1-bounded*.

**Definition.** <sup>A</sup> The *configuration* of an A-transducer is a triple  $(q, u, v)$ , where  $q \in K$  is a current internal state,  $u \in \Sigma_1^*$  is the remaining part of the input and  $v$  is the already written output.

**Definition.** A *computational step* is a relation  $\vdash$  on configurations defined as follows:

$$(q, xu, v) \vdash (p, u, vy) \Leftrightarrow (q, x, y, p) \in H.$$

**Definition.** <sup>a</sup> An *image* of language  $L$  by <sup>an</sup> A-transducer  $M$  is a set

$$M(L) = \{w \mid \exists u \in L, q_F \in F; (q_0, u, \epsilon) \vdash^* (q_F, \epsilon, w)\}$$

**Definition.** For  $i = 0, 1, 2, 3$ ,  $w \equiv (x_0, x_1, x_2, x_3) \in H$ , we define  $pr_i(w) = x_i$  and call  $pr_i$  an *i-th projection*.

**Definition.** A *computation* of an A-transducer  $M$  is a word  $h_0 h_1 \dots h_m \in H^*$ , such that

1.  $pr_0(h_0) = q_0$  ( $q_0$  is the initial state of  $M$ ),
2.  $\forall i : pr_3(h_i) = pr_0(h_{i+1})$
3.  $pr_3(h_m) \in F$

**Notation.** We denote a language of all computations of  $M$  by  $\Pi_M$ . Note, that  $\Pi_M$  is regular ([2]).

**Definition.** Alternatively, we can define an image of  $L$  by an A-transducer  $M$  as

$$M(L) = \{pr_2(pr_1^{-1}(w) \cap \Pi_M | w \in L\}.$$

**Definition.** An *A-transduction* is a function  $\Phi : \Sigma_1^* \rightarrow 2^{\Sigma_2^*}$  defined as follows:

$$\forall x \in \Sigma_1^* : \Phi(x) = \{M(x)\}.$$

We have described the core model of our thesis, namely an A-transducer, and now we define two similar, but simpler models.

**Definition.** A *sequential transducer* is a 7-tuple  $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0, F)$ , where

- $K, \Sigma_1, \Sigma_2, q_0, F$  are like in an A-transducer,
- $\delta$  is a transition function, which maps  $K \times \Sigma_1 \rightarrow K$ ,
- $\sigma$  is an output function, which maps  $K \times \Sigma_1 \rightarrow \Sigma_2^*$ .

A sequential transducer can be seen as a "deterministic" 1-bounded A-transducer, which set  $H$  fulfills following conditions:

1. for every couple  $(q, a) \in K \times \Sigma_1$ , there is exactly one element  $h \in H$ , such that  $pr_0(h) = q$  and  $pr_1(h) = a$ ,
2.  $\forall h \in H : pr_1(h) \neq \epsilon \wedge pr_2(h) \neq \epsilon$ .

**Notation.** By  $\hat{\delta}$  and  $\hat{\sigma}$  we denote an extension of  $\delta$  ( $\sigma$ ) on  $K \times \Sigma_1^*$ , defined recursively as  $\forall q \in K, w \in \Sigma_1^*, a \in \Sigma_1$  :

- $\hat{\delta}(q, a) = \delta(q, a), \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a),$
- $\hat{\sigma}(q, a) = \sigma(q, a), \hat{\sigma}(q, wa) = \sigma(\hat{\delta}(q, w), a).$

We omit the definitions of a configuration, computational step and image related to sequential transducers, since they are very similar to the A-transducer.

**Definition.** A *sequential function* is a function represented by a sequential transducer. Formally, if  $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0, F)$  is a sequential transducer, then

$$\forall w \in \Sigma_1^*, \text{ s. t. } \hat{\delta}(q_0, w) \in F: f_M(w) = \hat{\sigma}(q_0, w).$$

We conclude this section with a definition of one more model, which can be viewed as a generalization of a sequential transducers.

**Definition.** A *generalized sequential machine (gsm)* is a 6-tuple  $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0)$ , where  $K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0$  are as in sequential transducer.

As one can see, a generalized sequential machine is a sequential transducer with  $F \equiv K$  and therefore all other concepts are defined just like in a sequential transducer.

**Notation.** A sequential function described by a generalized sequential machine is called a *gsm mapping*.

### 1.3 Complexity, advisors and decomposition

In this section we define the concept of advisors and decompositions.

**Definition.** The *state complexity* of an A-transducer  $T = (Q, \Sigma_1, \Sigma_2, H, q_0, F)$  (a finite automaton  $A = (Q, \Sigma, \delta, q_0, F)$ ), denoted by  $\mathcal{C}_{state}(T)$  ( $\mathcal{C}_{state}(A)$ ), is the number of its states. Formally

$$\mathcal{C}_{state}(T) = |Q|.$$

**Definition.** The *state complexity* of a regular language  $L$ , denoted by  $\mathcal{C}_{state}(L)$ , is the state complexity of its minimal deterministic finite automaton. Formally

$$\mathcal{C}_{state}(L) = \min\{\mathcal{C}_{state}(A) | L(A) = L\}.$$

If  $L$  is not regular, then  $\mathcal{C}_{state}(L) = \infty$ .

**Definition.** In a similar way, we can define the *A-transducer state complexity* of a pair of languages  $L_1, L_2$ , denoted by  $\mathcal{C}_{state}(L_1, L_2)$ , as the state complexity of the minimal A-transducer, which translates language  $L_1$  to  $L_2$ . Formally

$$\mathcal{C}_{state}(L_1, L_2) = \min\{\mathcal{C}_{state}(T) | T(L_1) = L_2\}.$$

Note, that it is possible, that  $\mathcal{C}_{state}(L_1, L_2) \neq \mathcal{C}_{state}(L_2, L_1)$ .

and it may happen that  $\mathcal{C}_{state}(L_1, L_2) = \infty$   
(also possible  $L_1, L_2 \in \mathcal{Q}$ )

ToDo: definície podľa Gaziho diplomovky aj s citáciami

# Chapter 2

## Current State of Research

In this chapter, we would like to present some known results regarding transformation devices in general and their complexity aspects.

### 2.1 Basic Properties of A-transducers

This section contains few basic results from [2].

**Lemma 1.**  $\mathcal{R}$  and  $\mathcal{L}_{CF}$  are closed under a-transduction.

*Proof.* Let  $M$  be an A-transducer and  $L$  a regular (context-free) language. We use the alternative definition of the image  $L$ :

$$M(L) = \{pr_2(pr_1^{-1}(w) \cap \Pi_M) | w \in L\}$$

Since  $\Pi_M$  is regular and both classes, of regular and of context-free languages are closed under intersection with a regular language, homomorphism and inverse homomorphism ([1]), they are also closed under a-transduction.  $\square$

**Corollary 1.1.** Since sequential transducers and generalized sequential machines are just special forms of an A-transducer, this lemma also holds for these devices.

In previous chapter, we have defined a special class of 1-bounded A-transducers. Following theorem shows, that this limitation forms a normal form.

**Lemma 2.** Let  $M_1$  be an arbitrary A-transducer. Then there exists a 1-bounded A-transducer  $M_2$ , such that  $\forall L : M_1(L) = M_2(L)$ .





*Proof.* Let  $(q, u, v, p) \in H_1, u \equiv a_1 a_2 \dots a_m, v \equiv b_1 b_2 \dots b_n$ . Let  $m \geq n$  (for  $m < n$  the proof is very similar).  $M_2$  will have states  $q, q_{a_1}, q_{a_2}, \dots, q_{a_{n-1}}, q_{a_n} \equiv p$  and transitions in form  $(q_{a_i}, a_{i+1}, b_{i+1}, q_{a_{i+1}})$  for  $1 \leq i < n$ , resp.  $(q_{a_j}, a_{j+1}, \epsilon, q_{a_{j+1}})$  for  $n \leq j < m$ . This will be done for every  $h \in H$ . It is easy to see, that the a-transduction by  $M_1$  and  $M_2$  is the same and therefore  $\forall L : M_1(L) = M_2(L)$ .  $\square$

As one can see, this construction can increase the number of states of an A-transducer by a constant multiple. Sometimes it is more convenient to consider only 1-bounded A-transducer, since its complexity can be easier compared with other computational models.

In the next section, we quote results regarding the question, when it is possible to transform one language to another using an A-transducer. The next theorem gives us another view of this problem using the theory of language families.

**Lemma 3.** For every two ( $\epsilon$ -free) A-transducers  $M_1$  and  $M_2$  there exists an ( $\epsilon$ -free) A-transducer  $M_3$  such that  $\forall L : M_3(L) = M_2(M_1(L))$ .

*Proof.* We show just the idea of the proof: We may assume that  $M_1$  and  $M_2$  are 1-bounded.  $M_3$  simulates both of the A-transducers concurrently (so its internal state have the form  $(q \times p)$ ), reads the input according to transition function of  $M_1$  and writes the corresponding output of  $M_2$ , while the output of  $M_1$  forms the input of  $M_2$ . It is easy to see, that  $\forall L : M_3(L) = M_2(M_1(L))$ .  $\square$

**Lemma 4.** For every ( $\epsilon$ -free) homomorphism  $h : \Sigma_1^* \rightarrow \Sigma_2^*$  there is an ( $\epsilon$ -free) A-transducer  $M$ , such that  $\forall L : M(L) = h(L)$ .

*Proof.* The A-transducer  $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$  will look as follows:

- $K = F = \{q\}$ ,
- $q_0 = q$ ,
- $H = \{(q, a, h(a), q) | a \in \Sigma_1\}$ .  $\square$

**Lemma 5.** For every homomorphism  $h$  there is an A-transducer  $M$ , such that  $\forall L : M(L) = h^{-1}(L)$ .

*Proof.* As in previous Lemma, except  $H = \{(q, h(a), a, q) | a \in \Sigma_1\}$ .  $\square$

**Lemma 6.** For every  $R \in \mathcal{R}$ , there exists an  $\epsilon$ -free A-transducer  $M$ , such that  $M(L) = L \cap R$ .

*Proof.* Let  $A = (K, \Sigma, q_0, \delta, F)$  be a non-deterministic finite automaton, such that  $L(A) = R$ . Then  $M = (K, \Sigma, \Sigma, H, q_0, F)$ , where  $H = \{(q, a, a, \delta(q, a)) | q \in K, a \in \Sigma\}$ .  $\square$

**Notation.** For each family  $\mathcal{L}$  of languages,

$$\begin{aligned}\mathcal{M}(\mathcal{L}) &= \{M(L) | L \in \mathcal{L}, M \text{ is an } \epsilon\text{-free A-transducer}\} \\ \hat{\mathcal{M}}(\mathcal{L}) &= \{M(L) | L \in \mathcal{L}, M \text{ is an arbitrary A-transducer}\}\end{aligned}$$

**Theorem 7.** For each family  $\mathcal{L}$  of languages,  $\mathcal{M}(\mathcal{L})$  ( $\hat{\mathcal{M}}(\mathcal{L})$ ) is the smallest (full) trio containing  $\mathcal{L}$ .

*Proof.* Once again, we use the alternative definition of the image of  $L$ ,  $M(L) = \{pr_2(pr_1^{-1}(w) \cap \Pi_M | w \in L)\}$ . Considering previous lemmas,  $\mathcal{M}(\mathcal{L})$  ( $\hat{\mathcal{M}}(\mathcal{L})$ ) is clearly a (full) trio (note, that if  $M$  is  $\epsilon$ -free,  $pr_2$  is also  $\epsilon$ -free).

Now, let  $\mathcal{L}'$  be a (full) trio containing  $\mathcal{L}$ . Obviously,  $\mathcal{L}'$  also contains  $\mathcal{M}(\mathcal{L})$  ( $\hat{\mathcal{M}}(\mathcal{L})$ ), since it has to be closed under ( $\epsilon$ -free) homomorphism, inverse homomorphism and intersection with a regular language. Therefore,  $\mathcal{M}(\mathcal{L})$  ( $\hat{\mathcal{M}}(\mathcal{L})$ ) is the smallest (full) trio containing  $\mathcal{L}$ .  $\square$

**Notation.** If  $\mathcal{L}$  is a single language, we write  $\mathcal{M}(L)$  instead of  $\mathcal{M}(\{L\})$ .

In fact, it was shown in [3], that  $\mathcal{M}(L)$  ( $\hat{\mathcal{M}}(L)$ ) is the smallest (full) semi-AFL containing language  $L$ .

## 2.2 State Complexity of Finite State Devices

The topic of desriptional complexity of finite state devices has been widely researched in connection with finite state automata. Some results have been introduced also for sequential transducers, but the complexity of A-transducers has been on the periphery of interest. For this reason, this section contains the achievements for simpler devices, which can be later useful when dealing with our main model, an A-transducer.

### 2.2.1 Finite State Automata

We would like to occupy ourselves with the question, what is the lower bound of state count needed to accept a language  $L$ . Or, otherwise stated, what is the relation between the properties of a regular language and the minimal state size of its finite automaton?

For deterministic finite automaton, the answer was given by Nerode in [5]. We present his result in a slightly modified form, which suits better for our purposes.

**Theorem 8.** Let  $L$  be a regular language over alphabet  $\Sigma$ . Let  $R_L$  be a relation on strings from  $\Sigma^*$  defined as follows:

$$xR_Ly \Leftrightarrow \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L.$$

Let  $k$  be a number of equivalence classes of  $R_L$ . If  $A$  is a deterministic finite automaton accepting  $L$ , then  $A$  has at least  $k$  states.

*Proof.* Let  $A = (K, \Sigma, \delta, q_0, F)$ . We can construct a relation  $R'$  based on automaton  $A$  as follows:

$$\text{for } x, y \in \Sigma^*, xR'y \Leftrightarrow \delta(q_0, x) = \delta(q_0, y).$$

Since  $A$  is deterministic, it is easy to see, that  $\forall z \in \Sigma^* : xR'y \Leftrightarrow xzR'yz$ . Moreover, the number of its equivalence classes is exactly the number of reachable states of  $A$ . Now, we will show, that the relation  $R'$  is a refinement of  $R_L$  (i. e. each equivalence class of  $R'$  is contained in a equivalence class of  $R_L$ ).

Assume  $xR'y$ . As stated before, also  $xzR'yz$ . That means, that  $\delta(q_0, xz) \in F \Leftrightarrow \delta(q_0, yz)$  and therefore  $xR_Ly$ . It follows, that whole equivalence class of  $R'$  containing  $x$  (later noted as  $[x]$ ) is a subclass of an equivalence class of  $R_L$  and hence  $R'$  has not less equivalence classes than  $R_L$ .  $\square$

Important observation is, that this lower bound is tight, i. e. there really exists a DFA  $A'$  accepting  $L$  with  $k$  states. We can construct it from relation  $R_L$  as  $A' = (K', \Sigma, \delta', q'_0, F')$ :

- $K'$  is the set of equivalence classes of  $R_L$ ,
- $\delta([x], a) = [xa]$ ,
- $q'_0 = [\epsilon]$ ,
- $F' = \{[z] | z \in L\}$ .

It is easy to see, that this  $L(A') = L$  and  $A'$  has exactly  $k$  states.

Similar result was achieved for non-deterministic automata. However, its lower bound is not always tight (i. e. sometimes the minimal number of states of NFA is even bigger) and moreover, it is not practically computable, since the problem, if there is a NFA with  $\leq k$  states equivalent to a given DFA is *PSPACE*-complete ([7]). Following theorem was introduced in [6].

**Theorem 9.** Let  $L \subseteq \Sigma^*$  be a regular language and suppose there exists a set of pairs  $P = \{(x_i, w_i) : 1 \leq i \leq n\}$  such that

1.  $x_i w_i \in L$  for  $1 \leq i \leq n$ ,
2.  $x_j w_i \notin L$  for  $1 \leq i, j \leq n$  and  $i \neq j$ .

Then any non-deterministic finite automaton accepting  $L$  has at least  $n$  states.

*Proof.* Let  $A = (K, \Sigma, \delta, q_0, F)$  be a NFA accepting  $L$ . Now, let  $S = \{q | \exists i, 1 \leq i \leq n : \delta(q_0, x_i) \ni q\}$ . For every  $i$ , there must be a state  $p_i \in S$ , such that  $p_i \in \delta(q_0, x_i)$  and  $\delta(p_i, w_i) \cap F \neq \emptyset$  (since  $x_i w_i \in L$ ).

Now it is sufficient to show, that all states  $p_i$  are distinct. Indeed, if  $p_i = p_j$ , then  $\delta(p_i, w_i) = \delta(p_j, w_i)$ . Especially,  $\delta(p_i, w_i) \cap F \neq \emptyset \Leftrightarrow \delta(p_j, w_i) \cap F \neq \emptyset$ . It follows, that  $x_j w_i \in L$ , which is contradiction with definition of  $P$ .

Since  $|S| \geq n$ ,  $A$  has at least  $n$  states.  $\square$

## 2.2.2 Sequential Transducers

The natural question arises, how can be these results extended if we add an output function, in other words, what is the lower bound for number of states of an (sequential, a-) transducer, which transforms a language  $L_1$  to a language  $L_2$ ? Unfortunately, we do not have a answer in such a general form yet. However, in the case of sequential transducers, in [8] was given an answer to a simplified question: what is the minimal number of states of a sequential transducers representing a sequential function?

**Notation.** If  $f$  is a sequential function (see Chapter 1), we denote

- $Dom(f)$  is a set of strings  $w$ , for which  $f(w)$  is defined,
- $D(f) = \{u \in \Sigma^* | \exists w \in \Sigma^* : uw \in Dom(f)\}$ .

**Notation.** By  $\setminus$  we denote the operation of a left quotient.

**Definition.** For a sequential function  $f$  we define a relation  $R_f$  on  $D(f)$  as follows:

$$\begin{aligned} \forall (u, v) \in D(f) \times D(f) : uR_f v &\iff \\ \exists (x, y) \in \Sigma_2^* \times \Sigma_2^* : \forall w \in \Sigma_1^*, uw \in \text{Dom}(f) &\iff vw \in \text{Dom}(f) \wedge \\ \wedge uw \in \text{Dom}(f) \Rightarrow x \setminus f(uw) &= y \setminus f(vw). \end{aligned}$$

**Theorem 10.** A number of states of a sequential transducer  $M$  representing a sequential function  $f$  is greater or equal to a number of equivalence classes of  $R_f$ .

*Proof.* Let  $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0, F)$ . Choosing  $x = \sigma(q_0, u)$  and  $y = \sigma(q_0, v)$ , it is easy to see, that

$$\forall (u, v) \in D(f) \times D(f), \delta(q_0, u) = \delta(q_0, v) \Rightarrow uR_f v.$$

Moreover,  $\delta(q_0, u) = \delta(q_0, v)$  also defines an equivalence relation on  $D(f)$ . As we can see, this relation is just a special case of  $R_f$ , which means, that its number of equivalence classes (ergo the number of states of  $M$ ) is greater or equal to the number of equivalence classes of  $R_f$ .  $\square$

It was also shown, that this lower bound is tight, i. e. there is a sequential transducer realizing  $f$  with  $|K|$  equal to number of equivalence classes of  $R_f$ . However, we do not induct the proof of this claim, since it is quite technical and is not of vast relevance itself.

As mentioned before, we do not know, how can be this result applied to a couple of languages  $L_1$  and  $L_2$ , if we do not have the exact sequential function transforming the former to the latter.

## 2.3 Decompositions of finite automata

ToDo: results from Gazi about SB, ASB and advisors (necessary and sufficient conditions, maybe closure properties etc.).

# Chapter 3

## Complexity of A-transducers

This section is concerned with the complexity of A-transducers. Since the majority of the results published to this date involve sequential transducers and sequential functions, we try to investigate two new concepts in this area - nondeterminism and the fact, that we deal with pairs of languages, without exactly defined transduction.

However, at this time we do not have any universal way of proving the minimality of an A-transducer (in regard to number of states). For this reason, we would like to look at some special classes of transformations and languages and present the results concerning these.

### 3.1 "Modular-counting" languages

Under "modular counting" languages we mean languages in the form

$$L_k = \{a^k | k \equiv 0(mod k)\}.$$

As we know, for a regular language  $R$  (and therefore also, when  $R$  is modular counting language), there always exists an A-transducer with  $\mathcal{C}_{state}(R)$  states, which generates  $R$  "from scratch", regardless of the input - we take the finite automaton for  $R$  and alter its transition function from reading to generating symbols.

Formally, for an automaton  $A = (Q, \Sigma, \delta, q_0, F)$  we can construct an A-transducer  $T = (Q, \Sigma, \Sigma, H, q_0, F)$ , where  $H = \{(p, \epsilon, a, q) | \delta(p, a) = q\} \cup \{(q_0, a, \epsilon, q_0) | a \in \Sigma\}$ . It is easy to see, that for any nonempty language  $L$ ,  $T(L) = R$ .

However, the purpose of A-transducers clearly is to make the generating of languages easier using the input. Therefore, we would now like to present our results concerning the minimum complexity of an A-transducer for a pair of languages.

**Notation.** By  $\gcd(a, b)$  we denote a greatest common divisor of integers  $a, b$ .

**Lemma 11.** For a pair of languages  $L_k, L_l$ , the minimal state complexity of an A-transducer  $T$ , such that  $T(L_k) = L_l$ , is

1.  $l$ , if  $k$  and  $l$  are coprime integers,
2.  $\frac{l}{\gcd(k, l)}$ , if  $k \leq l$ ,
3.  $\frac{k}{\gcd(k, l)}$ , if  $k > l \wedge k < l^2$ ,
4.  $l$ , if  $k \geq l^2$ .

*Proof.* For sake of clarity, we prove the four parts of the Lemma separately. However, as stated before,  $l$  states are always sufficient, so we have a natural upper bound for parts 1. and 4.

1. Let  $T = (Q, \Sigma, \Sigma, H, q_0, F)$  be an A-transducer, such that  $T(L_k) = L_l$ . Let  $T$  have  $l - 1$  states. Now, let us look at an accepting computation (in this case the sequence of states) of  $T$  on some sufficiently long word  $x \in L_k$  ( $|x| \geq l$ ), by which  $T$  generates a word  $y \in L_l$ . Clearly, there has to be a cycle, i. e. the computation has a form  $q_0, q_1, \dots, q_i, \dots, q_j, \dots, q_F$ , where  $q_F \in F$  and  $q_i \equiv q_j$ , while  $j < i + l$  (we assume that this is the shortest cycle in the computation, during that  $T$  generates a non-empty output). In this cycle,  $T$  reads a subword  $a^r$  and generates output  $a^s$ ,  $1 \leq s \leq l - 1$ .

The two occurrences of the state  $q_i \equiv q_j$  in the sequence are indistinguishable from the point of view of  $T$ . Now, let us take two longer inputs  $x' \equiv x.a^{k*r}$  and  $x'' \equiv x.a^{2k*r}$ . On these two inputs,  $T$  generates outputs  $y' \equiv y.a^{k*s}$  and  $y'' \equiv y.a^{2k*s}$ , respectively. Since  $k$  and  $l$  are coprime integers and  $s < l$ ,  $k * s$  is not divisible by  $l$ , therefore one of these outputs does not belong to  $L_l$ , while both  $x', x'' \in L_k$ . We have generated an incorrect output, thus  $T$  cannot have less than  $l$  states.

2. First we will show, that  $\frac{l}{\gcd(k, l)}$  states suffice. We can construct an A-transducer  $T = (Q, \{a\}, \{a\}, H, q_0, F)$ , where

- $Q = \{q_0, q_1, \dots, q_{\frac{l}{\gcd(k, l)} - 1}\}$
- $F = q_{\frac{l}{\gcd(k, l)} - 1}$
- $H = \{(q_i, a, a, q_{i+1}) | 0 \leq i < \frac{k}{\gcd(k, l)} - 1\} \cup \{(q_i, \epsilon, a, q_{i+1}) | \frac{k}{\gcd(k, l)} - 1 \leq i < \frac{l}{\gcd(k, l)} - 2\} \cup \{(q_{\frac{l}{\gcd(k, l)} - 1}, \epsilon, a, q_0)\}$ .

It is easy to see, that the number of iterations of this cycle on a correct input (from  $L_k$ ) is divisible by  $\gcd(k, l)$ . Each iteration creates  $\frac{l}{\gcd(k, l)}$  symbols  $a$  on the output, therefore  $T(L_k) = L_l$ .

Now we need to prove, that this number really forms a lower bound for state count: suppose, that there is an A-transducer  $T' = (Q, \Sigma, \Sigma, H, q_0, F)$  with  $\frac{l}{\gcd(k, l)} - 1$  states. Similarly to the proof of part 1., we look for a cycle, in this case of the length  $\frac{l}{\gcd(k, l)} - 1$  states. With very similar series of arguments, we can construct two inputs  $x' \equiv x.a^{k*s}$  and  $x'' \equiv x.a^{2k*s}$ , which produce outputs  $y' \equiv y.a^{k*s}$  and  $y'' \equiv y.a^{2k*s}$ , respectively. If both of these numbers were divisible by  $l$ , then also  $k * s$  would be divisible by  $l$ . However, this is not possible, since  $s < \frac{l}{\gcd(k, l)}$ .

3. Just like in part 2., we show, that if  $k > l \wedge k < l^2$ , then  $\frac{k}{\gcd(k, l)}$  states is enough. The corresponding A-transducer will look as follows:  $T = (Q, \{a\}, \{a\}, H, q_0, F)$ , where

- $Q = \{q_0, q_1, \dots, q_{\frac{k}{\gcd(k, l)} - 1}\}$
- $F = q_{\frac{l}{\gcd(k, l)} - 1}$
- $H = \{(q_i, a, a, q_{i+1}) | 0 \leq i < \frac{l}{\gcd(k, l)} - 1\} \cup \{(q_i, a, \epsilon, q_{i+1}) | \frac{l}{\gcd(k, l)} - 1 \leq i < \frac{k}{\gcd(k, l)} - 2\} \cup \{(q_{\frac{k}{\gcd(k, l)} - 1}, \epsilon, a, q_0)\}$ .

For similar reason as in part 2., it is clear, that  $T(L_k) = L_l$ .

**ToDo:** tu ide o to, aby to stihalo citat vstup, inak by sme nevedeli vygenerovat VSETKY korektne vystupy - treba to dotiahnut a spisat - v zmysle "v kazdej iteracii cyklu musi vygenerovat, najblizsi korektny vstup je x', ten by ale vyplul privela vystupu (a to plati pre akykolvek cyklus, takže nemozme generovat nejako inak)".

**ToDo:** tieto argumenty treba obhajit nejakou teoriou cisel (ak su vobec korektne) Napríklad tu ide aj o to, aby to stihalo citat vstup, inak by sme nevedeli vygenerovat VSETKY korektne vystupy - treba to dotiahnut a spisat.

4. This part follow directly from previous claim - if  $k \geq l^2$ , then  $l \leq \frac{k}{\gcd(k, l)}$  and therefore it is easier to generate  $L_l$  "from scratch", using an A-transducer based on its final automaton (see above).

□

**Theorem 12.** We can summarize previous lemmas in following claim:

$$\mathcal{C}_{state}(L_k, L_l) = \min(l, \frac{\max(k, l)}{\gcd(k, l)})$$



## 3.2 Common transformations

ToDo: dovodit, preco je to uzitocne (vyuzijeme pri advisors, hopefully)

ToDo: nejak rozumne formulovat vysledok toho tvaru, ze na zmenu abecedy z k-arnej na l-arnu treba nejaký logaritmus stavov

ToDo: vysledky pri pocitani XOR s nejakým specific klucom

# Chapter 4

## Foreign advisors

### 4.1 Description of the framework

**Definition.** Let  $T$  be an a-transducer and  $L$  a language. Then  $T^{-1}(L)$  is the set of all words, such that its images belong to  $L$ . Formally

$$T^{-1}(L) = \{w | T(w) \subseteq L\}.$$

Description of the framework: we have a decider  $D$ , which should construct a (deterministic?) finite automaton for the language  $L_{dec}$ . Moreover, we have an advisor (oracle)  $O$ . Now,  $O$  sends  $D$  a dual information: an a-transducer  $T$  and a regular language  $L_{adv}$ . This information forms a promise, that if  $D$  transforms a correct input (that is, input from  $L_{dec}$ ) using  $T$ , it will belong to  $L_{adv}$ . Now,  $D$  has two possibilities:

1. it trashes the information from  $O$  and constructs an automaton  $A$  for  $L_{dec}$  "from scratch"
2. it creates a simpler automaton  $A'$ , such that  $L[T^{-1}(L_{adv})](A') = L_{dec}$ .

Moreover,  $L_{adv}$  has to be verifiable by a finite automaton  $V$ .

**Definition.** A language  $L_{adv}$  with an a-transducer  $T$  is an *effective advice with regard to  $L_{dec}$* , if there exists an automaton  $A'$ , such that  $L_{dec} = L[T^{-1}(L_{adv})](A')$  and  $\mathcal{C}_{state}(A') + \mathcal{C}_{state}(T) + \mathcal{C}_{state}(L_{adv}) \leq \mathcal{C}_{state}(L)$ .

**Example 1.** Let  $L_{dec} = \{a^{12k} | k \geq 0\}$ ,  $T$  be an one state a-transducer computing the identity and  $L_{adv} = \{a^{2k} | k \geq 0\}$ .  $D$  can now construct a simpler finite automaton  $A'$  for the language  $L_{simple} = \{a^{6k} | k \geq 0\}$ . Clearly,  $\mathcal{C}_{state}(A') + \mathcal{C}_{state}(T) + \mathcal{C}_{state}(L_{adv}) = 6 + 1 + 2 \leq 12 = \mathcal{C}_{state}(L_{dec})$ , which means, that  $L_{adv}$  with  $T$  is an effective advice with regard to  $L_{dec}$ .

**Example 2.** Let  $L_{dec} = \{a^{12k} | k \geq 0\}$ . Let  $T = (\{q_0, q_1\}, \{a\}, \{a\}, H, q_0, \{q_0\})$ , where  $H = \{(q_0, a, a, q_1), (q_1, a, \epsilon, q_0)\}$  and  $L_{adv} = \{a^{2k} | k \geq 0\}$ . It is easy to see, that  $T^{-1}(L_{adv}) = \{a^{4k} | k \geq 0\}$ .  $D$  can now construct a simpler finite automaton  $A'$  for the language  $L_{simple} = \{a^{3k} | k \geq 0\}$ . Clearly,  $\mathcal{C}_{state}(A') + \mathcal{C}_{state}(T) + \mathcal{C}_{state}(L_{adv}) = 3 + 2 + 2 \leq 12 = \mathcal{C}_{state}(L_{dec})$ , which means, that  $L_{adv}$  with  $T$  is an effective advice with regard to  $L_{dec}$ .

Two interesting questions arise. The first is, for given language  $L$  and a-transducer  $T$ , how to get the language  $T^{-1}(L)$ ? The answer was quite easy to find in previous two examples (and, in fact, for all languages in form  $\{(a^k)^+\}$  and a-transducers, which just manipulates the number of symbols  $a$ ).

Another question is in some sense the inverse perspective of this problem. We have a fixed language  $L$  and we want to transform it to a language  $L_{adv}$ . Since we want to minimize the complexity of the advice, our question is, what is the minimal state complexity of an a-transducer  $T$ , such that  $T(L) = L_{adv}$ . We address this question in Section [number of complexity section].

## 4.2 T-decomposable and T-undecomposable languages

Now we would like to compare our setting to the setting presented by [Gazi, diploma thesis] (see Section [number of definitions Section]).

**Definition 1.** The language  $L$  is called *T-decomposable*, if there is a language  $L_{adv}$ , which is an effective advice for  $L$ . Otherwise, we call  $L$  *T-undecomposable*.

**Theorem 13.** If there exists a non-trivial *ASB*-decomposition for language  $L$  [would like to write it in different way (using advisors, not decomposition), but did not find suitable definitions in Gazi's diploma thesis], then  $L$  is T-decomposable.

*Proof.* Easy to see, using an A-transducer computing the identity.  $\square$

However, the next theorem shows, that the reverse implication does not hold.

**Theorem 14.** There are infinitely many T-decomposable languages, that are not *ASB*-decomposable.

*Proof.* Such languages are for example  $L_x = \{u\$xv \mid u, v \in \{a, b\}^*\}$  for a fixed string  $x \in \{a, b\}^*$ ,  $|x| \geq 14$  and even.

We prove this claim in two steps. First, we need to show, that  $L_x$  is T-decomposable. It is easy to see, that a DFA accepting  $L_x$  needs at least  $|x| + 1$  states, therefore  $\mathcal{C}_{state}(L_x) = |x| + 1$ .

However, we can use an advice to simplify the accepting automaton as follows: our A-transducer  $T$  will read the input word in the initial state with no output, until it finds the special marker  $\$$ . Then, using another three states, it encodes pairs of symbols (i. e. sequences  $aa, ab, ba, bb$ ) into new letters  $c, d, e, f$ , respectively. If there is just one symbol in the end,  $T$  will read it and traverse into accepting state  $q_F$  with no further transitions, otherwise it will make an  $\epsilon$ -transition into  $q_F$ . Note, that  $T$  uses just five states.

Now, the advise language  $L_{x,adv} = \{x'v \mid x' \text{ is the aforementioned encoded form of } x \text{ into symbols } c, d, e, f\}$ . Clearly,  $|x'| = \frac{|x|}{2}$  and  $\mathcal{C}_{state}(L_{x,adv}) = \frac{|x|}{2} + 1$ .

The decider  $D$  needs construct just an automaton for  $\{a, b, \$\}^*$ , since the advice gives full information about  $L_x$ . Alltogether, we used  $5 + \frac{|x|}{2} + 1 + 1$  states, therefore for  $|x| \geq 14$  is  $L_{x,adv}$  with  $T$  an effective advice with regard to  $L_x$ .

Our next goal is to show, that  $L_x$  is not ASB-decomposable. [ToDo: directly or using necessary conditions from Gazi]□

**Corollary 14.1.** There are infinitely many T-decomposable languages.

**Theorem 15.** There are infinitely many T-undecomposable languages.

To prove this theorem, we show the following lemma.

**Lemma 16.** Each of the languages  $L_n = \{w \in \{a, b\}^* \mid \#_a(w) \bmod p \equiv 0, p \text{ is the } n\text{-th prime}\}$  is T-undecomposable.

*Proof.*

TODO: proof - probably using some alternation of pumping lemma, try to show, that the number of equivalence classes can't be reduced, since  $p$  is prime

□

As we have seen, the class of regular languages can be divided into three parts - both  $AS\ B$ -decomposable and  $T$ -decomposable languages;  $AS\ B$ -undecomposable, but  $T$ -decomposable; and  $T$ -undecomposable languages [I will try to write this in a better way]. In the next part of our thesis, we would like to investigate some properties of these classes.

### 4.3 Closure properties

ToDo: closure properties - hotovy je komplement a homomorfizmus (aj nevyamazavajuci)

# Conclusion

ToDo: Conclusion

# Bibliography

- [1] J.E. Hopcroft and J.D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley Pub. Co., 1969.
- [2] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. Elsevier Science Inc., New York, NY, USA, 1975.
- [3] S. Ginsburg and S. Greibach. *Principal AFL*. J. Comput. Syst. Sci. 4, 4 (August 1970), 308-338.
- [4] B. Rovan. *Proving Containment of Bounded AFL*. J. Comput. Syst. Sci. 11, 1 (August 1975), 1-55.
- [5] A. Nerode. *Linear Automaton Transformations*. Proceedings of the American Mathematical Society, 9, 4 (Aug., 1958), 541-544.
- [6] I. Glaister, J. Shallit. *A Lower Bound Technique for the Size of Nondeterministic Finite Automata*. Inf. Process. Lett. 59, 2 (July 1996), 75-77.
- [7] T. Jiang and B. Ravikumar. 1993. *Minimal NFA problems are hard*. SIAM J. Comput. 22, 6 (December 1993), 1117-1141.
- [8] M. Mohri. *Minimization Algorithms for Sequential Transducers*. Theor. Comput. Sci. 234, 1-2 (March 2000), 177-201.