Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Complexity of Language Transformations

Diploma Thesis

February 4, 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 Rovan, 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 ZÁVEREČNEJ PRÁCE

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 Rovan, 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, descriptional complexity, a-transducer

Contents

Introduction

A number of models for realization of language transformations were introduced in the past fifty years. Most of them can be described as an extension of a finite automaton, which has been augmented by an output tape. Such models are, e. g., Moore and Mealy machines, sequential transducers, general sequential machines, a-transducers, etc.

In our thesis, we would like to investigate some complexity properties of these transformation models, with emphasis on a-transducers.

We assume, that the reader is familiar with the 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, we present some basic notation and terminology used in our thesis.

1.1 Basic concepts and notation

Notation. In our thesis we use the following notation: ε denotes an empty string, $|w|$ the length of a word w ($|\varepsilon| = 0$), $|A|$ the number of elements of a finite set (or a finite language) A, $\#_a(w)$ the number of occurences of the symbol a in the word w, 2^A the set of all subsets of A.

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

$$\forall u, v \in \Sigma_1^* : h(uv) = h(u)h(v)$$

Notation. If $\forall w \neq \varepsilon : h(w) \neq \varepsilon$, we call h an *ε-free homomorphism*. Usually, we denote an ε-free homomorphism by h_ε.

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

$$\forall u \in \Sigma_1^* : h^{-1}(u) = \{v \in \Sigma_2^* | h(v) = u\}$$

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

1. Σ is an infinite set of symbols
2. every $L \in \mathcal{L}$ is a language over some finite set $\Sigma_1 \subset \Sigma$
3. $L \neq \emptyset$ for some $L \in \mathcal{L}$

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

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 +.

1.2 Transformation models

We shall now define some of the models mentioned in the Introduction. Although the central point of our interest is an 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 a-transducers.

Since all transformation models used in our thesis are, in fact, special cases of an a-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,
- Σ_1 and Σ_2 are the input and output alphabets, 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 *ε-free* a-transducer.

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

Definition. A *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. An *image* of a language L by an a-transducer M is a set

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

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 the *i-th projection* .

Definition. A *computation* of an a-transducer M is a word $h_0 h_1 ... h_m \in H^*$, 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 Π_M. Note, that Π_M is regular ([2]).

Definition. Alternatively, we can define an image of L by an a-transducer M by

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

Definition. An *A-transduction* is a function $\Phi : \Sigma_1^* \to 2^{\Sigma_2^*}$ 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 using the original notation (see e. g. [8]).

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,
- δ is a transition function, which maps $K \times \Sigma_1 \to K$,
- σ is an output function, which maps $K \times \Sigma_1 \to \Sigma_2^*$.

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

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

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

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

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

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

We conclude this section by a definition of one more model, which can be viewed as a special case 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 case.

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 $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ (a finite automaton $A = (K, \Sigma, \delta, q_0, F)$), denoted by $\mathscr{C}_{state}(T)$ ($\mathscr{C}_{state}(A)$), is the number of its states. Formally

$$\mathscr{C}_{state}(T) = |K|.$$

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

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

If L is not regular, we define $\mathscr{C}_{state}(L) = \infty$.

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

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

Note, that it is possible, that $\mathscr{C}_{state}(L_1, L_2) \neq \mathscr{C}_{state}(L_2, L_1)$ and it may happen, that $\mathscr{C}_{state}(L_1, L_2) = \infty$ (if there is no a-transducer M, such that $M(L_1) = L_2$).

Now we shall define the acceptance of a language with an advisor and some other concepts presented in [9].

Definition. For a language L_1 and an automaton $A = (K, \Sigma, \delta, q_0, F)$, a *language accepted by A with the advisor* L_1 is the language

$$L[L_1](A) = \{w \in L_1 | (q_0, w) \vdash_A^* (q, \varepsilon), q \in F\}.$$

Another way for looking at this fact is, that $L[L_1](A) = L(A) \cap L_1$.

Definition. Let $A' = (K', \Sigma, \delta', q_0', F')$ and $A = (K, \Sigma, \delta, q_0, F)$ be deterministic finite automata. We say, that A' realizes the the state behavior of A, if there is an injective mapping $\alpha : K \to K'$, such that:

- $\forall a \in \Sigma, \forall q \in K : \delta(\alpha(q), a) = \alpha(\delta(q, a))$,
- $\alpha(q_0) = q_0'$.

Moreover, if $\forall q \in K : \alpha(q) \in F' \Leftrightarrow q \in F$, we say, that A' *realizes the state and acceptation behavior* of A.

Definition. Let $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1), A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$ be deterministic finite automata. Their *parallel connection* is an automaton $A_1||A_2 = (K_1 \times K_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$, where $\forall (p_1, p_2) \in K_1 \times K_2, a \in \Sigma : \delta((q_1, q_2), a) = (\delta_1(p_1, a), \delta_2(p_2, a))$.

Definition. We say, that a pair (A_1, A_2) is a *state behavior (SB-) decomposition* of a deterministic finite automaton A, if $A_1||A_2$ realizes the state behavior of A. If $A_1||A_2$ realizes the state and acceptance behvaior of A, (A_1, A_2) forms a *state and acceptance (ASB-) behavior decomposition*.
If $\mathscr{C}_{state}(A_1) < \mathscr{C}_{state}(A)$ and $\mathscr{C}_{state}(A_2) < \mathscr{C}_{state}(A)$, the decomposition is called *nontrivial*.

Definition. A language L and its corresponding minimal deterministic finite automaton A are called (A)SB-undecomposable, if A has no nontrivial (A)SB-decomposition. The class of all regular (A)SB-undecomposable languages is denoted by $\mathcal{U}_{SB}$ ($\mathcal{U}_{ASB}$).

1.4 Group and number theory

ToDo: mozno definovat aspon zakladne pojmy z teorie grup, kedze ich neskor pouzivam a aj vysledok gcd(a,b).lcm(a,b)=a.b

Chapter 2

Current State of Research

In this chapter, we 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 $\mathcal{L}_{C\mathcal{F}}$ 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 of image L:

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

Since Π_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. □

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. The following theorem shows, that this is a normal form for a-transducer mappings.

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

Proof. Let $(q, u, v, p) \in H_1, u \equiv a_1a_2...a_m, v \equiv b_1b_2...b_n$. Let $m \geq n$ (for $m < n$ the proof is very similar). M_2 will have states $q, q_{a_1}, q_{a_2}, ..., q_{a_{n-1}}, q_{a_n} \equiv p$ and transitions in form $(q_{a_i}, a_{i+1}, b_{i+1}, q_{a_{i+1}})$ for $1 \leq i < n$, resp. $(q_{a_j}, a_{j+1}, \varepsilon, q_{a_{j+1}})$ 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 $\forall L : M_2(L) = M_1(L)$. □

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.

Lemma 3. For every (ε-free) homomorphism $h : \Sigma_1^* \to \Sigma_2^*$ there is an (ε-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:

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

Lemma 4. 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 $H = \{(q, h(a), a, q) | a \in \Sigma_1\}$. □

Lemma 5. For every language L and regular language R, there exists an ε-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 $H = \{(q, a, a, \delta(q, a)) | q \in K, a \in \Sigma\}$. □

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

$$\mathcal{M}(\mathcal{L}) = \{M(L) | L \in \mathcal{L}, M \text{ is an } \varepsilon\text{-free a-transducer}\}$$
$$\hat{\mathcal{M}}(\mathcal{L}) = \{M(L) | L \in \mathcal{L}, M \text{ is an arbitrary a-transducer}\}$$

Theorem 6. 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 ε-free, pr_2 is also ε-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 (ε-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 $\mathcal{L}$. □

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 descriptional complexity of finite state devices has been widely researched in connection with finite state automata. Some results have been introduced also for sequential transducers. This section contains the achievements for these 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, how to find $\mathscr{C}_{state}(L)$ for a regular language L. Or, otherwise stated, what is the relation between the properties of a regular language and the minimal state count of its finite automaton?

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

Theorem 7. Let L be a regular language over an alphabet Σ. Let R_L be a relation on strings from Σ^* 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. □

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,
- $\delta([x], a) = [xa]$,
- $q'_0 = [\varepsilon]$,
- $F' = \{[z] | z \in L\}$.

It is easy to see, that $L(A') = L$ 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]). The following theorem was introduced in [6].

Theorem 8. 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 by a state $p_i \in S$, such that $p_i \in \delta(p_0, x_i)$ 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. □

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 the number of states of a (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 transducer 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:

$$\forall (u, v) \in D(f) \times D(f) : uR_f v \iff$$
$$\exists (x, y) \in \Sigma_2^* \times \Sigma_2^* : \forall w \in \Sigma_1^*, uw \in Dom(f) \Leftrightarrow vw \in Dom(f) \wedge$$
$$\wedge uw \in Dom(f) \Rightarrow x \setminus f(uw) = y \setminus f(vw).$$

Theorem 9. 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. □

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 present the proof of this claim, since it is quite technical and is not of vast importance for the purpose of our thesis.

As mentioned before, we do not know, how to apply this result to a pair 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

We now show the relation between state behavior decompositions and advisors and the necessary and sufficient condition on SB- and ASB-decomposability, as presented in [9].

Theorem 10. Let A be a deterministic finite automaton. If there exists a nontrivial ASB-decomoposition of A, then there exists a regular language L and an automaton A', such that $L(A) = L[L](A')$ and both $\mathscr{C}_{state}(L) < \mathscr{C}_{state}(A)$ and $\mathscr{C}_{state}(A') < \mathscr{C}_{state}(A)$.

Proof. We claim, that for any nontrivial decomposition of A on (A_1, A_2), $L = L(A_1)$ and $A' = A_2$. We show, that $L[L(A_1)](A_2) = L(A)$ in two containments:

- $L[L(A_1)](A_2) \subseteq L(A)$: Since $A_1 || A_2$ realizes the state and acceptance behavior of A, we know, that any word $w \in L(A)$ is accepted by $A_1 || A_2$. Moreover, the accepting computation of $A_1 || A_2$ can be decomposed into accepting computations of A_1 and A_2 (as we can see from the definition). Therefore $w \in L(A_1) = L$ and $w \in L(A_2)$, which implies $w \in L[L(A_1)](A_2)$.

- $L[L(A_1)](A_2) \supseteq L(A)$: The proof of this containment is similar, except we join the computations of A_1 and A_2 on a word $w \in L(A_1) \cap L(A_2)$ into the computation of $A_1 || A_2$, which gives us a corresponding accepting computation of A.

□

To present the aforementioned condition, we first need som additional definitions.

Definition. A partition π on a finite set S is a set $\{S_1, S_2, ..., S_k\}$, such that $\forall i : S_i \neq \emptyset$ and $\bigcup_{i=1}^{k} S_i = S$.

Notation. We denote the trivial partition of $S = \{s_0, s_1, ..., s_k\}$ into $\{s_0\}, \{s_1\}, ..., \{s_k\}$ by 0.

Definition. Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton. We say, that a partition π of K has a *substitution property* (S. P.), if

$$\forall p, q \in K : p \equiv_\pi \Rightarrow (\forall a \in \Sigma : \delta(p, a) \equiv \delta(q, a)).$$

Definition. For a given pair of partitions π_1 and π_2 of a set S, then $\pi_1.\pi_2$ is a parition of S, such that $a \equiv_{\pi_1.\pi_2} b \Leftrightarrow a \equiv_{\pi_1} b \wedge a \equiv_{\pi_2} b$.

Definition. Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton. We say, that the partitions $\pi_1 = \{S_1, S_2, ..., S_k\}$ and $\pi_2 = \{T_1, T_2, ..., T_l\}$ on K *separate the final states* of A, if there are two sets of indices $i_1, ..., i_m$ and $j_1, ..., j_n$, such that $(S_{i_1} \cup ... \cup S_{i_m}) \cap (T_{j_1} \cup ... \cup T_{j_n}) = F$.

Now we finally can proceed to the necessary and sufficient condition on (A)SB-decomposability.

Theorem 11. Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton. A is SB-decomposable if and only if there are two nontrivial partitions π_1, π_2 of K with substition property, such that $\pi_1.\pi_2 = 0$. Moreover, if π_1 and π_2 separate the final states of A, this decomposition is an ASB-decomposition.

The proof of this claim can be found in [9].

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.

As we know, for a regular language R, there always exists an a-transducer with $\mathscr{C}_{state}(R)$ 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 $A = (K, \Sigma, \delta, q_0, F)$ we can construct an a-transducer $M = (K, \Sigma, \Sigma, H, q_0, F)$, where $H = \{(p, \varepsilon, a, q) | \delta(p, a) = q\} \cup \{(q_0, a, \varepsilon, q_0) | a \in \Sigma\}$. We can look at the computation of A as a sequence of pairs (q, a), where in each step, A is in the state q and reads symbol a. The a-transducer M will work in the same way, except instead of reading symbol a, M reads in each step ε and writes a on the output. Then, in the final state, M consumes the whole input without generating any output. It is easy to see, that for any nonempty language L, $M(L) = R$.

3.1 "Modular-counting" languages

By "modular counting" languages we understand languages in the form

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

We would now like to present our results concerning the minimum complexity of an a-transducer for a pair of modular counting languages.

Notation. By $\gcd(k, l)$ we denote a greatest common divisor of integers k, l, by $\mathrm{lcm}(k, l)$ their lowest common multiple.

Lemma 12. For a pair of languages L_k, L_l, the minimal state complexity of an a-transudcer M, such that $M(L_k) = L_l$, is

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

Proof. For the 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 $M = (K, \{a\}, \{a\}, H, q_0, F)$ be an a-transducer, such that $M(L_k) = L_l$. Let M have $l - 1$ states. Now, let us look at an accepting computation (in this case the sequence of states) of M on some sufficiently long word $x \in L_k$ ($|x| \geq l$), on which M generates a word $y \in L_l$. Clearly, there has to be a cycle, i. e. the computation has a form $q_0, q_1, ..., q_i, ..., q_j, ..., q_F$, 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 M generates a non-empty output). In this cycle, M reads a subword a^r and generates output a^s for some $r, s; 1 \leq r, s \leq l - 1$.

   Now, let us take two longer inputs $x' \equiv x.a^{k.r}$ and $x'' \equiv x.a^{2k.r}$. On these two inputs, M 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 (the least common multiple of two coprimes is their product), therefore at least one of these outputs does not belong to L_l, while both $x', x'' \in L_k$. We have generated an incorrect output, thus M cannot have less than l states.

2. As claimed before, an a-transducer with l states does the job. Therefore, in further we assume, that $\frac{l}{\gcd(k,l)} < l$.

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

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

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 $M(L_k) = L_l$.

Now we need to prove, that this number really forms a lower bound for state count: suppose, that there is an a-transducer $M' = (K, \{a\}, \{a\}, H, q_0, F)$ with at most $\frac{l}{\gcd(k,l)} - 1$ 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 $x' \equiv x.a^{k.r}$ and $x'' \equiv x.a^{2k.r}$, 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 $s < \frac{l}{\gcd(k,l)}$ and as stated in Chapter 1 (probably), $\text{lcm}(k, l) = \frac{k.l}{\gcd(k,l)}$.

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: $M = (K, \{a\}, \{a\}, H, q_0, F)$, where

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

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

However, the second part of the proof is a little bit different. We will not show, that an a-trandsucer $M' = (K', \{a\}, \{a\}, H', q_0', F')$ with fewer states generates an incorrect output, but we claim, that it is not able to generate all correct outputs (i. e. all outputs from language L_l). What is the shortest nonempty word, that we can generate from L_k using M'?

We have assumed, that $k < l^2$, therefore we can also state, that $\frac{k}{\gcd(k,l)} < l$. Once again, we look for a cycle in the computation of M'. Since $|Q'| < l$, to produce an output of length l the computation must have a form $q'_0, q'_1, ..., q'_i, ..., q'_j, ..., q'_F$, where $q'_F \in F'$ and $q'_i \equiv q'_j$, while $j < i + \frac{k}{\gcd(k,l)}$. In each iteration of this cycle, M' has to output at least one symbol a.

We claim, that in each iterarion of the cycle (i. e. in any of all possible cycles in its computation), M' has to generate at least $\frac{l}{\gcd(k,l)}$ symbols a. Really, in the proof of the second part of our Lemma we have seen, that if the number s - the number of output symbols generated in one iteration of the cycle - is smaller than $\frac{l}{gcd(k,l)}$, $M'(L_k) \cap L_l^c \neq \emptyset$, which leads to a contradiction.

Moreover, since $|Q'| < \frac{k}{\gcd(k,l)}$, we also know, that in one iteration of each cycle, M' reads less than $\frac{k}{\gcd(k,l)}$ symbols. Now, the shortest nonempty word from L_k (if $M'(\varepsilon) \neq \emptyset$, it could be trivially proven, that $M'(L_k)$ contains also words not from L_l) is a^k. The total number of iterations of all cycles is hence more than $\frac{k}{\frac{k}{\gcd(k,l)}} = \gcd(k,l)$. However, as we have claimed, every cycle generates at least $\frac{l}{\gcd(k,l)}$ symbols. Then, the smallest output length $n > \gcd(k,l).\frac{l}{\gcd(k,l)} = l$, hence we have no way to generate word $a^l \in L_l$.

4. The correctness of the lower bound l is clear from the construction based on its final automaton (see above). The impossibility of existence of a smaller a-transducer follows directly from previous part of Lemma - if $k \geq l^2$, then $l \leq \frac{k}{\gcd(k,l)}$.

□

Theorem 13. We can summarize previous lemmas in following claim:

$$\mathscr{C}_{state}(L_k, L_l) = \min(l, \tfrac{\max(k,l)}{\gcd(k,l)}) \quad .$$

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 nejaky logaritmus stavov

ToDo: vysledky pri pocitani XOR s nejakym specific klucom

Chapter 4

Foreign advisors

4.1 Description of the framework

We now proceed to definitions associated to the central matter of our thesis, which is the framework for advisory information and its transformations.

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

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

Definition. Let L_{dec} be a regular language. A pair (L_{adv}, M), where L_{adv} is a regular language and M an a-transducer is called an *advice with regard to* L_{dec}, if there exists a deterministic finite automaton A', such that $L_{dec} = L[M^{-1}(L_{adv})](A')$. Moreover, (L_{adv}, M) is called *effective*, if $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{adv}) \leq \mathscr{C}_{state}(L_{dec})$.

Example 1. Let $L_{dec} = \{a^{12k} | k \geq 0\}$, M be 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, $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{adv}) = 6+1+2 \leq 12 = \mathscr{C}_{state}(L_{dec})$, which means, that L_{adv} with M is an effective advice with regard to L_{dec}.

Example 2. Let $L_{dec} = \{a^{12k} | k \geq 0\}$. Let $M = (\{q_0, q_1\}, \{a\}, \{a\}, H, q_0, \{q_0\})$, where $H = \{(q_0, a, a, q_1), (q_1, a, \varepsilon, q_0)\}$ and $L_{adv} = \{a^{2k} | k \geq 0\}$. It is easy to see, that $M^{-1}(L_{adv}) = \{a^{4k} | k \geq 0\}$. D can now construct a simpler finite automaton A' for the language $L_{simple} = \{a^{3k} | k \geq 0\}$. Clearly, $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{adv}) = 3 + 2 + 2 \leq 12 = \mathscr{C}_{state}(L_{dec})$, which means, that L_{adv} with M is an effective advice with regard to L_{dec}.

Two interesting questions arise. The first is, for given language L and a-transducer M, how to get the language $M^{-1}(L)$? 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). The answer in general is resolved by the following Lemma.

Lemma 14. For an a-transducer $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ and a language L, $M^{-1}(L) = L'$ if and only if $M(L') = L$ and $\forall w \in L'^c : M(w) = \emptyset \vee M(w) \subseteq L^c$. The mapping M^{-1} can be simulated by an a-transducer M' dual to M, such that $M'(L) = L'$ and $\forall w \in L^c : M'(w) = \emptyset \vee M'(w) \subseteq L'^c$. Moreover, $\mathscr{C}_{state}(M') = \mathscr{C}_{state}(M)$.

Proof. The first part is quite easy to see, since by definition, $M^{-1}(L)$ contains all words, whose images by a-transducer M belong to L. If for a word $v \in L'^c$ there is a word u, such that $u \in M(v)$ and $u \notin L^c$, then $u \in L$ and by definition, $v \in L'$, which leads to a contradiction. The proof of the reverse implication is very similar.

We prove the second part of our Lemma constructively. Let $M' = (K, \Sigma_2, \Sigma_1, H', q_0, F)$, where

$$H' = \{(p, x, y, q) | (p, y, x, q) \in H\}.$$

Clearly, $\mathscr{C}_{state}(M) = \mathscr{C}_{state}(M')$. It remains to show, that M' simulates M^{-1}, namely that $M'(L) = L'$ (since $L' = M^{-1}(L)$).

- $L' \subseteq M'(L)$: Take an arbitrary word $u \in L'$. By definition of M^{-1}, there is a word $v \in L$, such that $M(u) = v$. Now, look at the computation of M on u as a word $h \in \Pi_M$ (see Chapter 1). Since this computation is accepting and its output is v, we can rewrite h as a sequence of quadruples $(q_0, x_1, y_1, p_1)\,(p_1, x_2, y_2, p_2)...\,(p_{i-1}, x_i, y_i, p_i)...(p_{n-1}, x_n, y_n, q_F)$, such that $pr_1(h) = u$ and $pr_2(h) = v$. We now present the computation of M', that show, that $u \in M'(v)$. The computation is $h' \equiv (q_0, y_1, x_1, p_1)(p_1, y_2, x_2, p_2)...$ $(p_{i-1}, y_i, x_i, p_i)...(p_{n-1}, y_n, x_n, q_F)$. The plausibility of this computation follows from the construction of M'. We have shown, that $u \in M'(L)$ and therefore $L' \subseteq M'(L)$.

- $M'(L) \subseteq L'$: Once again, let us take a word $u \in M'(L)$. There is a word $v \in L$, such that $u \in M'(v)$. Again, we can look at the respective computation of M' on v as a word $h' \equiv (q_0, y_1, x_1, p_1)(p_1, y_2, x_2, p_2)...\,(p_{i-1}, y_i, x_i, p_i)...(p_{n-1}, y_n, x_n, q_F)$, where $pr_1(h') = v$ and $pr_2(h') = u$. We construct the computation h of M in the same way as in previous part of the proof. The computation h shows, that $v \in M(u)$ and therefore $M(u) \subseteq L$ (whole $M(u)$, since all words v, such that $u \in M'(v)$ have to belong to L according to the first part of Lemma). From the definition of M^{-1} it follows, that $u \in M^{-1}(L) = L'$.

- $\forall w \in L^c : M'(w) = \emptyset \vee M'(w) \subseteq L'^c$: Assume there is a word $w \in L^c$, such that $M'(w) \ni u \wedge u \in L'$. From previous part of Lemma it follows, that $u \in M'(L)$. However, then $w \in M(u) \subseteq L$, which leads to a contradiction.

□

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 M, such that $M^{-1}(L_{adv}) = L$? Previous Lemma has shown us, that this is slightly more difficult thatn the question addressed in Chapter 3, i. e. how to compute $\mathscr{C}_{state}(L, L_{adv})$. However, $\mathscr{C}_{state}(L, L_{adv})$ clearly is a lower bound for $\mathscr{C}_{state}(M)$ satisfying aforementioned conditions.

4.2 T-decomposable and T-undecomposable languages

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*.

Now we to compare our setting to the setting presented by [9] (see Section 2.3). To make the comparison more meaningful, we have to strengthen the condition presented by Gazi in a following way:

Definition 2. A language L is called *A-decomposable*, if there exists an advisor L_1 and an automaton A, such that $\mathscr{C}_{state}(L_1) + \mathscr{C}_{state}(A_1) < \mathscr{C}_{state}(L)$ and $L(A, L_1) = L$.

Theorem 15. Every A-decomposable language is T-decomposable.

Proof. Easy to see, using an a-transducer computing the identity. □

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

Theorem 16. There are infinitely many T-decomposable languages, that are not a-decomposable.

Proof. Such languages are for example $L_n = \{a^n\}$ for $n \geq 10$ and even.

We prove this claim in two steps. First, we need to show, that L_n is T-decomposable. It is easy to see, that a DFA accepting L_n needs at least $n + 2$ states, therefore $\mathscr{C}_{state}(L_n) = n + 2$.

However, we can use an advice to simplify the accepting automaton as follows: our a-transducer M will encode pairs of letter a into new letters b using two states, where the first state is accepting.

Now, the advise language is $L_{n,adv} = \{b^{\frac{n}{2}}\}$. Clearly, $\mathscr{C}_{state}(L_{n,adv}) \leq \frac{n}{2} + 2$.

We need to construct just an automaton for $\{a\}^*$, since the advice gives the full information about L_n. Altogether, we used $2 + \frac{n}{2} + 2 + 1$ states, therefore for $n \geq 10$ is $L_{n,adv}$ with M an effective advice with regard to L_n.

Our next goal is to show, that L_n is not A-decomposable. As we have said before, a minimal DFA A for L_n has $n + 1$ states and its states correspond to the equivalence classes of the relation defined by Myhill-Nerode theorem (see Section 2.2.1). These equivalence classes are:

1. $[c_0] = \{\varepsilon\}$,
2. $[c_i] = \{a^i\}$ for $1 \leq i \leq n$,
3. $[c_{n+1}] = \{a^k | k > n\}$.

We proceed by contradiction, therefore we assume, that we can find an automaton A' and a language L_{adv} (with an automaton A_{adv}), such that $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(L_{adv}) < \mathscr{C}_{state}(L_n) = n + 1$. We will show, that both A' and A_{adv} need at least n states, otherwise they would accept an input from $[c_{n+1}]$, which leads to a contradiction, since $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(L_{adv}) \geq n + n \geq n + 1 = \mathscr{C}_{state}(L_n)$.

Let us now look at the automaton A_{adv}. Since the inequality has to hold, A_{adv} can have at most n states. Also, A_{adv} has to accept the language L_n, that means, in our case, the word a^n. Clearly, by reading a^n, A_{adv} runs in the cycle. Without loss of generality, assume that in one iteration of the shortest cycle A_{adv} reads a^l. Therefore, it accepts also incorrect outputs in form of $a^{n+s.l}$, $s \geq 1$.

The same argument can be used for A'. Assume, that it accepts also words $a^{n+s.k}$, $s \geq 1$. However, this means, that $a^{n+s.k.l} \in L_{adv}$ and also $a^{n+s.k.l} \in L(A')$ and our model accepts the word $a^{n+s.k.l}$. However, $a^{n+s.k.l} \notin L_n$. □

Corollary 16.1. There are infinitely many T-decomposable languages.

Theorem 17. There are infinitely many T-undecomposable languages.

Proof. Each of the languages $L_p = \{(a^p)^* | p \text{ is a prime number } \}$ is T-undecomposable.

It is easy to see, that $\mathscr{C}_{state}(L_p) = p$. Let us fix a particular p (the arguments will work for all prime numbers p, we fix it in order to simplify the notation). We want to decompose L_p to get a simpler automaton A'. Let $L_{simple} = L(A')$. Moreover, we will be looking for an advisory language L_{adv} and an a-transducer M. Let $L_{trans} = M^{-1}(L_{adv})$.

Now, we present some constraints on aforementioned languages. From the definition of the framework, we know, that $L[L_{trans}](A') = L_p$ and therefore $L_p = L_{simple} \cap L_{trans}$. We claim, that $\mathscr{C}_{state}(L_{simple}) \geq p$ or $\mathscr{C}_{state}(L_{trans}) \geq p$. This can be proven using a series of arguments, which have been already used couple of times in our thesis - since both languages must contain L_p as their subset, if both finite automata have fewer than p states, their computation would run in a cycle of some lengths k, l. Then, both automata would accept some word extended by a suitable common multiple of k and l symbols a, which however does not belong to L_p.

On the other side, since we claim, that L_p is T-decomposable, it must hold, that $\mathscr{C}_{state}(L_{simple}) < p-1$ (together with another two devices, the total number of states is at most p). It follows, that $\mathscr{C}_{state}(L_{trans}) \geq p$. What do we know about the complexity of L_{adv}? For similar reasons as for L_{simple}, also for L_{adv} it has to hold, that $\mathscr{C}_{state}(L_{adv}) < p - 1$.

That means, that in fact, we want to encode the language L_{trans} into the language L_{adv} with a smaller complexity using an a-transducer M. However, we not only need, that $M(L_{trans}) = L_{adv}$. Lemma 14 gives us another supplementary conditions on M. We want to find two dual a-transducers M, M', such that $M(L_{adv}) = L_{trans}$, $M'(L_{trans}) = L_{adv}$ and both $\mathscr{C}_{state}(M) < p$ and $\mathscr{C}_{state}(M') < p$. Besides, $M(L^c_{adv}) \cap L_{trans} = \emptyset$ and $M'(L^c_{trans}) \cap L_{adv} = \emptyset$ (this is just another notation of conditions from Lemma 14).

Now, let us consider an a-transducer M' with aforementioned properties and the language $M'(L_{trans})$. We know, that L_{trans} contains all words of the form $(a^p)^*$ and all this words have to be transduced by M' to words from $L_{adv}(M'(L_{trans}))$. Since M' has fewer than p states, the computation of M' on such words contains a cycle. Without loss of generality, take one of the accepting computations on a^p and let the shortest cycle with output have length c (if no

cycle has an output, $M'(a^p) = M'(a^{p+c})$, which violates the condition in Lemma 14). Let the output of this cycle be $x \neq \varepsilon$. Moreover, assume, that the word generated by this computation is w.

Now, we know, that $w \in L_{adv}$. M' gives correct outputs on all words from L_{trans}, so also if we iterate the cycle few more times, we get to the same accepting state. The substring x can be anywhere in the word w, however, without loss of generality, we may assume, that it is in the end. Otherwise the arguments are the same, but with slithly more complex notation. So, we assume $w.(x^{k.p}) \in L_{adv}$ for any $k \geq 0$ (because $w.(x^{k.p}) \in M'(a^{p+k.p.c})$). Consider words of the form $u_i = a^{p+i.c}$ and $v_i = w.(x^i)$ for all $i > 0$. We know, that $v_i \in M'(u_i)$. Using the equivalence relation from Myhill-Nerode theorem (see Section 2), we show, that the number of equivalence classes of L_{adv} is bigger than p, which contradicts our assumptions.

We claim, that each of the words v_i for $1 \leq i \leq p$ yields another equivalence class. Assume there is a pair of indices $k, l; 1 \leq k < l \leq p$, such that v_k and v_l fall into the same equivalence class. Let m be the smallest number such, that $m > p \wedge v_m$

We will call a number n "bad", if $v_k.(x^n) \notin L_{adv}$. We know, that $m-k$ and $m-l$ are bad. For the sake of simplicity, let $r = l - k$. Since $r < p$, at least one of $m - k, m - l$ is not a multiple of p. Without loss of generality, let it be $m - k = s$ (the other case is very similar). v_k and v_l are in the same equivalence class, so if s is bad, also $s + r$ is bad. For the same reason, if $s + r$ is bad, also $s + 2r$ is bad and all $s + t.r$ for $t \geq 0$ are bad. However, from the group theory we know, that $\mathbb{Z}_p$ is a cyclic group, where every $i \neq 0$ is a generator. Therefore, also r is a generator and for some j, $j.r$ is the inverse element to $s + k$. However, this would mean, that $v_{j.r+(s+k)} \notin L_{adv}$, which further means, that $u_{j.r+(s+k)} \in L_{trans}$, while $j.r + (s + k)$ is divisible by p, which leads to a contradiction.

We need to examine one additional case - if $\forall i, p \leq i : v_i \in L_{adv}$. This means, that $\forall i, p \leq i : u_i = a^{p+i.c} \in L_{trans}$. Though, we have seen, that the automaton A' for L_{simple} has less than p states and again, it runs in a cycle of length $d < p$ when accepting words from L_{dec}. Hence a word $a^{p+d.c} \in L_{simple}$. As we have seen, also $a^{p+d.c} \in L_{trans}$, but then $a^{p+d.c} \in L_p$, which is a contradiction, because both $b, c < p$ and p is a prime number. □

As we have seen, the classes of regular languages concerning T-decomposability are different as the classes of A-decomposable and A-undecomposable languages. In the next part of our thesis, we investigate some properties of these classes.

4.3 Closure properties

When looking at a new class of languages, one of the first natural question, that arises, are its closure properties. In this section, we want to examine the closure of T-decomposable and T-undecomposable languages under some basic operations and then under deterministic operations presented in [10].

4.3.1 T-undecomposable languages

In this part, we mainly use two types of T-undecomposable languages. First of them are languages of type $L_p = \{a^{pk} | k \geq 0\}$ for p a prime number. The T-undecomposability of these languages is proved in previous Section. The second type is a language $L = \{a\}^*$. This language is clearly undecomposable, since $\mathscr{C}_{state}(L) = 1$ and all three devices contained in our foreign advisor concept have non-zero number of states.

Theorem 18. The class of T-undecomposable languages is not closed under

1. complement,
2. (non-erasing) homomorphism,
3. inverse homomorphism,
4. Kleene star, Kleene plus,
5. intersection,
6. union.

Proof.

1. najst schodny dokaz, posledne dva nevysli
2. Consider an undecomposable language $L_1 = \{a^{13k} | k \geq 0\}$ and a homomorphism $h : \{a\}^* \to \{a\}^*$, such that $h(a) = aa$. Clearly, $h(L_1) = \{a^{26k} | k \geq 0\}$ and this language can be decomposed in a following way: let us take an a-transducer M_1 computing the identity mapping and a language $L_1' = \{a^{2k} | k \geq 0\}$. This two items form the desired effective advice for L_1, since we only have to chceck the language $\{a^{13k} | k \geq 0\}$.

   Since this homomorphism is non-erasing, our class is not closed even under this kind of mapping.
3. ToDo: najst nejaky schopny protipriklad

4. ToDo: dobra otazka, mozno nakoniec aj bude - ak ma jazyk tvar L^*, potom asi musi byt v automate prechod akceptacny -> pociatocny a tu to vieme roztrhnut a potom ak sa dal zjednodusit ten cely, tak sa musi dat aj ten maly. lenze, tazko povedat, mozno sa ten jazyk tak nejak moze zvrhnut, ze sa to zrazu bude dat rozkladat. zistit!

5. Consider two languages, $L_{41} = \{a^{13k} | k \geq 1\}$ and $L_{42} = \{a^{2k} | k \geq 1\}$. As stated before, both of these languages are T-undecomposable. However, $L_{41} \cap L_{42} = \{a^{26k} | k \geq 1\}$ is a T-decomposable language, as we have seen in the first part of this proof.

6. Let us take two languages, $L_{51} = \{a^k | k \neq 0(mod5)\}$ and $L_{52} = \{a^k | k \neq 0(mod7)\}$. The T-undecomposability of these languages can be shown in very similar way, which we have seen in Theorem 17. We show just the first part of the proof, since the rest follows the same pattern as in aforementioned result.

   Assume, that L_{51} is T-decomposable (the same train of thoughts can be used for L_{52}). This means, that we can decompose L_{51} into two languages L_{trans} and L_{simple}, such that $L_{trans} \cap L_{simple} = L_{51}$. As we see, $L_{51} \subseteq L_{simple}$. However, since the finite automaton for L_{51} has fewer than 5 states, it is easy to see, that $L_{simple} \supseteq \{a^*\}$ (otherwise at least one of the words $a, aa, aaa, aaaa$ would be rejected). It follows, that $L_{trans} \cap \{a^k | k = 0(mod5)\} = \emptyset$. As we have said, the rest of the proof is almost identical to the proof of Theorem 15.

   Now we claim, that the language $L_{53} = L_{51} \cup L_{52} = \{a^k | k \neq 0(mod35)\}$ is T-decomposable. Clearly, $\mathscr{C}_{state}(L_{53}) = 35$. Take the a-transducer M_5 from Image 1. It can be seen, that $M_5(a^{5k}) = \{b^k\}$ and $\forall k \neq 5 : M_5(a^k) = \{a^k\}$. Now, let $L_{adv} = \{w \in \{a, b\}^* | \nexists i : w = b^i \wedge i = 0(mod7)\}$. Then, $L_{simple} = \{a\}^*$ and clearly, (L_{adv}, M_5) is an effective advice with regard to L_{53}.

□

4.3.2 T-decomposable languages

Theorem 19. The class of T-decomposable languages is not closed under

1. complement,
2. (non-erasing) homomorphism,
3. inverse homomorphism,
4. Kleene star, Kleene plus,

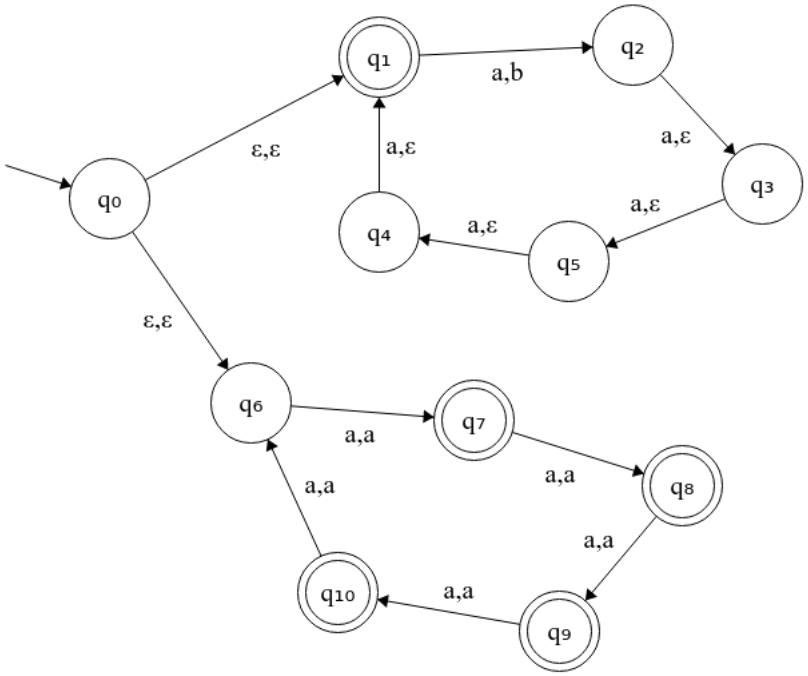

Figure 4.1: Image 1: A-transducer M_5

5. intersection,

6. union.

Proof.

1. This claim follows directly from previous Theorem.

2. Let us take a language $L_1 = \{w | w \in \{a,b\}^* \wedge \#_a(w) \mod 42 \equiv 0\}$. Clearly, a language $L_1' = \{w | w \in \{a,b\}^* \wedge \#_a(w) \mod 14 \equiv 0\}$ with an a-transducer M_1 computing the identity mapping is an effective advice for L_1.

   Let us now consider a homomorphism $h : \{a,b\}^* \to \{a\}^*$, defined as $h(a) = a, h(b) = a$. Note, that h is a non-erasing homomorphism. It easy to see, that $h(L_1) = \{a\}^*$, however, as stated before, this language is T-undecomposable.

3. Consider a language $L_2 = \{a^{26k} | k \geq 1\}$. The decomposition of this language was shown in the proof of previous theorem. The desired homomorphism is $h : \{a\}^* \to \{a\}^*$, where $h(a) = aa$. Now, $h^{-1}(L_2) = \{a^{13k} | k \geq 1\}$, which is T-undecomposable.

4. The counterexample is a language $L_3 = \{a^9\}$. Let us take a language $L_3' = \{a^5\}$; an a-transducer $M_3 = (\{q_0, q_1, q_2\}, \{a\}, \{a\}, H, q_0, \{q_1\})$, where $H = \{(q_0, a, a, q_1), (q_1, a, \varepsilon, q_2),$

$(q_2, a, a, q_1)\}$; and an automaton $A_3 = \{q_0, \{a\}, \delta, q_0, \{q_0\}$, where $\delta(q_0, a) = q_0$. Clearly, $M_3^{-1}(L_3') = L_3$ and $\mathscr{C}_{state}(L_3') + \mathscr{C}_{state}(T) + \mathscr{C}_{state}(A_3) = 5 + 3 + 1 \leq 9 = \mathscr{C}_{state}(L_3)$, therefore L_3 is T-decomposable. Though, $(L_3)^+ = \{a^{9k} | k \geq 1\}$ and $(L_3)^* = \{a^{9k} | k \geq 0\}$ are T-undecomposable.

5. Let us take a look at two languages, $L_{41} = \{a^{9k} | k \geq 1\} \cup \{a^{8k} | k \geq 1\}$ and $L_{42} = \{a^{9k} | k \geq 1\} \cup \{a^{12k} | k \geq 1\}$. We show the decomposition of L_{41}, since that of L_{42} is very similar.

   Let $L_{41}' = \{a^{9k} | k \geq 1\} \cup \{a^{2k} | k \geq 1\}$ and M_{41} compute the identity mapping. With this advice, we need to check just the language $L_{41}'' = \{a^{9k} | k \geq 1\} \cup \{a^{4k} | k \geq 1\}$ with an automaton A_{41}''. It is easy to see (and provable by Myhill-Nerode theorem), that a DFA for language L_{41} needs at least $9.8 = 72$ states. However, $\mathscr{C}_{state}(L_{41}') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{41}'') = 18 + 1 + 32 = 51$ and clearly $L(A_{41}'', L_{41}') = L_{41}$, which means, that L_{41} is T-decomposable.

   However, if we take the language $L_4 = L_{41} \cap L_{42} = \{a^{9k} | k \geq 1\}$, we get a T-undecomposable language, therefore our class is not closed under intersection.

6. In previous section we have seen, that languages $L_{61} = \{a^{10}\}$ and $L_{62} = \{a^{12}\}$ are T-decomposable. Now we show, that also their complements are T-decomposable. Let $M_6 = (\{q_0, q_1, q_2\}, \{a\}, \{a, b\}, H, q_0, \{q_0, q_2\})$, where $H = \{(q_0, a, \varepsilon, q_1), (q_1, a, b, q_0), (q_0, a, a, q_2)\}$. It can be easily seen, that $M(a^{2k}) = b^k$ and $M(a^{2k+1}) = b^k a$. Now, the effective advice for L_{61}^c consits of M and $L_{61,adv} = \{b^5\}^c$. Clearly, $\mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{61,adv}) + \mathscr{C}_{state}(\{a\}^*) = 3 + 7 + 1 \leq 12 = \mathscr{C}_{state}(L_{61}^c)$ and $L_{61}^c = M^{-1}(L_{61,adv}) \cap \{a\}^*$. The effective advice for L_{62}^c can be constructed in the same way.

   However $L_{61}^c \cup L_{62}^c = \{a\}^*$ and since $\mathscr{C}_{state}(\{a\}^*) = 1$, this language is T-undecomposable.

□

4.3.3 Deterministic operations

As we have seen, the situation with closure propertie of T-decomposable languages is not very interesting. However, this changes, if we are considering the deterministic version of operations (Kleene star/plus, union).

Definition. Let $L \subseteq \Sigma^*$ be a language and $c \notin \Sigma$ a new symbol. Then, a *marked Kleene star (plus)* of L is a language $\{\varepsilon\} \cup L.(cL)^*$ ($L.(cL)^*$). [je takato definicia prilis zvrhla? lebo normalna robi trocha problemy s pridavanim pociatocneho stavu]. We denote it as $L^{c*}(L^{c+})$.

Definition. Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be languages and $a \notin \Sigma_1$ and $b \notin \Sigma_2$ two new symbols. A *marked union* of L_1 and L_2 is a language $aL_1 \cup bL_2$.

ToDo: napisat nejaky vhodny pokec o tom, ze neskumame len jazyk samotny, ale previazany s prekladom (a mozno dokonca s pomocnym jazykom)

Theorem 20. The class of T-decomposable languages is closed under marked Kleene star/plus.

Proof. Let L be a T-decomposable language with an effective advice (M, L_{adv}). Suppose, that we use the advice and check L using an automaton A_{simple}. Now consider a language $L' = L^{c*}$. First, we show, that $\mathscr{C}_{state}(L') \geq \mathscr{C}_{state}(L)$.

Let $A' = (K', \Sigma \cup \{c\}, \delta', q'_0, F')$ be a minimal DFA, such that $L(A') = L'$. We claim, that $\forall q' \in F$, there is a transition in form of $\delta(q', c) = p'_{q'}$. Assume this is not the case and for a state q'', there is no such transition. Since A' is minimal and q'' is accepting, there is an input $w \in L'$, such that the computation of A' on w ends in q''. Now, consider a word $w' \in L$ and an input $wcw' \in L'$. Since there is no c-transition from q'' and A' is deterministic, wcw' will not be accepted, which is a contradiction with the condition, that $L(A') = L'$.

For similar reasons, we claim, that for any of aforementioned states $p'_{q'}$, the computation of A' on a correct input word $w' \in L$ has to be accepting (otherwise there would again be an input wcw', which would not be accepted). Let us pick arbitrary one of $p'_{q'}$ and denote it by p'.

Because of these reasons, we can construct an automaton for L as follows: $A = (Q', \Sigma, \delta, p', F')$, where $\delta = \delta' - \{\text{transitions on } c\}$. Clearly, as shown in the previous paragraph, $L(A) = L$ and therefore $\mathscr{C}_{state}(L') \geq \mathscr{C}_{state}(L)$.

Now we show the decomposition of L'. Let us take $L'_{adv} = L^{c*}_{adv}$ and we can get M' from M adding transitions (p, c, c, q_0) for each $p \in F$ and q_0 the initial state. Now we claim, that the pair (T', L'_{adv}) forms an effective advice for L'. Clearly, $M'^{-1}(L'_{adv}) = (M^{-1}(L_{adv}))^{c*}$ (since the computation possibilities of the tranducer between markers c is the same as before) and therefore we only have to check the language $L(A_{simple})^{c*}$.

It remains to show that this advice really is effective. To do this, we claim, that for any regular language R, $\mathscr{C}_{state}(R^{c*}) \leq \mathscr{C}_{state}(R)$, since we can get an automaton for R^{c*} by adding c-transitions from accepting states to the initial state to the minimal DFA for R. Moreover, clearly $\mathscr{C}_{state}(M') = \mathscr{C}_{state}(M)$. Therefore, $\mathscr{C}_{state}(L^{c*}) \geq \mathscr{C}_{state}(L) \geq \mathscr{C}_{state}(L_{adv}) + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(A_{simple}) \geq \mathscr{C}_{state}(L'_{adv}) + \mathscr{C}_{state}(M') + \mathscr{C}_{state}(A'_{simple})$ and therefore $\mathscr{C}_{state}(L^{c*}) \geq \mathscr{C}_{state}(L'_{adv}) + \mathscr{C}_{state}(M') + \mathscr{C}_{state}(A'_{simple})$, quod erat demonstrandum. □

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.

[9] P. Gaži. *Parallel Decompositions Of Finite Automata*. Master's thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava (2006).

[10] W. J. Chandler. *Abstract families of deterministic languages*. Proceedings of the first annual ACM symposium on Theory of computing (STOC '69). ACM, New York, NY, USA (1969), 21-30.