Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Complexity of Language Transformations

Diploma Thesis

2013

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, 2013

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

In last fifty years, 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 an output tape and the image 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, 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 acknowledged with 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 would like to clarify some basic notations and terminology used in our thesis.

1.1 Basic concepts and notation

Notation. We denote

- by ϵ 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 symbols a in a word w,
- if $u \equiv a_1a_2...a_m$, $v \equiv b_1b_2...b_n$, then by $u.v$ or simply uv we denote a word $a_1a_2...a_mb_1b_2...b_n$,
- 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. Σ 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 $h : \Sigma_1^* \to \Sigma_2^*$, such that

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

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

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

Definition. An *inverse homomorphism* is a function $h^{-1} : \Sigma_1^* \to 2^{\Sigma_2^*}$, 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 ϵ-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 used families of language, included in 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 $\mathcal{L}_{\mathcal{CF}}$, 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

Now we would like to 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 chapter, 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 a-transducer is 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,
- Σ_1 and Σ_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 ϵ-*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. 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. An *image* of language L by a-transducer M is a set

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

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 $h_0 h_1 ... h_m \in H^*$, 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 as

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

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

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

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,
- 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, 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 δ (σ) 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

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

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*.

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 $\mathcal{L}_{\mathcal{CF}}$ 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 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 generalizations 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 $(q, u, v, p) \in H_1, u \equiv a_1 a_2 ... a_m, v \equiv b_1 b_2 ... b_n$. 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 $(q_{a_i}, a_{i+1}, b_{i+1}, a_{a_{i+1}})$ for $1 \leq i < n$, resp. $(q_{a_j}, a_{j+1}, \epsilon, a_{a_{j+1}})$ for $n < j < m$. 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 $\forall L : M_1(L) = M_2(L)$. □

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 (ϵ-free) a-transducers M_1 and M_2 there exists an (ϵ-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 $\forall L : M_3(L) = M_2(M_1(L))$. □

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

Lemma 6. For every $R \in \mathcal{R}$, 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,

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

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 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 $\mathcal{M}(\mathcal{L})$ ($\hat{\mathcal{M}}(\mathcal{L})$) is the smallest (full) semi-AFL containing language L.

2.2 Existence of an A-transducer for a Pair of Languages

Here we present few results regarding the question, when it is possible to transform a language L_1 to a language L_2.

2.2.1 Regular Languages

The answer to aforementioned question, when dealing with regular languages, is very simple.

Theorem 8. For every regular language R and arbitrary language L there exists an a-transducer M, such that $M(L) = R$.

Proof. Let $A = (K, \Sigma, q_0, \delta, F)$ be a non-deterministic finite automaton, such that $L(A) = R$. Then $M = (K, \Sigma_1, \Sigma, H, q_0, F)$, where $H = \{(q, \epsilon, a, \delta(q, a)) | q \in K, a \in \Sigma\} \cap \{(q_0, a, \epsilon, q_0) | a \in \Sigma_1\}$. $\square$

2.2.2 Context-free Languages

Contrary to the case of regular languages, it has showed itself, that the task to determine for two context-free languages U and V, if $U \in \mathcal{M}(V)$, is not trivial to solve at all. Actually, it is quite easy to show, that $U \in \mathcal{M}(V)$, but to prove the revers, i. e. $U \notin \mathcal{M}(V)$, is relatively difficult. For this reason, it is convenient to consider only a special subclass of $\mathcal{L}_{\mathcal{CF}}$, called bounded languages.

Definition. A language is called *bounded*, if $L \subseteq w_1 w_2 ... w_n$ for some words $w_1, w_2, ..., w_n$.

Moreover, we consider only bounded languages, where $\forall i, j, 1 \leq i, j \leq n : |w_i| = 1 \wedge w_i \neq w_j$.

Following theorems show some necessary and sufficient conditions for existence of an a-transduction between languages U and V. These results were achieved with extensive use of the AFL theory, which is motivated by the first lemma. Major part of these results are not accompanied with a proof, since they are listed only for illustration of properties of a-transducers and do not form the main aim of our thesis.

Notation. Let $n \geq 2$ and f_i be a strictly increasing function (i. e. $x_1 < x_2 \Rightarrow f_i(x_1) < f_i(x_2)$) from $\mathbb{N}$ to $\mathbb{N}$ for all $1 \leq i \leq n$. By

$$\langle a_1^{f_1(x)} ... a_n^{f_n(x)} \rangle$$

we denote the set $\{a_1^{f_1(x)} ... a_n^{f_n(x)} | x \in \mathbb{N}\}$.

Notation. By $\mathcal{F}(L)$ ($\hat{\mathcal{F}}(L)$) we denote the smallest (full) AFL containing L.

Lemma 9. Let $L_1 = \langle b_1^{g_1(x)} ... b_n^{g_n(x)} \rangle$ and L_2 be a nonempty language. $L_1 \in \mathcal{F}(L_2) \Leftrightarrow L_1 \in \mathcal{M}(L_2)$ and $L_1 \in \hat{\mathcal{F}}(L_2) \Leftrightarrow L_1 \in \hat{\mathcal{M}}(L_2)$.

Notation. By $\psi_{\langle a_1,...,a_n \rangle}$ we denote a mapping from $a_1^*...a_n^*$ into $\mathbb{N}^n$ defined as follows:

$$\psi_{\langle a_1,...,a_n \rangle (w)} = (\#_{a_1}(w), ..., \#_{a_n(w)}).$$

Notation. Let $c = (c_1, c_2, ..., c_n)$ and $l = (l_1, l_2, ..., l_n) \in \mathbb{N}^n$ and let $L_1 \subseteq a_1^* a_2^* ... a_n^*$. Let

$$\mathcal{K}(L, c, l) = \{k_1...k_n | a_1^{c_1+k_1.l_1} ... a_n^{c_n+k_n.l_n} \in L, \forall i : k_i \geq 0\} \qquad .$$

Theorem 10. Let $U \subseteq a_1^*...a_n^*$ and $V = \langle b_1^{g_1(x)}...b_m^{g_m(x)} \rangle$ [with $\epsilon \in U \Leftrightarrow \epsilon \in V$]. Then V is in $\mathcal{F}(U) \Leftrightarrow$

$V = \bigcup_{i=1}^{q}$ for some $q \geq 1$, each V_i of the form

$$\psi^{-1}_{\langle b_1,...,b_n \rangle}[\{(d_{i1} + \sum_{j=z_{i1}}^{z_{i2}-1} k_j p_{ij}, ..., d_{i1} + \sum_{j=z_{im}}^{z_{im+1}-1} k_j p_{ij}) | (k_1, ..., k_n) \in \mathcal{K}(L, c, l)\}],$$

where $c_i, l_i = (l_{i1}, ..., l_{in}), (p_{i1}, ..., p_{in}) \in \mathbb{N}^n$, all $l_{i,j} > 0, (d_{i1}, ..., d_{im}) \in \mathbb{N}^m$ and $(z_{i1}, ..., z_{im+1}) \in \mathbb{N}^{m+1}, 1 = z_{i1} < ... < z_{im+1} = n + 1$.

As one can see, this condition is quite difficult to put to use, when dealing with a concrete two languages. Fortunately, if both U and V are restricted to a form $\langle a_1^{f_1(x)}...a_n^{f_n(x)} \rangle$, some easier applicable conditions arise. In fact, we can slightly relax this restriction to following.

Notation. Let $p \in \mathbb{N}$, $n \geq 2$ and f_i be a strictly increasing function $\underline{\text{for } x \geq p}$ (i. e. $x_1 < x_2 \Rightarrow f_i(x_1) < f_i(x_2)$) from $\mathbb{N}$ to $\mathbb{N}$ for all $1 \leq i \leq n$. By

$$\langle a_1^{f_1(x)}...a_n^{f_n(x)} \rangle_p$$

we denote the set $\{a_1^{f_1(x)}...a_n^{f_n(x)} | x \in \mathbb{N}\}$.

Note, that $\langle a_1^{f_1(x)}...a_n^{f_n(x)} \rangle = \langle a_1^{f_1(x)}...a_n^{f_n(x)} \rangle_0$.

For U and V of this form, we present few necessary conditions and then some examples of their use.

Notation. The function $f_i, 1 \leq i \leq q$, is a *largest element* of $\{f_s | 1 \leq s \leq q\}$, if $\lim_{x \to \infty} \frac{f_i(x)}{f_j(x)}$ exists and is nonzero for all $j, 1 \leq j \leq q$.

Theorem 11. Let $U = \langle a_1^{f_1(x)}...a_n^{f_n(x)} \rangle_{p_1}$ and $V = \langle b_1^{g_1(x)}...b_m^{g_m(x)} \rangle_{p_2}$, with $V \in \hat{\mathcal{F}}(U)$. Then there exists a set $Q \subseteq \{1, ..., n\}$ and integers $z_1, ..., z_{m+1}, 1 = z_1 < ... < z_{m+1} = n + 1$ with the following two properties.

1. $Q_i = \{s | z_i \leq s < z_{i+1}, s \notin Q\} \neq \emptyset$ for all $i, 1 \leq i \leq m$.

2. For all integers i and j, $1 \leq i, j \leq m$, and for all positive real numbers k and l the following holds. Suppose there exist a largest element f_i' of $\{f_s | s \in Q_i\}$ and a largest element f_j' of $\{f_s | s \in Q_j\}$. Then
$$\lim_{x\to\infty} \frac{(g_i(x))^k}{(g_j(x))^l} > 0 \Leftrightarrow \lim_{x\to\infty} \frac{(f_i(x))^k}{(f_j(x))^l} > 0$$
if both limits exist.

If $m = n$, there holds a corollary, which can be quite easily used to show, that $V \notin \hat{\mathcal{F}}(U)$.

Corollary 11.1. Let $U = \langle a_1^{f_1(x)} ... a_n^{f_n(x)} \rangle_{p_1}$ and $V = \langle b_1^{g_1(x)} ... b_n^{g_n(x)} \rangle_{p_2}$, with $V \in \hat{\mathcal{F}}(U)$. Then for every pair of integers $i, j, 1 \leq i, j \leq n$, and for all positive real numbers k and l
$$\lim_{x\to\infty} \frac{(g_i(x))^k}{(g_j(x))^l} > 0 \Leftrightarrow \lim_{x\to\infty} \frac{(f_i(x))^k}{(f_j(x))^l} > 0$$
if both limits exist.

Example. Let $U = \langle a_1^{2^{3x}} a_2^{2^{2x}} a_3^{2^x} \rangle$ and $V = \langle b_1^{2^{4x}} b_2^{2^{3x}} \rangle$. Suppose $V \in \mathcal{F}(U)$. By theorem 11, $Q = \emptyset$, there are only two possible choices for z_1, z_2 and z_3:

1. $z_1 = 1, z_2 = 2$ and $z_3 = 4$. Then $f_1'(x) = 2^{3x}$ and $f_2'(x) = 2^{2x}$. For $k = 4, l = 3, i = 2$ and $j = 1$,
$$\lim_{x\to\infty} \frac{(g_i(x))^k}{(g_j(x))^l} = \frac{(2^{3x})^4}{(2^{4x})^3} = 1 \text{ and } \lim_{x\to\infty} \frac{(f_i'(x))^k}{(f_j'(x))^l} = \frac{(2^{2x})^4}{(2^{3x})^3} = 0 .$$
Thus Theorem 11 does not hold.

2. $z_1 = 1, z_2 = 3$ and $z_3 = 4$. Then $f_1'(x) = 2^{3x}$ and $f_2'(x) = 2^x$. For $k = 4, l = 3, i = 2$ and $j = 1$,
$$\lim_{x\to\infty} \frac{(g_i(x))^k}{(g_j(x))^l} = \frac{(2^{3x})^4}{(2^{4x})^3} = 1 \text{ and } \lim_{x\to\infty} \frac{(f_i'(x))^k}{(f_j'(x))^l} = \frac{(2^x)^4}{(2^{3x})^3} = 0 .$$
Thus Theorem 11 does not hold and therefore $V \notin \mathcal{F}(U)$.

As we have seen, Theorem 11 gave us an interesting result in form of example above, but since this condition is only necessary, but not sufficient, in some cases we need another conditions to show, that $V \notin \hat{\mathcal{F}}(U)$. Such a couple of languages is for example $U = \langle a^{(x+1)^2} b^{x^2} \rangle$ and $V = \langle a^{x^2} b^{x^2} \rangle$. In this case, following theorem works.

Theorem 12. Let $U = \langle a_1^{f_1(x)} ... a_n^{f_n(x)} \rangle_{p_1}$ and $V = \langle b_1^{g_1(x)} ... b_m^{g_m(x)} \rangle_{p_2}$, with $V \in \hat{\mathcal{F}}(U)$. Then there exists a set $Q \subseteq \{1, ..., n\}$ and integers $z_1, ..., z_{m+1}, 1 = z_1 < ... < z_{m+1} = n + 1$ with the following two properties.

1. $Q_i = \{s | z_i \leq s < z_{i+1}, s \notin Q\} \neq \emptyset$ for all $i, 1 \leq i \leq m$.

2. Let $i \in \{1, .., m\}$ and real numbers $q_1, ..., q_m, l, l > 0$, be such that

(a) there exists a largest element f_i' of $\{f_s | s \in Q\}$,

(b) $\lim_{x\to\infty} \frac{\sum_{s=1}^{m} q_s g_s(x)}{(g_i(x))^l}$ exists, and

(c) $\lim_{x\to\infty} \frac{\sum_{s=1}^{n} r_s f_s(x)}{(f_i'(x))^l}$ exists for all real numbers r_s which satisfy $sgn(r_s) = sgn(q_j)$ for all $s \in Q_j, 1 \leq j \leq m$, and $r_s = 0$ for all $s \in Q$ (sgn is a signum function).

Then there exists real numbers $v_1, ..., v_n$ ($sgn(v_s) = sgn(q_j)$ for all $s \in Q_j, 1 \leq j \leq m$, and $v_s = 0$ for $s \in Q$) for which

$$\lim_{x\to\infty} \frac{\sum_{s=1}^{m} q_s g_s(x)}{(g_i(x))^l} = \lim_{x\to\infty} \frac{\sum_{s=1}^{n} v_s f_s(x)}{(f_i'(x))^l}.$$

With help with this theorem, we can now present aforementioned example.

Example. Using Theorem 12, $Q = \emptyset, z_1 = 1, z_2 = 2, z_3 = 3$. Let $q_1 = 1, q_2 = -1, i = 2$ and $l = \frac{1}{2}$. Clearly, (a), (b) and (c) of 2 of Theorem 12 holds. However,

$$\lim_{x\to\infty} \frac{g_1(x) - g_2(x)}{(g_2(x))^{1/2}} = \lim_{x\to\infty} \frac{x^2 - x^2}{(x^2)^{1/2}} = 0$$

and

$$\lim_{x\to\infty} \frac{v_1 f_1(x) - v_2 f_2(x)}{(f_2'(x))^{1/2}} = \lim_{x\to\infty} \frac{v_1 (x+1)^2 - v_2 x^2}{(x^2)^{1/2}} \neq 0$$

for every choice of nonzero real v_1 and v_2. Therefore $V \notin \hat{\mathcal{F}}(U)$.

2.3 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, 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.

2.3.1 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 13. Let L be a regular language over 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. $\square$

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 is 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 14. Let $L \subseteq \Sigma^*$ be a regular language and suppose there exists a set of pairs $P = \{(x_i, w_i) : 1 \le i \le n\}$ such that

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

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 $S = \{q | \exists i, 1 \le i \le n : \delta(q_0, x_i) \ni q\}$. For every i, there must by a state $p_i \in S$, such that $p_i \in \delta(p_0, x_i)$ and $\delta(p_i, w_i) \cap F \ne \emptyset$ (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, $\delta(p_i, w_i) \cap F \ne \emptyset \Leftrightarrow \delta(p_j, w_i) \cap F \ne \emptyset$. It follows, that $x_j w_i \in L$, which is contradiction with definition of P.

Since $|S| \ge n$, A has at least n states. □

2.3.2 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 \ 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 15. 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 induct the proof of this claim, since it is quite technical and is not of vast relevance itself.

As mention 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.

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.