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:

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

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 tuplet $(q, a) \in K \times \Sigma_1$, 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 $\mathcal{L}_{C\mathcal{F}}$ 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 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 $(q, u, v, p) \in H_1, u \equiv a_1a_2...a_m, v \equiv b_1b_2...b_n$. 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 $(q_{a_i}, a_{i+1}, b_{i+1}, a_{a_{i+1}})$ 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 nondeterministic 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 $\mathcal{M}(\{L\})$.

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. Most of these results come from [3].

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 Foemal Languages*. Elsevier Science Inc., New York, NY, USA, 1975.

[3] B. Rovan. *Proving containment of bounded AFL.* J. Comput. Syst. Sci. 11, 1 (August 1975), 1-55.