Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Using Transformation in Solving Problems with Supplementary Information

ACM SPY

2015

Boris Vida

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.

Abstract

The main task of this thesis is to examine the use of transformation in solving problems with supplementary information. The thesis is a continuation of the research devoted to various aspects of information. It belongs to the part, where the topic of the research is the "usefulness" of the information, while as a tool used for the measurement of the usefulness is the descriptional complexity of deterministic finite automata. The results presented to this date relied on the fact, that the supplementary information is in the form, which can be used directly.

In this thesis, we propose a framework for studying the possibility to transform the instance of a problem to match the format of the advisory information. A widely used and quite general transformation device is an a-transducer. It turned out, that it is not always convenient to allow the use of nondeterminism in transformations. Therefore, we examine also the case, that the transformation device is a sequential transducer. Both of these devices are defined and reviewed, e. g., in [2].

The thesis is devoted to the notion of information and to the question, how can its various aspects be defined, studied and measured. The proposed thesis develops a model for studying the usefulness of information also in cases, where it is not usable directly. This model allows to consider the earlier results about the usefulness of information as a special case and therefore allows a direct comparison with the previous model.

Keywords: language transformations, descriptional complexity, a-transducer, information

Abstrakt

Hlavným cieľom tejto práce je skúmať využitie transformácie pri riešení problémov s pomocnou informáciou. Práca je pokračovaním výskumu týkajúceho sa oblasti skúmania rôznych aspektov informácie. Zapadá do časti, v ktorej sa skúma "užitočnosť" informácie, pričom ako prostriedok pre meranie užitočnosti je používaná popisná zložitosť deterministických konečných automatov. Doterajšie výsledky sa spoliehali na skutočnosť, že pomocná informácie je vo formáte, ktorý vieme priamo využiť.

V našej práci navrhujeme prostredie pre štúdium možnosti transformovať inštanciu problému tak, aby zodpovedala formátu pomocnej informácie. Často používaným a pomerne všeobecným transformačným modelom je a-prekladač, no ukázalo sa, že nie je vždy vhodné umožniť použitie nedeterminizmu v transformácii. Preto sa zaoberáme aj prípadom, že transformačným modelom je sekvenčný prekladač. Oba tieto modely sú definované a popísané napr. v [2].

Práca je o výskume pojmu informácia a o tom, ako by sa jej rôzne aspekty dali definovať, skúmať a merať. V predloženej práci je vybudovaný rámec pre skúmanie užitočnosti informácie aj pre prípady, že nie je použiteľná priamo. Tento rámec umožňuje doterajšie skúmanie užitočnosti informácie chápať ako špeciálny prípad a teda umožňuje aj priame porovnanie s predchádzajúcim modelom.

KĽÚČOVÉ SLOVÁ: transformácie jazykov, popisná zložitosť, a-prekladač, informácia

Contents

Introduction

The main task of our thesis is to examine the use of transformation in solving problems with supplementary information. The thesis is a continuation of the research devoted to various aspects of information. It belongs to the part, where the topic of the research is the "usefulness" of information. The tool used for the measurement of usefulness is the descriptional complexity of deterministic finite automata. The results presented to this date relied on the fact, that the supplementary information is in the form, which can be used directly. However, from real life applications we know, that this is not always the case.

In this thesis, we propose a framework for studying the possibility to transform the instance of a problem to match the format of the advisory information. A widely used and quite general transformation device is an a-transducer. It turned out, that it is not always convenient to allow the use of non-determinism in transformations. Therefore, we examine also the case, that the transformation device is a sequential transducer. Both of these devices are defined and reviewed, e. g., in [2].

The thesis is devoted to the notion of information and to the question, how can its various aspects be defined, studied and measured. The proposed thesis develops a model for studying the usefulness of information also in cases, where it is not usable directly. This model allows to consider the earlier results about the usefulness of information as a special case and therefore allows a direct comparison with the previous model.

In the first chapter of our thesis we present some basic definitions and notation used in our thesis.

The second chapter contains known results concerning transformation models, state complexity and the previous work in the area of solving problems with supplementary informations on deterministic finite automata.

In the third chapter we examine the state complexity of a-transducers and present our own results regarding a particular class of languages, which will be further used in our thesis.

In the last chapter of our thesis we propose a framework for studying the possibility to transform the instance of a problem to match the format of the advisory information. We study the classes of languages regarding the possibility to simplify their complexity using the supplementary information. Moreover, we compare our results to those achieved for supplementary information without the use of transformation.

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 1.1. 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 1.2. 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 1.3. 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 1.4. 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 1.5. A (full) trio is called a *(full) semi-AFL*, if it is closed under union.

Definition 1.6. 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 1.7. 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 1.8. 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 1.9. 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 1.10. 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 1.11. 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 1.12. 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 1.13. 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 1.14. 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 1.15. 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 1.16. 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 1.17. 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

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

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

Definition 1.18. 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 1.19. The *state complexity* of an a-transducer $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ (a sequential transducer $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0, F)$, 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 1.20. 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 1.21. In a similar way, we can define the *sequential (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 sequential (a-)transducer M, 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 sequential (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 1.22. 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 1.23. 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 1.24. Let $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1)$ and $A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$ 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 1.25. 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 1.26. 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 $\mathcal{U}_{SB}$ ($\mathcal{U}_{ASB}$).

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 2.1. $\mathcal{R}$ 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 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 2.1.1. 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 2.2. 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 2.3. 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 2.4. 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 2.5. 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 2.6. 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 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 state count of its minimal 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 2.7. 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 denoted as $[x]_L$) 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.

A similar result was achieved for non-deterministic automata in [6]. However, the lower bound presented there 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, wheter there is an NFA with at most k states equivalent to a given DFA is $PSPACE$-complete ([7]). The following theorem was introduced in [6].

Theorem 2.8. 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 an 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 \ we denote the operation of a left quotient.

Definition 2.1. 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 2.9. 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 the number of equivalence classes of R_f. However, we do not present the proof of this claim, since it is quite technical and we shall not use these technicalities in our thesis.

As mentioned above, 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

When we consider advisory information which is "checkable" by finite automata, we can view acceptance using advice as (centrain type of) finite automata decomposition. We present here some definitions and results from [9].

Theorem 2.10. 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 2.2. 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 2.3. 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 2.4. 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 2.5. 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 can finally proceed to the necessary and sufficient condition on (A)SB-decomposability.

Theorem 2.11. 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 substitution 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

In this thesis we initiated the study of usefulness of information in situations, where the advisory information has to be transformed into some usable form (indirect advice, see Chapter 4). The transformation we shall consider are both deterministic and nondeterministic. It shall be crucial to know the complexity of the a-transducer involved. We identified a simple type of languages (see Section 3.1) which suffices to exhibit possible behaviour of such advice utilisation.

This section is thus devoted to 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 (with respect to the number of states). For our purposes it suffices to consider a special class of languages and present the results concerning these. This allows to present the basic concepts and opens a possibility to initiate similar study for other types of languages.

It is easy to see that 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 (nonempty) input. It suffices to take the minimal 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 $H = \{(p, \varepsilon, a, q) | \delta(p, a) = q\} \cup \{(q_F, a, \varepsilon, q_F) | a \in \Sigma, q_F \in F\}$. 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

We look for a simple class of languages, which would be used in the next chapter as a series of examples applied to show the basic properties of our framework. Such a suitable class of languages are "modular counting languages". By modular counting languages we understand languages in the form

$$L_k = \{a^k | k \equiv 0 \pmod{k}\}.$$

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 3.1. 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 corresponding 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 $q_i = q_j$, where $j < i + l$ (we assume that this is the shortest cycle in the computation, during which M generates a non-empty output). In this cycle, M reads a subword a^r and generates an output a^s for some r, s; $1 \leq r, s \leq l - 1$.

Now, let us take two longer inputs $x' = x.a^{k.r}$ and $x'' = x.a^{2k.r}$. On these two inputs, M generates outputs $y' = y.a^{k.s}$ and $y'' = y.a^{2k.s}$, 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. Since the case $\gcd(k,l) = 1$ was treated in 1, we can assume $\gcd(k,l) > 1$. Therefore, in what follows 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 the 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, such that $M'(L_k) = L_l$. Similarly to the proof of part 1., we look for a cycle, in this case of the length of at most $\frac{l}{\gcd(k,l)} - 1$ states. With very similar series of arguments, we can construct two inputs $x' = x.a^{k.r}$ and $x'' = x.a^{2k.r}$, which produce outputs $y' = y.a^{k.s}$ and $y'' = y.a^{2k.s}$, respectively. If both $|y'|, |y''|$ 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 we know from the number theory, $\mathrm{lcm}(k,l) = \frac{k.l}{\gcd(k,l)}$.

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, such that $M'(L_k) = L_l$, generates an incorrect output, but we claim, that it is not able to generate all correct outputs (i. e., all words from language L_l). Let us consider 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 $q'_i = q'_j$ for some i and j, where $j < i + \frac{k}{\gcd(k,l)}$. In each iteration of this cycle, M' has to output at least one symbol a.

We claim, that in each iteration 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 the 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 $l \leq \frac{k}{\gcd(k,l)}$.

□

As a direct consequence of Lemma 3.1 we obtain the following theorem.

Theorem 3.2. $\mathscr{C}_{state}(L_k, L_l) = \min(l, \frac{\max(k,l)}{\gcd(k,l)})$.

Chapter 4

Indirect Advice

4.1 Description of the Framework

We now proceed to definitions associated to the central matter of our thesis, which is the framework for using transformation in problem solving with advisory information. We enrich the framework studied in [9] by allowing the advice to be on some transformation of the input. We shall consider three particular cases. The first two cases shall be based on general a-transducers and the third case shall be based on deterministic sequential transducers. The results of this chapter provide the comparison of these cases.

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

$$M_{\forall}^{-1}(L) = \{w | M(w) \neq \emptyset \wedge M(w) \subseteq L\}.$$

Notation. Let M be an a-transducer and L be a language. Then $M_{\exists}^{-1}(L)$ is the set of words w, such that there is at least one image of w belonging to L. Formally

$$M_{\exists}^{-1}(L) = \{w | M(w) \cap L \neq \emptyset\}.$$

Example 4.1. Let $M = (\{q_0, q_1\}, \{a\}, \{b\}, H, q_0, \{q_0, q_1\})$ be an a-transducer, where $H = \{(q_0, a, b, q_1), (q_1, \varepsilon, b, q_0)\}$. Moreover, let $L = \{b^2\}^*$. Every word $a^k \in \{a\}^+$ has two images: $M(a^k) = \{b^{2k-1}, b^{2k}\}$ and $M(\varepsilon) = \{\varepsilon\}$. Therefore, $M_{\exists}^{-1}(L) = \{a\}^*$, while $M_{\forall}^{-1}(L) = \{\varepsilon\}$.

Definition 4.1. Let L_{dec} be a regular language. Let $\Psi \in \{\forall, \exists\}$. A pair (L_{adv}, M), where L_{adv} is a regular language and M an a-transducer is called an *indirect advice*. The indirect advice is called and NT_{Ψ}-advice for a regular language L_{dec} and an finite automaton A if there exists a deterministic finite automaton A', such that $L_{dec} = L[M_{\Psi}^{-1}(L_{adv})](A')$. Moreover, (L_{adv}, M) is called *effective*, if $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{adv}) \leq \mathscr{C}_{state}(L_{dec})$.

Notation. We remind, that $L[M_{\Psi}^{-1}(L_{adv})](A') = L(A') \cap M_{\Psi}^{-1}(L_{adv})$.

Example 4.2. 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 $L_{adv} = \{a^{2k} | k \geq 0\}$. M shortens every word from a^* to half of its length, so it is easy to see, that $M_{\forall}^{-1}(L_{adv}) = M_{\exists}^{-1}(L_{adv}) = \{a^{4k} | k \geq 0\}$. We now construct a simpler finite automaton A' for the language $L_{simple} = \{a^{3k} | k \geq 0\}$. Clearly, $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{adv}) = 3 + 2 + 2 \leq 12 = \mathscr{C}_{state}(L_{dec})$ and $L[M_{\forall}^{-1}(L_{adv})](A') = L[M_{\exists}^{-1}(L_{adv})](A') = L_{dec}$ which means, that L_{adv} with M is an effective $NT_{\forall}$- and $NT_{\exists}$-advice with regard to L_{dec}.

Example 4.3. Let $L_{dec} = \{a^{3k} | k \geq 0\} \cup \{a^{5k} | k \geq 0\}$. Now, let M be an a-transducer from Figure 4.1, $L_{adv} = L_{simple} = \{a\}^*$. We can see, that M has accepting computations only on words from L_{dec} and so $M_{\exists}^{-1}(L_{adv}) = M_{\forall}^{-1}(L_{adv}) = L_{dec}$. $\mathscr{C}_{state}(M) + \mathscr{C}_{state}(L_{simple}) + \mathscr{C}_{state}(L_{adv}) = 11$, while $\mathscr{C}_{state}(L_{dec}) = 15$ (this could be proven by Myhill-Nerode theorem), so (L_{adv}, M) is an effective $NT_{\forall}$- and $NT_{\exists}$-advice with regard to L_{dec}.

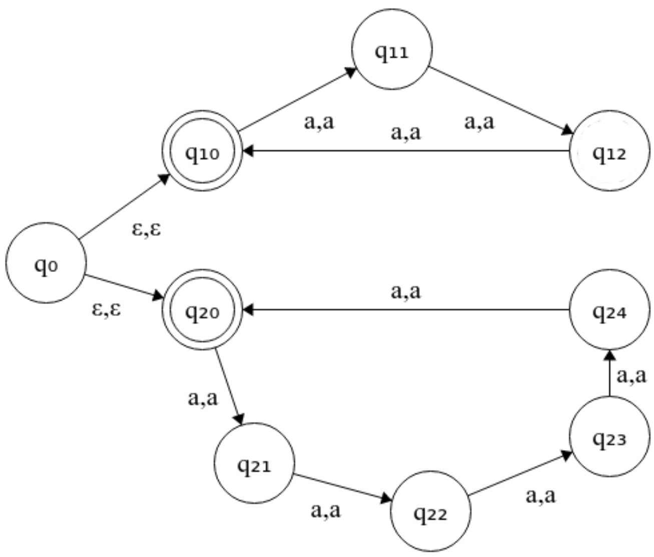

Figure 4.1: a-transducer M

In the last example, the whole advice was in some sense contained in the transformation and the efficiency was achieved just through the nondeterminism of the a-transducer. We would like to prevent such misuse of nondeterminism, so the saving of state count would mirror the actual possibility to disassemble the problem into some smaller subproblems, such that their results combined yield the solution of the task.

At first sight the previous example seems to imply, that the problem is due to the fact, that M does not have to generate the whole language L_{adv}. One possible solution is to add a condition, that the filtering of words not from L_{dec} should not happen only in M, but also in L_{dec} (and since the complexity of L_{dec} is the state count of its minimal deterministic automaton, the nondeterminism could not be misused). Formally, a pair (L_{adv}, M) is a ($NT_{\forall}$-) $NT_{\exists}$-advice with regard to L_{dec}, if it fulfills the condition from the original definitions and moreover, we demand, that $L_{adv} \subseteq M(\Sigma^*_{L_{dec}})$.

However, if we alter the a-transducer M, so that each traversal of the form (q_i, a, a, q_j) will be altered to $(q_i, a, \varepsilon, q_j)$, we can take $L_{dec} = \{\varepsilon\}$. Again, $M_{\exists}^{-1}(L_{adv}) = M_{\forall}^{-1}(L_{adv}) = L_{dec}$ and the complexity of the advice has increased by 1 (since $\mathscr{C}_{state}(\{\epsilon\}) = 2$), so (L_{adv}, M) is still an effective advice for L_{dec}. However, our problem with the misuse of nondeterminism is still apparent. Adding this simple condition did not help at all.

For aforementioned reason, we present the third possible definition of our framework, where the transformation model is not an a-transducer, but a (deterministic) sequential transducer. However, unlike by a-transducer, one-bounded sequential transducers are not a normal form of sequential transducers. It is easy to see, that with this restriction, we cannot generate an output word, which is longer than the input. However, this shortcoming can be easily solved by a small modification in the definition:

- δ is a partial transition function, that maps $K \times (\Sigma_1 \cup \{\varepsilon\}) \to K$,
- σ is a partial output function, that maps $K \times (\Sigma_1 \cup \{\varepsilon\}) \to \Sigma_2$,
- however, the ϵ-transition and ϵ-output in a state $q \in K$ are possible only if $q \in F$ (this transition has to be mandatory) there are no other transitions and outputs in this state, and
- since δ and σ are partial functions, we demand, that for $a \in \Sigma_1 \cup \{\varepsilon\}$ and $q \in K$, $\delta(q, a)$ is defined, if and only if $\sigma(q, a)$ is defined.

It can be easily shown, that this modified definition is a normal form of sequential transducers. The proof is straightforward and we do not include it in our thesis.

Definition 4.2. Let M be a sequential transducer and L be a language. Then $M_D^{-1}(L)$ is the set of all words, such that their images belong to L. Formally

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

Definition 4.3. Let L_{dec} be a regular language. A pair (L_{adv}, M), where L_{adv} is a regular language and M a sequential transducer is called a *T-advice with regard to L_{dec}*, if there exists a deterministic finite automaton A', such that $L_{dec} = L[M_D^{-1}(L_{adv})](A')$. 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 4.4. We can see, that the a-transducer M from Example 4.2 has neither ε-transitions, nor multiple transitions from one state on the same symbol. Moreover, the transition function is complete (the set H contains an element for every combination of a source state and an input symbol), so the corresponding sequential transducer M_D and its transition function δ and output function σ can be easily constructed. Therefore, the pair (L_{adv}, M) from Example 4.2 is an effective T-advice with regard to L_{dec}.

Remark. We shall often use this view of a sequential transducer - considering it a special case of an a-transducer. When it will be suitable, we shall identify a sequential transducer with an a-transducer in which the set H fulfills the aforementioned conditions (no ε-transitions and for each combination of state and input symbol precisely one element in H) without the formal definition of its δ and σ functions. We state their construction here.

$$\forall h \in H : \delta(pr_0(h), pr_1(h)) = pr_3(h)$$
$$\forall h \in H : \sigma(pr_0(h), pr_1(h)) = pr_2(h)$$

The correctness of the definition of these functions follows from the "determinism" of H.

We have defined three alternative ways to look at the use of transformation in solving problems with advisory information, which differ in the definition of the language $M^{-1}(L)$. This brings up the following question: for a given language L and an a-transducer M, how to get the languages $M_\forall^{-1}(L)$, $M_\exists^{-1}(L)$ and $M_D^{-1}(L)$? The answer was quite easy to find in previous two examples (and, in fact, for all languages in the form $\{(a^k)^*\}$ and transducers, which just manipulate the number of symbols a). We now search for the answer in general.

Lemma 4.1. Let $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ be an a-transducer and L be a language. Moreover, let $L' = M_\exists^{-1}(L)$. Then $\forall w \in L'^c : M(w) = \emptyset \vee M(w) \subseteq L^c$. The mapping $M_\exists^{-1}$ can be simulated by an a-transducer M' dual to M, such that $M'(L) = L'$, where $\mathscr{C}_{state}(M') = \mathscr{C}_{state}(M)$.

Proof. The first part is quite easy to see, since by definition, $M_\exists^{-1}(L)$ contains all words, such that at least one of their images by a-transducer M belongs 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.

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 $M_{\exists}^{-1}$, namely that $M'(L) = L'$ (since $L' = M_{\exists}^{-1}(L)$).

- $L' \subseteq M'(L)$: Take an arbitrary word $u \in L'$. By definition of $M_{\exists}^{-1}$, there is a word $v \in L$, such that $v \in M(u)$. Now, let us look at this 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), where $pr_1(h) = u$, $pr_2(h) = v$ and $q_F \in F$. We now present the computation of M', which shows, 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 correctness 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 the 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 $M_{\exists}^{-1}$ it follows, that $u \in M_{\exists}^{-1}(L) = L'$.

□

Very similar result can be stated for the setting with a sequential transducer. For a sequential transducer M and a language L, $M_D^{-1}(L) = M_{\exists}^{-1}(L)$, since every word $w \in M_D^{-1}(L)$ has exactly one image $M(w) \in L$. This means, that we can find L using the same dual a-transducer M'. Note, that this dual machine is not necessarily a sequential transducer, because the mapping by M is not necessarily injective (and even if it is, the functions δ and σ do not say anything about uniqueness of the combination of output symbol and resulting state). However, the determinism of the sequential transducer allows us to state some additional claims.

Lemma 4.2. Let $M = (K, \Sigma_1, \Sigma_2, \delta, \sigma, q_0, F)$ be a sequential transducer and L be a language. Moreover, let $L' = M_D^{-1}(L)$. Then, $M(L') \subseteq L$ and $\forall w \in L'^c : M(w) = \emptyset \vee M(w) \in L^c$. The mapping M_D^{-1} can be simulated by an a-transducer M' dual to the sequential transducer 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. We prove just those parts of our Lemma, which are different from the claims in the previous one. In the first part, we state, that $M(L') \subseteq L$. This claim follows directly from the fact, that every word $w \in L'$ has only one image $M(w)$ and by definition, this image belongs to L (otherwise $w \notin L'$). Moreover, since the image of a word w by a sequential transducer is a word, instead of a set, the condition on words from L'^c changes accordingly.

We provide the formal construction of the a-transducer M', since we construct it from the sequential transducer M. Again, $M' = (K, \Sigma_1, \Sigma_2, H, q_0, F)$, where

$$H = \{q, a, \sigma(a), \delta(q) | \forall q \in K, a \in \Sigma_1\}.$$

The proof of the claim, that $M'(L) = L'$, is very similar to the proof of previous Lemma. We provide the arguments for the last part of Lemma, i. e. $\forall w \in L^c : M'(w) = \emptyset \vee M'(w) \subseteq L'^c$: Assume there is a word $w \in L^c$, such that $M'(w) = u \wedge u \in L'$. From the previous part of Lemma it follows, that $u \in M'(L)$. However, then $w = M(u) \subseteq L$, which leads to a contradiction. □

We have seen, that finding the sets $M_\exists^{-1}(L)$ and $M_D^{-1}(L)$ for a given language L and a-transducer M is quite easy using a dual a-transducer M'. However, the situation with $M_\forall^{-1}$ is not that simple. The main problem is, that if some word $w \in \Sigma_{L'}^*$ has an image in L, it can also have other images in L^c, therefore $w \notin L'$. However, if we used a dual a-transducer M' from the previous Lemmas on L, the word w will be constructed, since $w \in M'(L)$. We now present the solution to this issue.

Lemma 4.3. Let $M = (K, \Sigma_1, \Sigma_2, H, q_0, F)$ be an a-transducer and L be a language. Moreover, let $L' = M_\forall^{-1}(L)$. Then, $M(L') \subseteq L$. The mapping $M_\forall^{-1}$ can be simulated by an a-transducer M' dual to the sequential transducer M, such that $L' = M'(L) - M'(L^c)$ 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 ($M(L') \subseteq L$) follows from the definition. If a word w belongs to L', all of its images by M are in L, therefore whole of $M(w) \subseteq L$ and furthermore $M(L') \subseteq L$.

Now we prove the second claim in three steps, using the same construction of the dual a-transducer M' as before.

- $L' \subseteq M'(L) - M'(L^c)$: Let $w \in L'$. By definition, $M(w) \neq \emptyset \wedge M(w) \subseteq L$, therefore, there is a word $u \in M(w) \wedge u \in L$. By the construction of M', it can be easily seen (and proven similarly to the proof of the $M_\exists^{-1}$ case), that $w \in M'(u) \subseteq M'(L)$. Furthermore, if $w \in M'(L^c)$, it means, that there is a word $v \in L^c$, such that $w \in M'(v)$. However, then $v \in M(w)$ and since $v \notin L$, then $M(w) \nsubseteq L$ and by definition $w \notin L'$.

- $M'(L) - M'(L^c) \subseteq L'$: Let $w \in M'(L) - M'(L^c)$. Since $w \in M'(L)$, we know, that there is at least one word u, such that $u \in L \cap M(w)$. The second part, i. e. $w \notin M'(L^c)$ secures, that $L^c \cap M(w) = \emptyset$ (if there was a word $u \in L^c \cap M(w)$, then $w \in M'(u) \subseteq M(L^c)$). Thus, w fulfills the definition of $M_\forall^{-1}(L)$, therefore $w \in L'$.

- $\forall w \in L^c : M'(w) = \emptyset \vee M'(w) \subseteq L'^c$: This claim follows directly from the fact, that $L' \cap M'(L^c) = \emptyset$.

□

We conclude this section by a note, which will later be useful for comparing our three settings to each other. This claim follows directly from the fact, that a sequential transducer is a special case of an a-transducer and the definition of M_D^{-1} fulfills the definitions for both $M_\forall^{-1}$ and $M_\exists^{-1}$.

Remark. Every (effective) T-advice is also a $NT_\forall$-advice. Every (effective) T-advice is a $NT_\exists$-advice.

4.2 Decomposable and Undecomposable Languages

In the previous section, we have defined the notion of an effective advice. Now we present another related concept, namely the T-, $NT_\forall-$ and $NT_\exists$ decomposability of regular languages.

Definition 4.4. The language L is called *T-decomposable*, if there is a sequential transducer M and a regular language L_{adv}, such that (L_{adv}, M) is an effective T-advice for L. Otherwise, we say L is *T-undecomposable*.

Definition 4.5. Let $\Psi \in \{\exists, \forall\}$. The language L is called *NT_Ψ-decomposable*, if there is an a-transducer M and a regular language L_{adv}, such that (L_{adv}, M) is an effective NT_Ψ-advice for L. Otherwise, we say L is *NT_Ψ-undecomposable*.

Lemma 4.4. Every T-decomposable language is $NT_\forall$- and $NT_\exists$-decomposable. Every $NT_\forall$- and $NT_\exists$-undecomposable language is T-undecomposable.

Proof. The proof follows directly from the final remark in the previous section. □

Later we shall see, that the reverse implication does not hold. We now compare our settings to the setting presented by [9] (see Section 2.3). To make the comparison more meaningful, we have to strengthen the condition presented in [9] in the following way:

Definition 4.6. 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 $L[L_1](A) = L$.

Notation. For $\Psi \in \{A, T, NT_\exists, NT_\forall\}$, we denote the class of Ψ-decomposable languages by $\mathscr{D}_\Psi$ and the class of Ψ-undecomposable languages by $\mathscr{U}_\Psi$.

For the sake of clarity, we present just the comparison of A-decomposable and T-decomposable languages, since the relations to $NT_\forall$ and $NT_\exists$ follow similarly.

Theorem 4.5. $\mathscr{D}_A \subseteq \mathscr{D}_T$.

Proof. The proof follows easily by using a sequential transducer computing the identity. □

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

Theorem 4.6. $\mathscr{D}_T \nsubseteq \mathscr{D}_A$.

Proof. Such languages are for example the singleton languages $L_n = \{a^n\}$ 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 $\mathscr{C}_{state}(L_n) = n + 2$.

Now we can use an advice to simplify the accepting automaton as follows: our sequential transducer M will encode each pair of letters a into one new letter b using two states, where the first state is accepting. Formally, $M = (\{q_0, q_1\}, \{a\}, \{b\}, \delta, \sigma, q_0, \{q_0\})$, where

$$\delta(q_0, a) = q_1; \delta(q_1, a) = q_0, \text{ and}$$
$$\sigma(q_0, a) = b; \sigma(q_1, a) = \varepsilon.$$

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

We need to construct an automaton A, such that $L[M_D^{-1}(L_{n,adv})](A) = L_n$. Let $L(A) = \{a\}^*$. Clearly, $M_D^{-1}(L_{n,adv}) = L_n$, so 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 $(L_{n,adv}, M)$ an effective T-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 $n + 2$ 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 $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(L_{adv}) < \mathscr{C}_{state}(L_n) = n + 2$ and $L[L_{adv}](A') = L_n$. 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 $\mathscr{C}_{state}(A') + \mathscr{C}_{state}(L_{adv}) \geq n + n \geq n + 2 = \mathscr{C}_{state}(L_n)$.

Let us now look at the minimal deterministic finite automaton A_{adv} of L_{adv}. Since the inequality holds, A_{adv} has at most n states. Also, A_{adv} accepts the language L_n, that means, in our case, the word a^n. Clearly, by reading a^n, A_{adv} runs in a 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 the form $a^{n+s.l}$, $s \geq 1$.

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 4.6.1. $\mathscr{D}_A \subsetneq \mathscr{D}_T$.

Corollary 4.6.2. There are infinitely many T-decomposable languages.

Corollary 4.6.3. There are infinitely many $NT_\forall$- and $NT_\exists$-decomposable languages.

We have seen, that adding the possibility of transformation in solving problems with supplementary information can help us to also decompose some languages, that are not decomposable without the use of transformation. Further we show, that also adding the possibility to use nondeterminism in the transformation gives us more power (i. e. the settings which use nondeterministic transformation yield bigger classes of decomposable languages).

Theorem 4.7. $\mathscr{D}_T \subsetneq \mathscr{D}_{NT_\exists}$

Proof. We have already seen, that every T-decomposable language is also $NT_\exists$-decomposable. We now show, that the reverse containment is not true.

Each of the languages $L_p = \{(a^p)^*\}$, where p is a prime number, is T-undecomposable. It is easy to see, that $\mathscr{C}_{state}(L_p) = p$. 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 a sequential transducer M. Let $L_{trans} = M_D^{-1}(L_{adv})$.

Now, we present some constraints on the 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 several 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 on a word a^p runs in a cycle of some lengths k, l. Then, both automata would accept the word $a^{p+k.l}$, which however does not belong to L_p (because $k, l < p$ and p is a prime number).

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 $\mathscr{C}_{state}(L_{adv}) < p - 1$.

Moreover, we claim, that L_{simple} contains every word of the form a^k for $k \geq p$. Assume this is not the case and there is a word a^l, such that $a^l \notin L_{simple}$. Since $\mathscr{C}_{state}(A_{simple}) < p$, the sequence of states in the computation of A_{simple} on a^l contains a cycle of length $r < p$. This means, that for every i, the computation of A_{simple} on $a^{l+i.r}$ is not accepting, too. From the group theory we know, that $\mathbb{Z}_p$ is a cyclic group, where every $m \neq 0 \pmod{p}$ is a generator. Since $0 < r < p$, also r is a generator, therefore there is a number s, such that $s.r$ is an inverse element of l in $\mathbb{Z}_p$. This means, that $a^{l+s.r} \in L_p$, but from aforementioned it follows, that A_{simple} does not accept $a^{l+s.r}$, which further means, that $a^{l+s.r} \notin L_{simple} \cap L_{trans}$, which leads to a contradiction.

That means, that in fact, we want to encode the language L_{trans} into the language L_{adv} with a smaller complexity using a sequential transducer M. However, we not only need, that $M(L_{trans}) = L_{adv}$. Lemma 4.2 gives us another supplementary condition on M: $M(L^c_{adv}) \cap L_{trans} = \emptyset$ (this is just another notation of condition from Lemma 4.2).

Now, let us consider a sequential 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. Let us now take the accepting computations on a^p. This computation has a form $(q_0, x_0, y_0, q_{i_1}), ..., (q_{i_j}, x_j, y_j, q_{i_{j+1}}), ...,$ (q_{i_n}, x_n, y_n, q_F), $..., (q_{i_j}, x_j, y_j, q_{i_{j+1}}), ..., (q_{i_n}, x_n, y_n, q_F)$, where $q_F \in F$ and $\forall k : x_k \in \{\epsilon, a\}$. It can be seen, that this computation contains a cycle starting and ending in q_F. Of course, q_F is not necessarily the first state, that occurs in our computations two times, but since M is a sequential transducer (i. e. its transition function is deterministic) and $p > \mathscr{C}_{state}(M)$, if the computations ends in q_F, this state occurs in this computation repeatedly. Let the input and the output of one iteration of this cycle be a^s and u, respectively.

Since the transition and output functions of M are deterministic, the computation on a word a^{p+s} also ends in q_F, because the computation differs only in the number of iterations of our considered cycle. The same holds for every $a^{p+i.s}, i \geq 0$. Moreover, $M(a^{p+i.s}) = M(a).u^i$.

Let us now look at the classes of equivalence relation from Myhill-Nerode theorem (see Section 2.2.1) of L_{trans}. We have already shown, that $L_{simple} \supseteq \{a^k | k \geq p\}$. This leads to a claim, that $\forall k > p, k \neq 0 \pmod{p} : L_{trans} \not\ni a^k$. From this wee see, that the equivalence classes of L_p are also equivalence classes of L_{trans}. We remind, that these classes correspond to individual remainder classes $\mod p$.

We claim, that for every one of these classes, there is a word in this class, such that the computation of this word on M ends in q_F. To prove this, we once again use the same observation from group theory as above - s (the length of the cycle input) is a generator in $\mathbb{Z}_p$. This means, that for $i = 1, 2, ..., p$, every $a^{p+i.s}$ belongs to another equivalence class, but the computation on every of these word ends in q_F. Hence, the claim is proven.

We have shown, that $\mathscr{C}_{state}(L_{adv}) < p$. From Dirichlet's principle it follows, that there are two values $1 < i_1 < i_2 < p$, such that $[M(a^{p+i_1.s})] = [M(a^{p+i_2.s})]$. We once again use the fact, that s is a generator of $\mathbb{Z}_p$. This means, that there are numbers $0 < j_1, j_2 < p$, such that $s.j_1$ and $s.j_2$ are inverse elements to $i_1.s$ and $i_2.s$ in $\mathbb{Z}_p$, respectively. Let us now take the words $a^{p+i_1.s+j_1.s}$ and $a^{p+i_2.s+j_1.s}$. We know, that $a^{p+i_1.s+j_1.s} \in L_{trans}$ and $a^{p+i_2.s+j_1.s} \notin L_{trans}$ (since $p \nmid |j_1 - j_2|.s$); moreover, $M(a^{p+i_1.s+j_1.s}) = M(a^{p+i_1.s}).u_1^j$ and $M(a^{p+i_1.s+j_2.s}) = M(a^{p+i_1.s}).u_2^j$. However, $M(a^{p+i_1.s}).u_1^j \in L_{adv} \Leftrightarrow M(a^{p+i_1.s}).u_2^j \in L_{adv}$ (because $[M(a^{p+i_1.s})] = [M(a^{p+i_2.s})]$). We have found a word, such that $w \in L_{trans} \wedge M(w) \notin L_{adv}$, or $w \notin L_{trans} \wedge M(w) \in L_{adv}$, which contradicts the conditions on sequential transducer M.

To finish the proof of our theorem, we prove the $NT_\exists$-decomposability of L_p for every $p \geq 7$. Let M' be the a-transducer from Figure and $L'_{p;adv} = (a^{\lfloor \frac{p}{2} \rfloor}.b)^*$. What is the language $M'^{-1}_\exists(L'_{p;adv})$ like? We can find this language using an a-transducer M'' dual to M' according to Lemma 14. Clearly, for a word $(a^{\lfloor \frac{p}{2} \rfloor}.b)^k$, M'' outputs two symbols a for every a on the input and one symbol a for every b on the input. This being said, it is easy to see, that $M''((a^{\lfloor \frac{p}{2} \rfloor}.b)^k) = a^k.p$, therefore from $L'_{p;adv}$, M'' generates exactly the language L_p. Therefore, $(L'_{p;adv}, M')$ is an $NT_\exists$-advice with regard to L_p, while the automaton A_{simple}, such that $L[M'^{-1}_\exists(L'_{p;adv})](A_{simple}) = L_p$, needs to accept the language $\{a\}^*$.

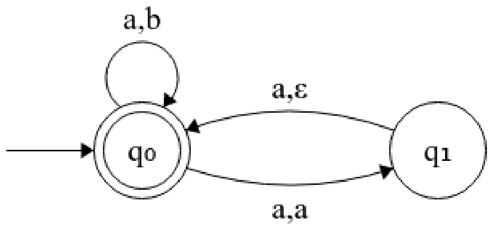

Figure 4.2: a-Transducer M'

Clearly, $\mathscr{C}_{state}(M') = 2$, $\mathscr{C}_{state}(A_{simple}) = 1$ and $\mathscr{C}_{state}(L_{p;adv}) = \lfloor \frac{p}{2} \rfloor + 2$ (we look for the iteration of a string of length $\lfloor \frac{p}{2} \rfloor + 2$ and one additional state is for words with b in incorrect positions). For $p \geq 11$, $\mathscr{C}_{state}(M') + \mathscr{C}_{state}(A_{simple}) + \mathscr{C}_{state}(L_{p;adv}) \leq p$. $\square$

Remark. We could prove the T-undecomposability of the languages $L_p = \{(a^p)^+\}$, where p is a prime number (with Kleene plus instead of the star) in the same way.

Similar result can be obtained for the class of $NT_\forall$-decomposable languages.

Theorem 4.8. $\mathscr{D}_T \subsetneq \mathscr{D}_{NT_\exists}$

Proof. The witness languages for this claim are the complements of L_p's from the previous Theorem. The T-undecomposability of L_p^c can be proven in a similar way to the proof of T-undecomposability of L_p. We show just the first part of the proof, since the rest follows the same pattern as in the aforementioned result.

Assume, that L_p^c is T-decomposable. This means, that we can decompose L_p^c into two languages $L_{p;trans}$ and $L_{p;simple}$, such that $L_{p;trans} \cap L_{p;simple} = L_p^c$. As we see, $L_p^c \subseteq L_{simple}$. However, since the finite automaton for L_p^c has fewer than p states, it is easy to see, that $L_{simple} \supseteq \{a^*\}$ (otherwise at least one of the words $a, aa, ..., a^{p-1}$ would be rejected). It follows, that $L_{trans} \cap \{a^k | k = 0 \pmod{p}\} = \emptyset$. As we have said, the rest of the proof is almost identical to the proof of Theorem 4.7.

It remains to show, that for $p \geq 11$, L_p^c is $NT_\forall$-decomposable. Once again, we use the a-transducer M' from Figure 4.2 and this time, let $L'_{p;adv} = \{a, b\}^* \setminus (a^{\lfloor \frac{p}{2} \rfloor}.b)^*$. We can find the language $M'^{-1}_\forall(L'_{p;adv})$ according to Lemma 16 as $M''(L'_{p;adv}) - M''(L'^c_{p;adv})$ for an a-transducer M'' dual to M'. It can be seen, that $M''(L'_{p;adv}) = \{a\}^*$ (in fact, to show this, it is sufficient to consider only the images of words from b^*). The language $M''(L'^c_{p;adv})$ was shown in the proof of previous Theorem - it is exactly L_p. Therefore, $M''(L'_{p;adv}) - M''(L'^c_{p;adv}) = \{a\}^* \setminus L_p$, which is exactly L_p^c. Moreover $L_{simple} = \{a\}^*$ and for $p \geq 11$, $\mathscr{C}_{state}(M') + \mathscr{C}_{state}(L_{simple}) + \mathscr{C}_{state}(L_{p;adv}) = 2 + 1 + \lfloor \frac{p}{2} \rfloor + 2 \leq p = \mathscr{C}_{state}(L_p)$ and it follows, that L_p^c is $NT_\forall$-decomposable. □

Theorem 4.9. $\mathscr{D}_{NT_\forall} \subsetneq \mathcal{R}$, $\mathscr{D}_{NT_\exists} \subsetneq \mathcal{R}$.

Proof. The definition of Ψ-decomposability for $\Psi \in \{NT_\exists, NT_\exists\}$ contains a requirement, that $L_{dec} = L[M_\Psi^{-1}(L_{adv})](A')$ for some a-transducer M, regular language L_{adv} and finite automaton A'. This condition can be rewritten as $L_{dec} = L(A') \cap M_\Psi^{-1}(L_{adv})$. We have already seen, that $M_\Psi^{-1}(L_{adv})$ can be found with an a-transducer M' dual to M. It is well known, that the class of regular languages is closed under a-transduction, complement and intersection, so the claim, that $\mathscr{D}_{NT_\forall} \subseteq \mathcal{R}$ and $\mathscr{D}_{NT_\exists} \subseteq \mathcal{R}$ follows.

Moreover, the language $\{a\}^*$ is clearly regular, but since $\mathscr{C}_{state}(\{a\}^*) = 1$, we cannot decompose it into three models (a regular language, an a-transducer and a DFA), since each of them has at least one state. Therefore $\{a\}^* \notin \mathscr{D}_{NT_\forall} \cup \mathscr{D}_{NT_\exists}$. $\square$

Previous theorems can be summarized in the following diagram:

$$\begin{tikzcd}
& & \mathscr{D}_{NT_\forall} \arrow[dr, phantom, "\subsetneq" sloped] & \\
\mathscr{D}_A \arrow[r, phantom, "\subsetneq"] & \mathscr{D}_T \arrow[ur, phantom, "\subsetneq" sloped] \arrow[dr, phantom, "\subsetneq" sloped] & & \mathcal{R} \\
& & \mathscr{D}_{NT_\exists} \arrow[ur, phantom, "\subsetneq" sloped] &
\end{tikzcd}$$

As we have seen, the classes of regular languages using T-decomposability differ from 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 questions, that arise, are its closure properties. In this section, we examine the closure of T-decomposable and T-undecomposable languages under some basic operations.

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 the previous section. The second type is the language $L = \{a\}^*$. This language is clearly undecomposable, since $\mathscr{C}_{state}(L) = 1$ and all three devices contained in our indirect advice concept have non-zero number of states.

Theorem 4.10. The class of T-undecomposable languages is not closed under

(a) (non-erasing) homomorphism,

(b) intersection,

(c) union.

Proof.

(a) Consider an undecomposable language $L_a = \{a^{13k} | k \geq 0\}$ and a homomorphism $h : \{a\}^* \to \{a\}^*$, such that $h(a) = aa$. Clearly, $h(L_a) = \{a^{26k} | k \geq 0\}$ and this language can be decomposed in a following way: let us take a sequential transducer M_a computing the identity mapping and a language $L'_a = \{a^{2k} | k \geq 0\}$. These two items form the desired effective T-advice for L_a, since we only have to construct a DFA A, such that $L(A) = \{a^{13k} | k \geq 0\}$, resulting in $L[M^{-1}_{a,D}(L'_a)](A) = L_a$.

Since this homomorphism is non-erasing, our class is not closed even under nonerasing homomorphism.

(b) Consider two languages, $L_{b1} = \{a^{13k} | k \geq 1\}$ and $L_{b2} = \{a^{2k} | k \geq 1\}$. As stated before, both of these languages are T-undecomposable. However, $L_{b1} \cap L_{b2} = \{a^{26k} | k \geq 1\}$ is a T-decomposable language, as we have seen in the first part of this proof.

(c) Let us take two languages, $L_{c1} = \{a^k | k \neq 0 \pmod 5\}$ and $L_{c2} = \{a^k | k \neq 0 \pmod 7\}$. The T-undecomposability of these languages was shown in Theorem 4.8.

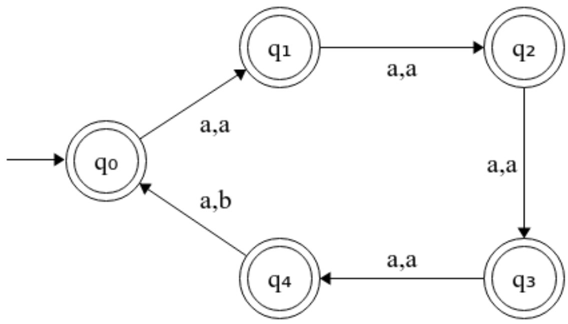

Figure 4.3: Sequential transducer M_c

Now we claim, that the language $L_{c3} = L_{c1} \cup L_{c2} = \{a^k | k \neq 0 \pmod{35}\}$ is T-decomposable. Clearly, $\mathscr{C}_{state}(L_{c3}) = 35$. Take the sequential transducer M_c from Figure 4.2. It can be seen, that the image of a word a^k is a word from $\{a, b\}^*$ with the same length and if and only if $5 \mid k$, the last symbol of this output is b. Now, let $L_{adv} = \{a, b\}^* \setminus (\{a, b\}^{7k-1}.\{b\})$. Clearly, $M^{-1}_{c;D}(L_{adv}) = L_{c3}$. Then, $L_{simple} = \{a\}^*$ and clearly, (L_{adv}, M_c) is an effective T-advice with regard to L_{c3}.

□

4.3.2 T-decomposable languages

Theorem 4.11. The class of T-decomposable languages is not closed under

(a) (non-erasing) homomorphism,

(b) inverse homomorphism,

(c) Kleene star, Kleene plus,

(d) intersection,

(e) union.

Proof.

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

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

(b) Consider the language $L_b = \{a^{26k} | k \geq 1\}$. 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, $h^{-1}(L_b) = \{a^{13k} | k \geq 1\}$, which is T-undecomposable.

(c) The counterexample is given by the language $L_c = \{a^{11}\}$. Let us take the language $L'_c = \{a^5\}$; the sequential transducer $M_c = (\{q_0, q_1, q_2\}, \{a\}, \{a\}, \delta, \sigma, q_0, \{q_1\})$, where

$$\delta(q_0, a) = q_1; \sigma(q_0, a) = \varepsilon$$
$$\delta(q_1, a) = q_2; \sigma(q_1, a) = \varepsilon$$
$$\delta(q_2, a) = q_1; \sigma(q_2, a) = a$$

and the automaton $A_c = (\{q_0\}, \{a\}, \delta, q_0, \{q_0\})$, where $\delta(q_0, a) = q_0$. Clearly, $M^{-1}_{c;D}(L'_3) = L_c$ and $\mathscr{C}_{state}(L'_c) + \mathscr{C}_{state}(T) + \mathscr{C}_{state}(A_c) = 5 + 3 + 1 \leq 9 = \mathscr{C}_{state}(L_c)$, therefore L_c is T-decomposable. Though, $(L_c)^+ = \{a^{11k} | k \geq 1\}$ and $(L_c)^* = \{a^{11k} | k \geq 0\}$ are T-undecomposable.

(d) Let us take a look at two languages, $L_{d1} = \{a\}^* \cup \{b^{15k} | k \geq 1\}$ and $L_{d2} = \{a\}^* \cup \{c^{15k} | k \geq 1\}$. We show the decomposition of L_{d1}, since that of L_{d2} is very similar.

Let $L'_{d1} = \{a\}^* \cup \{b^{3k} | k \geq 1\}$ and let M_{d1} compute the identity mapping. With this advice, we need to check just the language $L''_{d1} = \{a\}^* \cup \{b^{5k} | k \geq 1\}$ with an automaton A''_{d1}. It is easy to see (and provable by Myhill-Nerode theorem), that a DFA for language L_{d1} needs at least 18 states. However, $\mathscr{C}_{state}(L'_{d1}) + \mathscr{C}_{state}(M) + \mathscr{C}_{state}(L''_{d1}) = 6 + 1 + 8 = 15$ and clearly $L[L'_{d1}](A''_{d1}) = L_{d1}$, which means, that L_{d1} is T-decomposable.

However, if we take the language $L_d = L_{d1} \cap L_{d2} = \{a\}^*$, we get a T-undecomposable language, therefore our class is not closed under intersection.

(e) In the previous section we have seen, that the languages $L_{e1} = \{a^{10}\}$ and $L_{e2} = \{a^{12}\}$ are T-decomposable. Now we show, that also their complements are T-decomposable. Let $M_e = (\{q_0, q_1, q_2\}, \{a\}, \{a, b\}, H, q_0, \{q_0, q_2\})$, 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 $M_e(a^{2k}) = b^k$ and $M_e(a^{2k+1}) = b^k a$. Now, the effective advice for L_{e1}^c consists of M_e and $L_{e1,adv} = \{b^5\}^c$. Clearly, $\mathscr{C}_{state}(M_e) + \mathscr{C}_{state}(L_{e1,adv}) + \mathscr{C}_{state}(\{a\}^*) = 3 + 7 + 1 \leq 12 = \mathscr{C}_{state}(L_{e1}^c)$ and $L_{e1}^c = M_e^{-1}(L_{e1,adv}) \cap \{a\}^*$. The effective advice for L_{e2}^c can be constructed in the same way.

However $L_{e1}^c \cup L_{e2}^c = \{a\}^*$ and since $\mathscr{C}_{state}(\{a\}^*) = 1$, this language is T-undecomposable.

□

Conclusion

In our thesis we studied the use of indirect advice in solving problems with supplementary information. We have presented three formalizations of this idea using deterministic finite automata and sequential and a-transducers. We have briefly examined these three frameworks and compared some of the corresponding classes of decomposable and undecomposable languages, i. e., the classes of languages where the corresponding type of indirect advice does not help. We have moreover compared these classes to the previously know class of A-decomposable languages and to the class of regular languages $\mathcal{R}$. Furthermore, we have presented an original result concerning the complexity of a-transducers. In the last Section we have examined closure properties of T-decomposable languages under some basic operations.

There are many possibilities for further research in this area. One of them is to examine further properties of presented classes of languages. Another one is to find the necessary and/or sufficient conditions on T-, $NT_\forall$- and $NT_\exists$-decomposability of regular languages. Moreover, an interesting direction of research would be looking for classes of languages, that can be decomposed with similar advice, or with the use of a fixed type of transformation (e. g., change of the alphabet).

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.