Transduced-Input Finite Automata with Translucent Letters

Madeeha Fatima

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
January 2020
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Benedek Nagy
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Prof. Dr. Gergely Kovács
3. Prof. Dr. Benedek Nagy
4. Prof. Dr. Hasan Taseli
5. Asst. Prof. Dr. Müge Saadetoğlu

ABSTRACT

Finite automata with translucent letters are extensions of the usual finite state automata allowing to proceed the input not strictly left to right manner. There are some letters which are translucent for each internal state such that the automaton cannot read them. These are finite state devices that are able to accept a class of languages that is a superset of the regular languages, moreover, it contains some non-context-free languages. The class is closed under union, concatenation, however, it is not closed under intersection with regular sets. There are three linguistically important non-context-free languages: the multiple agreement, the cross dependencies and the marked copy. These languages cannot be accepted by finite automata with translucent letters.

In this thesis an extension of the model is presented in which the input is preprocessed by a finite state transducer. The transduced input is given to the finite automata with translucent letters, and it decides on acceptance. This is named as T-inputFAwtl i.e. Transduced-Input Finite Automata with Translucent Letters. We prove that all the three mentioned languages are accepted by the deterministic variant of the new model i.e. T-inputDFAwtl. Because of this we say that T-inputFAwtl has more expressive power than original finite automata with translucent letters. We also presented some closure properties of the class of languages accepted by non deterministic variant of this model i.e. T-inputNFAwtl. We proved that the language class accepted by that T-inputNFAwtl is closed under union (if the same transducer is used), and it is closed under intersection with regular languages.

Keywords: t-input automata, automata with translucent letters, Mealy automata, formal languages, finite state machines, transducers, finite state machines, closure properties

ÖZ

Yarı saydam harflere sahip sonlu otomatlar, girişin kesinlikle sağdan sola doğru ilerlememesine izin veren olağan sonlu durum otomatının uzantılarıdır. Her iç durum için yarı saydam olan bazı harfler vardır, böylece otomat onları okuyamaz. Bunlar, normal dillerin bir üst kümesi olan bir dil sınıfını kabul edebilen sonlu durum aygıtlarıdır, ayrıca bazı bağlamsız diller içerir. Sınıf birleşme, yan yana koyma işlemleri altında kapalıdır, ancak düzenli kümelerle kesişme işlemi altında kapalı değildir. Dilsel açıdan önemli üç bağlamsız dil vardır: çoklu anlaşma, çapraz bağımlılıklar ve işaretli kopya. Bu diller yarı saydam harfli sonlu otomatlar tarafından kabul edilemez.

Bu tezde, girdinin sonlu durum transdüseri tarafından önceden işlendiği modelin bir uzantısı sunulmaktadır. Dönüştürülen giriş yarı saydam harfli sonlu otomata verilir ve kabul etmeye karar verir. Buna T-inputFAwtl, yani Translucent Letters ile Transduced-Input Sonlu Otomata denir. Bahsedilen üç dilin hepsinin yeni modelin deterministik değişkeni, yani T-inputDFAwtl tarafından kabul edildiğini kanıtlıyoruz. Bu nedenle T-inputFAwtl'ın yarı saydam harflerle orijinal sonlu otomattan daha etkileyici bir güce sahip olduğunu söylüyoruz. Bu modelin deterministik olmayan değişken tarafından kabul edilen dil sınıfının bazı kapatma özelliklerini yani T-inputNFAwtl de tezde sunduk, T-inputNFAwtl tarafından kabul edilen dil sınıfının birleşim altında (aynı dönüştürücü kullanılıyorsa) ve normal dillerle kesişme işlemleri altında kapalı olduğunu kanıtladık.

Anahtar Kelimeler: t-girdi otomatları, yarı saydam harfli otomatlar, Mealy otomatları, biçimsel diller, sonlu durum makineleri, dönüştürücüler, sonlu durum makineleri, kapalılık özellikleri

This thesis is dedicated: to Almighty Allah, the most Beneficent, the most Merciful, the Omniscent, the All-Knowing, to His Beloved Prophet Mohammad Peace Be Upon Him, my master, and a Mercy for all mankind, to my precious daughter Ayleen Hussain.

ACKNOWLEDGEMENT

First and foremost, I would like to be grateful to Allah SWT for bestowing me with an opportunity to learn, to become who I am today. I know without His mercy and grace, this work would have never become reality.

I owe a deep debt of gratitude and great admiration to my esteemed supervisor Prof. Dr. habil. Benedek Nagy for his immense guidance, incredible patience, and support throughout the research. I am really lucky to be his student and thankful for that I have learnd a lot which wasn’t possible without his dedicated supervision. Also, I am really thankful for my monitoring jury members, Asst. Prof. Dr. Müge Saadetoğlu and Prof. Rashad Aliyev for their valuable suggestions during the past years.

I would like to express my sincerest grartitude and respect to my teacher, my mentor Sir Armaghan Saqib for his motivation, undescribeable moral and psychological support throughout my degree.

I would like to wholeheartedly thank all my beloved family, especially my Father, Mr. Maqsood Ahmed Qureshi who trusted my abilities and encourged me to fly over the unimaginable horizons and my lovely mother Mrs. Zohra Tabbasum for her never ending prayers and unbounded love, my siblings for all their emotional and unconl support, my friends for their kind words and well wishes.

Last but not the least, I owed profound gratitude to my husband, Atif Hussain, for his immeasurable giving, great sacrifice, ultimate love and unquestioning support, to my dearest daughter for simply being there and source of all love and contentment in me.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Automata theory and formal languages are bases of (theoretical) computer science and closely linked to each other. Automata theory is the theoretical study of abstract machines which help in solving real computational problems. Formal languages help to state the behavior of those abstract machines. With time complexity constraint these abstract machines are developed and introduced over time from simpler form to complex forms. They are analyzed and modeled with keeping the fact of limitations of computations. Major families of these models include Finite state machines, Push down automata, Linearly-bounded automata and Turing machines. These machines have some states for performing the transition function on the input which is taken in the form of strings.

With ongoing development in this field, the main purpose of studying automata theory is to develop such computational models which are simple and have more expressive power than the ones which are already developed with about the same simplicity. With continues research, researchers are working to improve models which can help us in solving computational problems in more efficient way. That’s why this subject always has great opportunity for researching new concepts and ideas.

Finite state machines have finite set of states and no additional storage possibilities hence these are the simplest automata. Finite state machines are further categorized into two types i.e. acceptors/recognizers and transducers. Acceptors decides acceptance or rejection on an input, a transducer transduces an input and gives an output. Finite state acceptors or finite automata are used to recognize patterns. They have applied in various fields including designing compilers, spell checkers, text editors, and formal linguistics. In fact, due to the finite number of states and the relatively simple control they are very efficient in terms of time and space complexity. But for the same reason they are inadequate for complex scenarios due to their limited computational power. Finite automata have various extensions which are still finite state models, e.g., multihead finite automata, Watson-Crick finite automata and finite automata with translucent letters.

In this thesis we have focused on this area of automata theory and introduced a new combo automaton using a further extension of finite automata. Our purpose is to achieve a better model for enhanced capability to characterize some class of languages which is beyond regular. Instead of developing a really new model, we combine two relatively simple finite-state machines i.e. Mealy machines and finite state automata with translucent letters, to obtain a still finite state model with a relatively large accepting power.

Consequently, this thesis is divided into four chapters. This chapter 1 gives the overall introduction. In chapter 2 we recall definitions and notations. Chapter 3 gives the definition of the new model and proofs of its enhanced expressiveness and in chapter 4 we have provided important normal form definitions and two closure properties.

This chapter is divided into 2 more sections. Section 1.1 describes about the motivation of this research and brief history of finite automata and its extensions related to our model. Section 1.2 talks about the finite automata and its variants regarding new model.

1.1 A Brief History of Finite Automata

Finite automata are used in text processing, formal linguistics, and hardware design. They are also called finite state acceptors. It is a mathematical model of a finite-state computing device that recognizes a set of words over some alphabet; this set of words is called the language accepted by the automaton. There is a path through the automaton, for each word over the alphabet; if the path ends in a final or accepting state, then that word is in the language accepted by the automaton [20]. There are two classes of finite automata. These are Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata (NFA).

The class of finite automata only accepts the class of regular languages. However, there are various important languages that are not regular and some of them are not even context-free. Various extensions of the finite automata exist in the literature, such as pushdown automata and Turing machines.

In this research another model, the finite automata with translucent letters i.e. FAwtl is considered which is a more powerful model [1] than the finite automata. In each state of these automata some input letters may be translucent and the automaton can read and erase the first visible letter of the input tape, in this sense, the model is closely related to a kind of cooperative distributed systems of restarting automata [2,3]. The accepted language class has some nice properties, e.g., it is closed under the regular operations, union, concatenation and Kleene star and it contains all rational trace languages [2,4]. Relation to linguistics is studied in [5,6]. The language class accepted by FAwtl is a superset of regular languages including some non-context-free languages.

There are three linguistically important non context-free languages, namely the multiple agreement $\{a^n b^n c^n\}$, the cross dependencies $\{a^n b^m c^n d^m\}$ and the marked copy $\{wcw \mid w \in \{a,b\}^*\}$. These three languages are belonging to the classes of mildly context-sensitive languages [7,8]. A brief overview is given later in the chapter 2. Some examples of these languages are discussed in [21]. There are various computational models that are motivated to have generating/accepting power including all these three languages but, on the other hand, having moderate complexity [9,10,11].

Here we study T-inputFAwtl, it is the extension of the computing model finite automata with translucent letters. Mealy machines are finite-state machines transforming the input to an output. In our model, first, the input is transduced and, then, it is given to a deterministic or non-deterministic FAwtl for deciding the acceptance.

1.2 Variants of Finite Automata Related to New Model

In this section we are recalling the informal definition of variants of finite automata. For any detailed concepts one is referred to a standard textbook, e.g., [12,13,14,15].

- A *deterministic finite automaton* (DFA), has a finite set of states and a finite set of input symbols. One state is selected the start state, and zero or more states are accepting states. A transition function determines how the state is changes each time an input symbol is processed [12]. Finite automata are usually represented by their graphs [12,13,14,15].

- A *non-deterministic finite automaton* (NFA), differs from a DFA such that NFA can have any number of transitions (including zero) to next states from a given state by given input symbol [12].
- A *finite state transducer* (FST), a Mealy automaton [16], is finite automaton which is transforming the input to output (and does not accept a language).
- A *non-deterministic finite automaton with translucent letters* (NFAwtl), does not read its input strictly from left to right as in the traditional setting, but for each of its internal states, certain letters are translucent, that is, in this state the NFAwtl cannot see them. Accordingly, it may read (and erase) a letter from the middle or the end of the given input [1,2,3,4,5,6].
- A *deterministic finite automaton with translucent letters* (DFAwtl), a variant of NFAwtl and less expressive than the NFAwtl. [1,2,3,4,5,6].

Chapter 2

NOTATIONS AND DEFINITIONS

In this chapter we will show some formal definitions of the variants of finite automata related to our new combo automata. Also, we will discuss the classes of languages that are called Mildly Context Sensitive; three of its main important languages are accepted by our new model. We recall some earlier definitions and fix our notation. Then in next chapter we will give the definition of our new model.

Finite automata are good computational models with finitely limited memory. They can do many useful things, we interact with such machines all the time e.g. think about an automatic door which sense a person approaching and hold the door open for long enough so one can pass through it [22].

Formal definitions are extremely important to state the ability and limitation of a specific model. Formally, finite automaton has a set of states, an input alphabet, initial state(s), accepting states, and rules for moving i.e. transition function. In every step of the computation, the automaton reads an input letter and determines that what will be the next state. There are two basic variants of finite automata. We write formal definition below. The first variant is Deterministic Finite Automata (DFA) and the other is Non-Deterministic Finite Automata (NFA) [22]. For more examples and details one can go through the books [12, 14, 20, 22].

2.1 Deterministic Finite Automata

Formally,

A *deterministic finite automaton,* a DFA, is a 5-tuple. $A = (Q, \Sigma, q_0, F, \delta)$, where,

Q is the finite set of internal states

Σ is the finite alphabet of input letters

$q_0 \in Q$ is the initial state

$F \subseteq Q$ is the set of final (or accepting) states

δ is the transition function of the form $Q \times \Sigma \rightarrow Q$

It may happen that the transition function is partial, i.e., it is not defined for some of the pairs (q, a) with $q \in Q$, $a \in \Sigma$.

The transition relation can be extended to words $w \in \Sigma^*$ in the usual way. A word w is accepted by A if $\delta\,(q_0, w)$ is defined and it is an accepting state. The set of all accepted words form the accepted language $L(A)$.

2.2 Non-Deterministic Finite Automata

One can follow that in the literature there exist other definitions for NFA. For instance, from [22] we clearly know that DFA and NFA are differentiated in one significant way that is the transition function. In DFA transition function δ takes a state and an input symbol, and produces next state. In NFA transition function δ takes a state and an input symbol *or the empty string*, and produces the set of next possible states. But we use the definition stated below to continue to finite automata with translucent letters and also to our combo model i.e. we have no transition on empty word, but we have a set of initial states.

Formally,

A non-deterministic finite automaton, a NFA, is a 5-tuple $A = (Q, \Sigma, I, F, \delta)$, where,

Q is the finite set of internal states

Σ is the finite alphabet of input letters

$I \subseteq Q$ is the set of initial states

$F \subseteq Q$ is the set of final (or accepting) states

δ is the transition relation of the form $Q \times \Sigma \rightarrow 2^Q$.

If $|I| = 1$ and $|\delta(q, a)| \leq 1$ holds for all $q \in Q$, $a \in \Sigma$, then A is a deterministic finite automata, a DFA. The transition relation can be extended to words $w \in \Sigma^*$ as usual. A word w is accepted by A if $q_f \in \delta(q_0, w)$, where $q_f \in F$, $q_0 \in I$, i.e., $\delta(q_0, w)$ includes an accepting state. The set of all accepted words form the language L(A) accepted by A.

2.3 Mealy Automata

In a Mealy machine, with each transition we get an output [16]. For better understanding the detailed concept of Mealy machine is given here with examples [12]. These are helping in various fields, few of them are detailed in [24, 25, 26].

Formally,

It is a system of $T = (Q, \Sigma, q_0, \Delta, \gamma)$, where,

Q is the finite set of internal states

Σ is the finite alphabet of input letters

$q_0 \in Q$ is the initial state

Δ is the finite set of output symbols

γ is the transition function of the form $Q \times \Sigma$ to $Q \times \Delta$.

Originally T is in its initial state, the input tape contains an input word $w \in \Sigma^*$ and the output tape is empty. When T reads a letter, it writes an output letter to the output tape (concatenating it to the previously written letters if any) and changes its state accordingly. By $T(w) \in \Delta^*$ we denote the output produced by T on input w.

Mealy automata can also be represented by their graphs. You may note that the Mealy machine, we formally defined above is deterministic, and we will use only this, however, this concept can also be generalized to nondeterministic or even further. This is discussed in [23] where details are given.

2.4 Deterministic Finite Automata with Translucent Letters

In the field of Natural languages and trace languages, Deterministic Finite Automata with translucent letters or DFAwtl is more expressive than the regular DFAs. One can see the definition of DFAwtl, other details and proofs of its expressiveness are shown in [1-6].

Formally,

It is a septuple $A = (Q, \Sigma, q_0, F, \$, \tau, \delta)$, where

Q is the finite set of internal states

Σ is the finite alphabet of input letters

$q_0 \in Q$ is the initial state

$F \subseteq Q$ is the set of final (or accepting) states

$\$ \notin \Sigma$ is a special symbol that is used as an end marker of the input

τ is the translucency mapping of the form $Q \rightarrow 2^{\Sigma}$

$\delta : Q \times \Sigma \rightarrow Q$ is the transition relation that satisfies the following condition:

$\forall q \in Q \; \forall \; a \in \tau(q)$: $\delta(q,a) = \emptyset$. For each state $q \in Q$, the letters from the set $\tau(q)$ are translucent for q and therefore there is no transition which reads that letter in the given state.

The automaton A start from the initial state and the whole input w with end marker, i.e., $w\$$ is on the input tape. If A is in state q and its tape content is of the form $uav\$$ such that $u \in (\tau(q))^*$, $a \notin \tau(q)$, $v \in \Sigma^*$, A erases the first occurrence of the non-translucent letter a, obtaining the tape content $uv\$$ and changing the state to $\delta(q,a)$. Whenever, there is no transition is defined on letter a, A could not continue the computation and rejects. Otherwise, if the tape content is $u\$$ such that $u \in (\tau(q))^*$ in a state q, the input w is accepted if $q \in F$ and rejected if $q \notin F$. The set of accepted words w is the accepted language $L(A)$.

A DFAwtl may not process the input strictly from left to right and may accept a word without reading/erasing all of its letters due to the translucency mapping. A DFAwtl can also be given by its graph [1,5,6].

2.5 Non-Deterministic Finite Automata with Translucent Letters

Now we recall the concept of non-deterministic finite automaton with translucent letters (NFAwtl) from [1-6].

Formally,

It is a septuple $A = (Q, \Sigma, I, F, \$, \tau, \delta)$, where,

Q is the finite set of internal states

Σ is the finite alphabet of input letters

$I \subseteq Q$ is the set of initial states

$F \subseteq Q$ is the set of final (or accepting) states

$\$ \notin \Sigma$ is a special symbol that is used as an end marker of the input

τ is the translucency mapping of the form $Q \rightarrow 2^{\Sigma}$

$\delta : Q \times \Sigma \rightarrow 2^{Q}$ is the transition relation that satisfies the following condition:

$\forall q \in Q\ \forall\ a \in \tau(q)$: $\delta(q, a) = \emptyset$. For each state $q \in Q$, the letters from the set $\tau(q)$ are translucent for q.

A is called DFAwtl (deterministic FAwtl), if $|I|=1$ and $|\delta(q, a)| \leq 1$ holds for all $q \in Q$, $a \in \Sigma$.

The automaton A starts the process from the initial state and the whole input w with end marker, i.e., $w\$$ is on the input tape. If A is in a state q and its tape content is of the form $uav\$$ such that $u \in (\tau(q))^*$, $a \notin \tau(q)$, $v \in \Sigma^*$, A erases the first occurrence of the non-translucent letter a, obtaining the tape content $uv\$$ and changing the state to a state in $\delta(q,a)$. Whenever, there is no transition is defined on letter a, A could not continue the computation and rejects.

Otherwise, if the tape content is $u\$$ such that $u \in (\tau(q))^*$ in a state q, the input w is accepted if $q \in F$ and rejected if $q \notin F$. The set of accepted words w is the accepted language L(A).

An NFAwtl may not process the input strictly from left to right and may accept a word without reading/erasing all of its letters due to the translucency mapping. Deterministic and non-deterministic FAwtl can also be given by their graphs [5,6,1].

2.6 Mildly Context Sensitive Langugaes

As a step to simplify and formalize natural languages, Noam Chomsky provided a framework named as Chomsky hierarchy. It has four levels of increasing complexity: regular, context free, context sensitive, and recursively enumerable languages [27]. It does not only help the linguists but also it opened the door for theoretical computer science for grouping the abstract computational models according to the class of their accepted languages. So, they can be descriptive, simplified hence understandable.

Over the time many researcher's work exhibit the refined version of this hierarchy, which was important to adhere the complexity of natural languages. Such a refinement is the classes of mildly context sensitive languages.

The classes of mildly context languages come between context sensitive languages and context free languages under set theoretical inclusion. We can say that they are proper subclasses of the class of context sensitive languages but as they include all context free languages, moreover they inherit some interesting properties of context free languages thus we can also say that these classes are extended classes of the class of context free languages. [27,28, 29].

Natural languages are not context free, and there are three widely known non-context free languages that have patterns in natural languages defined below;

1. The language of cross dependencies; $L_{abcd} = \{a^n b^m c^n d^m \mid m,n \geq 1\}$
2. The marked copy language; $L_{ww} = \{wcw \mid w \in \{a,b\}^*\}$
3. The language of multiple agreements; $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$

Mildly context sensitive classes contain these specific languages [7]. [28] states properties of the languages of these classes. For example, they have polynomial parsing complexity among other properties.

Chapter 3

TRANSDUCED-INPUT FINITE AUTOMATA WITH TRANSLUCENT LETTERS

Our models Transduced-input Finite Automata with translucent letter or T-inputFAwtl is motivated by [17,18] where various types of pushdown automata had such a preprocessed input.

Thus, in our combo model the input is preprocessed by a finite state transducer. The transduced input is given to the finite automata with translucent letter, and it decides on acceptance.

Following is a formal definition of our new combo automata.

Definition 1. Let A be FAwtl and $T = (Q, \Sigma, \Delta, q_0, \gamma)$ be a Mealy machine such that the output alphabet Δ of T is the same as the input alphabet of A. Then, the pair (T, A) is called a *transduced-input finite automaton with translucent letters*, *T-inputFAwtl* for short. The language accepted by (T, A) is defined as

$L(T, A) = \{w \in \Sigma^* \mid T(w) \in L(A)\}$.

The FAwtl and Mealy automaton, both can be deterministic or non-deterministic. Based on their determinism we can define four variants of T-inputFAwtl;

- DDT-inputFAwtl; The Mealy automaton and FAwtl both are deterministic.
- DNT-inputFAwtl; The Mealy automaton is deterministic and FAwtl is non-deterministic.
- NDT-inputFAwtl; The Mealy automaton is non-deterministic and FAwtl is deterministic.
- NNT-inputFAwtl; The Mealy automaton and FAwtl both are deterministic.

All these variants are further grouped accordingly if the combo automaton use same transducer or not.

Below is a simple example to understand the behavior of the automata T-inputNFAwtl. In the next section it is shown that combo automata accept three important non-context-free languages where both the Mealy machine *T* and the FAwtl *A* are deterministic.

Example 1. The language $L_c = \{c^n w c^n w c^n \mid w \in \{a,b\}^+ \text{ and } n>0\}$ is accepted by the T-inputNFAwtl presented graphically in Figure 1. The figure shows the graphical representation of the transducer *T* (up) and the NFAwtl *A* (bottom). The Mealy automaton *T* has the following roles:

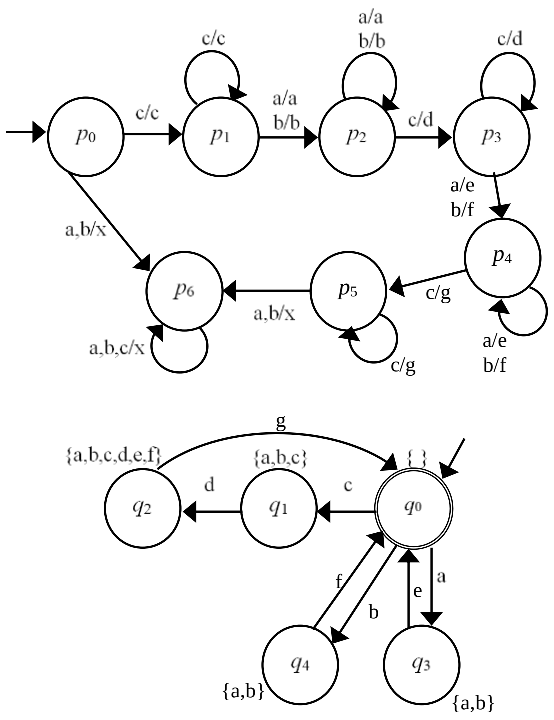


Figure 1. The T-inputDFAwtl that accepts the language L_c.

- It checks if the input is of the form $c^+(a+b)^+c^+(a+b)^+c^+$; particularly if there are three factors of letter *c* in the input word, moreover, they are separated by non-empty words over $\{a,b\}$. Whenever the form of the input does not match, *T* puts at least one *x* to the output tape noticing this fact.

- *T* rewrites the second and third blocks of *c*'s to *d*'s and *g*'s, respectively.

- It also rewrites the second block over $\{a,b\}$ by the alphabetic morphism $h(a) = e$, $h(b) = f$, in this way the original letters *a* and *b* are mapped to *e* and *f*, respectively, on the output tape of *T* in this block.

At the NFAwtl *A*, the set of translucent letters is shown at each state. Observe that in fact, *A* is also deterministic, it is a DFAwtl. It works as follows. In its initial state, which is also the only final state, there is no translucent letter, thus it must read the first letter of the word T passes to it. By the transitions of its first three states, it erases a letter c, a letter d and a letter g, thus in this cycle it checks if the number of c's and d's and g's are the same. If in the original input the format was appropriate, and the number of the c's in each of the three blocks were the same, then and only then, all c's, d's and g's are erased by A. Otherwise, either it gets stuck (if there were more c's in the first block than in any of the other blocks) or some d's and/or g's are left (if the first block of c's was shorter than the other blocks). In the second phase of the computation states q_0, q_3 and q_4 are used (in an accepting run). A erases the first letter of the remaining input, and depending on if it is an a or a b, q_3 or q_4 is reached. From this state the original letters a and b are both translucent, and the first letter of the other block, an e or an f is read such that it must fits to the previously read original letter. Observe that A cannot read any letter x. Therefore, it follows that (T, A) accepts the language L_c.

For instance, the input word *cabcabc* is in the language L_c. T preprocesses it as follows. The preprocessing starts at state p_0. The first c is kept in the transduced input as c, and state p_1 is reached. Then the first a is kept in the transduced input as a, then b has also been kept, and state p_2 is reached. After that c is rewritten to d, and state p_3 is reached. Here, the next letter, a, is rewritten to e, similarly b is transformed to f, and state p_4 is reached. Then c is transduced to g, and p_5 is reached. Thus, the word *cabdefg* is obtained and passed to A with the end marker \$. In its initial state q_0 nothing is translucent, therefore first letter c is read and state q_1 is reached with remaining input *abdefg*\$. Here a, b and c are translucent, thus A reads d (which is the image of the second c) by changing its state to q_2 with remaining input *abefg*\$. Here again a, b, c, and also *d, e, f* are translucent, therefore *A* reads *g* (which in fact refers for the third block of *c*'s) by changing its state back to q_0 with remaining input *abef*\$. Here nothing is translucent, and now the first letter is *a*. *A* reads it and state q_3 is reached with remaining input *bef*\$. Here *a* and *b* are translucent, thus, *e* is read from the remaining input and *A* moves into the state q_0 with remaining input *bf*\$. Here, there is no translucent letter, *b* is read and state q_4 is reached. Here *a* and *b* are translucent, the last letter *f* is read and *A* moves into its accepting state q_0 with a fully processed input. Thus, the string *cabcabc* is accepted by (*T*, *A*).

On the other hand, for example the input word *abcabc* is preprocessed by the Mealy automaton *T* to *xxxxxx*. It is clearly not accepted by *A*. The word *abcabc* is not in the language L_c. Observe that *A* cannot read any letter *x* because no transition is defined with letter *x*. It is used as a kind of failure symbol.

3.1 Accepting Mildly Context-Sensitive Languages

In this section we will show that all the three important mildly context-sensitive languages are accepted by our deterministic finite state model, i.e., we present T-inputDFAwtl for each of them.

Let us start with the language of multiple agreements.

3.1.1 The Language of Multiple Agreements

Theorem 1. The language $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$ is accepted by a T-inputDFAwtl.

Proof: We present an FST T_1 and a DFAwtl A_1 such that the pair (T_1, A_1) accepts L_{abc}. $T_1 = (\{p_0, p_1, p_2, p_3\},\{a,b,c\},\{a,b,c,x\}, p_0,\gamma)$ with $\gamma(p_0,a) = (p_0,a)$, $\gamma(p_0,b) = (p_1,b)$, $\gamma(p_0,c) = (p_3,x)$, $\gamma(p_1,a) = (p_3,x)$, $\gamma(p_1,b) = (p_1,b)$, $\gamma(p_1,c) = (p_2,c)$, $\gamma(p_2,a) = \gamma(p_2,b) =$

(p_3,x), $\gamma(p_2,c) = (p_2,c)$ and $\gamma(p_3,a) = \gamma(p_3,b) = \gamma(p_3,c) = (p_3,x)$. Further, let $A_1 = (\{q_0,q_1,q_2\},\{a,b,c,x\},\$,\tau,q_0, \{q_0\},\delta)$ with $\tau(q_0)=\{\}$, $\tau(q_1)=\{a,c\}$, $\tau(q_2)=\{a,b\}$ and $\delta(q_0,a) = q_1$, $\delta(q_1,b) = q_2$, and finally, $\delta(q_2,c) = q_0$.

Their graphical representations are shown in Figure 2. In fact, if the input is of the form $a^*b^+c^+$, T_1 is copying it, otherwise at least one x will appear on its output. Then, A_1 accepts only words that contain the same number of a's, b's and c's and do not contain any x (observe that no transition is defined with letter x). Hence exactly the following words are accepted: the empty word is accepted, if the input is nonempty, then it must start with a, after some a's, the same number of b's and finally the same number of c's must follow, e.g., *abc*, *aabbcc*, *aaaabbbbcccc*.

T_1: A_1:

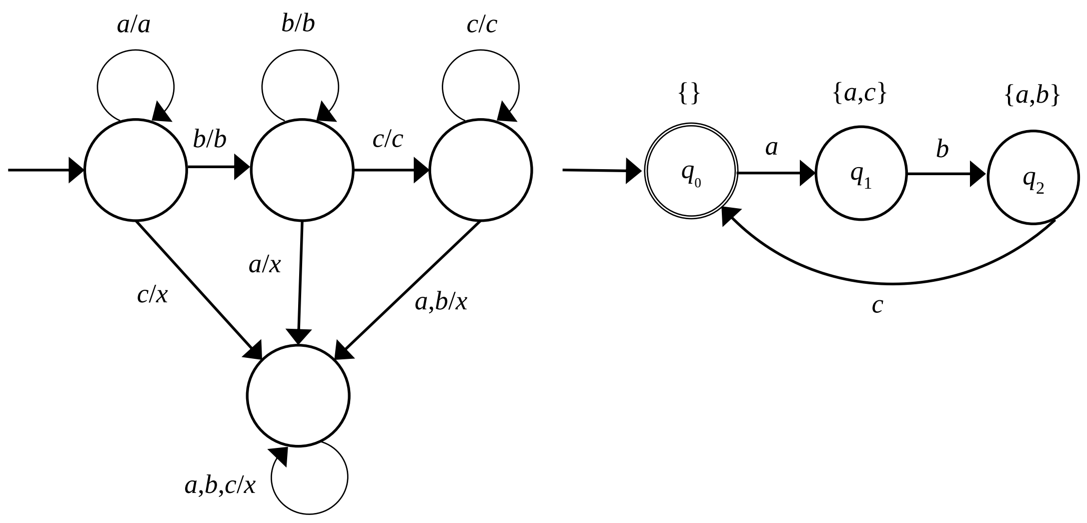


Figure 2. The T-inputDFAwtl that accepts the language L_{abc}.

Let us consider the word *aabbcc* which is in the language. T_1 starts with initial state p_0, it reads a, transduce it to a, then again reads an a and transduces in a and stay at same state p_0, then reads a b, and transduce it to b and reach to state p_1, then again read a b, transduce it to b by remaining in same state. Then T_1 reaches to state p_2 and by transducing first *c* to *c*, and transduce second *c* to *c* by remaining it in same state. The output is *aabbcc* which is passed to A_1. At its initial state q_0 nothing is translucent, A_1 reads first letter *a* of the input *aabbcc*\$ and move to state q_1 with remaining input *abbcc*\$. At this state *a* and *c* are translucent, it will skip the second *b* and will read first *b* and move to state q_2. Now *a* and *b* are translucent, A_1 reads first *c* and go to state q_0 with the remaining input *abc*\$. At q_0 we can see that nothing is translucent, A_1 reads first letter that is *a* and go to q_1 with remaining input *bc*\$ where *a* and *c* are translucent. Here, we read *b* and reach to q_2 with remaining input *c*\$. Finally, we read the last letter *c* and reach to accepting state q_0 with \$. It means that the word is accepted by T-inputDFAwtl (T_1,A_1).

The word *acb* is not accepted by (T_1,A_1) since T_1 transform is to *axx* and A_1 gets stuck on this.

The word *aabcc* is transformed to itself by T_1, and then, A_1 processes a cycle by erasing an *a*, a *b* and a *c*, such that it is state q_0 with remaining input *ac*. Now *a* is read and erased, *c* is left, however, in q_1 *c* is translucent, A_1 reaches the end marker, but the state q_1 is not accepting, thus this input is rejected.

3.1.2 The Language of Cross Dependencies

Theorem 2. The language $L_{abcd} = \{a^n b^m c^n d^m \mid m,n \geq 1\}$ is accepted by a T-inputDFAwtl.

Proof: We give a T-inputDFAwtl (T_2, A_2) that accepts L_{abcd} in Figure 3. First, the Mealy automaton T_2 preprocesses the input by changing some letters to *x* if the input is not in the form of $a^+b^+c^+d^+$. Then, the DFAwtl A_2 is checking if the number of *a*'s and c's and the number of b's and d's match by accepting exactly those which are in L_{abcd} after the preprocessing phase.

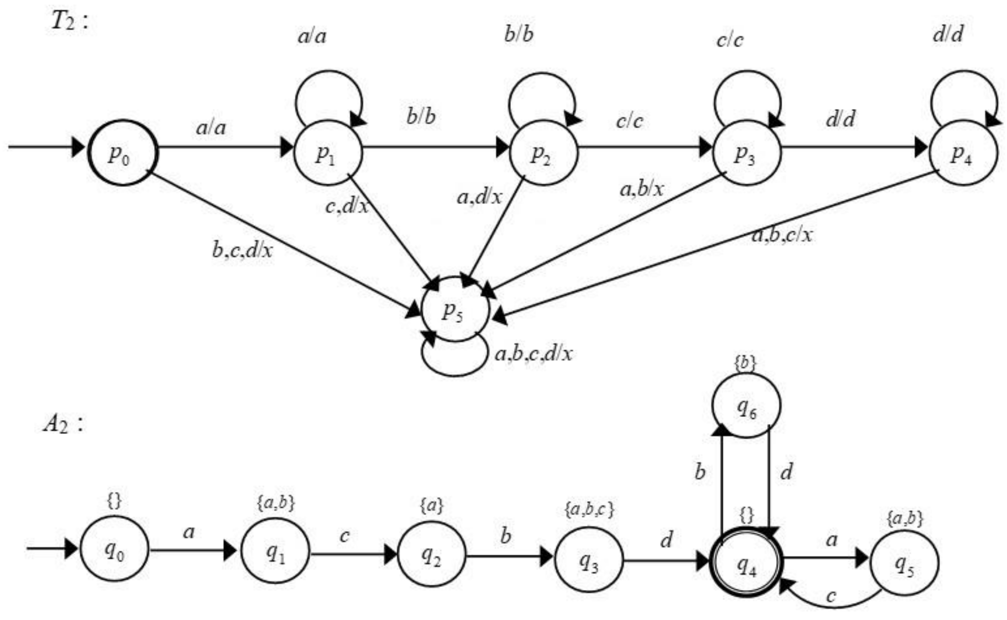


Figure 3. The T-inputDFAwtl that accepts the language L_{abcd}.

For instance, *aabccd* is in the language and T_2 passes it unchanged to A_2. In the initial state q_0 nothing is translucent, the automaton reads a, hence the input is shortened to *abccd* and the automaton changes its state to q_1. Here a and b are translucent, it will read the first c letting the remaining input *abcd* and then it moves to state q_2. Now a is translucent and b is read, *acd* left, state q_3 is approached. Letters a, b, c are translucent, letter d is read, thus at q_4 the input is reduced to *ac*. No translucent letter in this state, acceptance goes only with fully processed input in this automaton, thus a is read, q_5 is reached. The last letter c is read and the accepting state q_4 is reached. The word *aabccd* is accepted.

Now consider the word *abcdd*. T_2 transforms the word to itself *abcdd* and passes it to A_2. At the initial state nothing is translucent and A_2 reads first a and go to state q_1 with remaining input *bcdd*\$. At this state *a* and *b* are translucent. A_2 reads *c* and move to q_2 with remaining input word *bdd*\$. Now we read *b* and move to state q_3. The remaining input word is *dd*\$. We read a *d* and reach to q_4. One can see that it's an accepting state but we still didn't reach to end marker \$ as the remaining input word still have *d*\$ and there is no transition to read this letter. Hence the input word is rejected.

The word *abccdabd* is not in the language. T_2 transforms *abccdabd* to *abccdxxx*. A_2 clearly has no transition for letter *x*, hence it is also not accepted by A_2.

Finally, after, the language of cross dependencies, the copy language is considered.

3.1.3 The Marked Copy Language

Theorem 3. The language $L_{ww} = \{wcw \mid w \in \{a,b\}^*\}$ is accepted by a T-inputDFAwtl.

Proof: Figure 4 shows the graphical representation of the combo T-inputDFAwtl (T_3,A_3). The Mealy automaton T_3 has two roles: it checks if there is at most one letter *c* in the input word, moreover, it rewrites the suffix of the input, after the (first) *c* in such a way, that original letters *a* and *b* are mapped to *d* and *e*, respectively. A_3 checks and erases the first letter of the preprocessed input, and depending on if it is an *a* or a *b* it erases the first letter after the *c* if it matches to the checked letter. Finally, when the word starts with a *c*, it is processed, and if the preprocessed input is fully erased by this time, it is accepted. Observe that A_3 cannot read any letter *x*. It follows that (T_3,A_3) accepts the language L_{ww}.

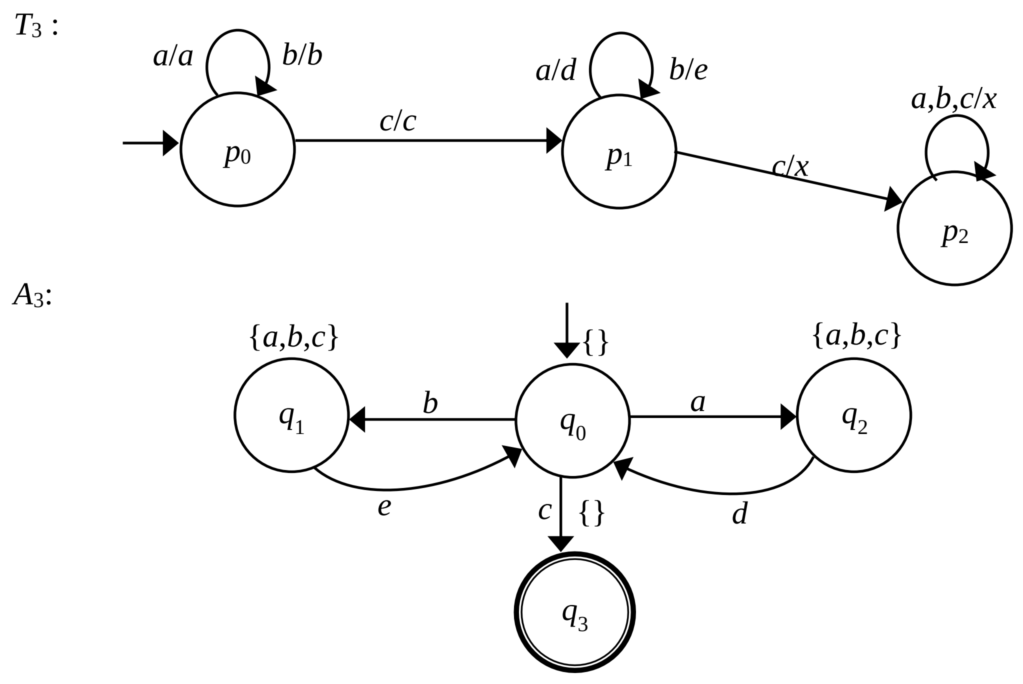


Figure 4. The T-inputDFAwtl that accepts the language L_{ww}.

For instance, *abcab* is in the language L_{ww} and T_3 preprocesses it as follows. The preprocessing starts at state p_0. The first *a* is kept in the transduced input as *a*, then *b* has also been kept, after that *c* is kept, and state p_1 is approached. Here, *a* is rewritten as *d* and, similarly *b* is translated to *e*. Thus, the word *abcde* is obtained and passed to A_3. In its initial state q_0 nothing is translucent, therefore *a* is read and state q_2 is approached with remaining input *bcde*. Here *a*,*b* and *c* are translucent, A_3 reads *d* by changing its state back to q_0 with remaining input *bce*. Here nothing is translucent, *b* is read and state q_1 is approached. Here *a*,*b*,*c* are translucent, *e* is read and A_3 moves into state q_0 with remaining input *c*. Here there is no translucent letter, *c* is read and A_3 reach its accepting state q_3 with a fully processed input. Thus, the string *abcde* is accepted by (T_3,A_3).

The word *bacba* is in the language. T_3 at its initial state p_0 transduce the letters *b, a,* to *b, a,* and stays in the same state. Then it transduces the *c* to *c* and reached to state p_1.

At this state the letters *b, a,* are transduced to *e, d,* and the state is not changed. We get an output word *baced* and passes it to A_3. Initially nothing is translucent it reads first letter *b* and move to state q_1 by shortening the word to *aced*. q_1 has translucent letters *a, b, c*. A_3 will skip those letters in the input word and will read an *e*, and will to move to state q_0. The remaining input word is now *acd*\$. Again, nothing is translucent at this stage, A_3 will read first letter that is *a* and will go to state q_2 with input *cd*\$. Here again letter *a*, *b*, and *c* are translucent. A_3 will read now *d* by the shortening the input word to *c*. A_3 is now in its initial state and the remaining input is *c*\$. It will read *c* and will reach to endmarker \$ with completely processed input which shows that this is accepted by (T_3, A_3).

On the other hand, for example the input word *abcabc* is preprocessed by the Mealy automaton T_3 to *abcdex* which is not in the language L_{ww} and it is clearly not accepted by (T_3, A_3).

Chapter 4

CLOSURE PROPERTIES

In this section we present few closure properties of the language class accepted by DNT-inputFAwtl. First, the regular operation union is studied, we show that the language class accepted by T-inputNFAwtl is closed under union if the same transducer is used independently if the transducer is deterministic or non deterministic.

4.1 DNT-inputFAwtl Closure under Union Using Same Transducer

Theorem 4. Let (T, A_1) and (T, A_2) be two T-inputFAwtl. T is detreministic transducer and A_1, A_2 are NFAwtl. The union of the languages accepted by (T, A_1) and (T, A_2) is also accepted by a transduced-input non-deterministic finite automaton with translucent letters with transducer T.

Proof: Given the T-inputNFAwtl (T, A_1) and (T, A_2), where $T = (Q, \Sigma, \Delta, q_0, \gamma)$ is a Mealy machine and $A_1 = (Q_1, \Delta, \$, \tau_1, I_1, F_1, \delta_1)$, $A_2 = (Q_2, \Delta, \$, \tau_2, I_2, F_2, \delta_2)$ are two NFAwtl, we will construct the combined automaton (T, B), where B is an NFAwtl such that $L(T, A_1) \cup L(T, A_2) = L\ (T, B)$.

Without loss of generality, we may assume that $Q_1 \cap Q_2 = \emptyset$.

Then, let $B = (Q_1 \cup Q_2, \Delta, \$, \tau, I_1 \cup I_2, F_1 \cup F_2, \delta)$, where

$$\delta(q) = \begin{cases} \delta_1(q) & \text{if } q \in Q_1 \\ \delta_2(q) & \text{if } q \in Q_2 \end{cases}$$

$$\tau(q) = \begin{cases} \tau_1(q) & \text{if } q \in Q_1 \\ \tau_2(q) & \text{if } q \in Q_2 \end{cases}$$

Since there is no interference between the computations done by A_1 and A_2 encoded in B, each of the accepting (and non-accepting) computations of A_1 and A_2 has a one-to-one correspondence with an accepting (non-accepting) computation of B, respectively. Thus, $L(B) = L(A_1) \cup L(A_2)$, and therefore, $L(T, A_1) \cup L(T, A_2) = L(T, B)$.

The proof is finished. This is explained also with example 4 in Section 4.4

4.2 Dual Normal Form

To present another closure property result in section 4.3 first we need to recall that all NFAwtl can be converted into “last letter normal form”. In [2] (Theorem 6.5) it is proven that every NFAwtl A has an equivalent NFAwtl A’ accepting the same language with special properties.

Definition 2. An NFAwtl $A' = (Q_{A'}, \Sigma, \$, \tau_{A'}, I_{A'}, F_{A'}, \delta_{A'})$ is in “last letter normal form” if the following conditions hold;

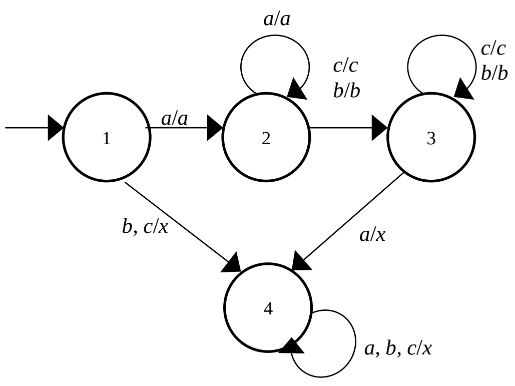
1. In each state $q \in Q_{A'}$, there is exactly 1 letter for which transitions are allowed.
2. The last occurrence of each letter $a \in \Sigma$ of the input word is erased in a transition (from a state) such that the translucency mapping is empty at that state.
3. Every input letter is processed in an accepting computation i.e. the translucency mapping assigns an empty set to any final state.
4. The automaton has exactly 1 accepting state $q_f \in F_{A'}$.

A kind of extension of the normal form is helpful to prove the closure property we will show in the next section. Now let us consider an example.

Example 2: The Language $L_{aw=}\{a^n w \mid w \in \{b,c\}^*$ and $|b|=|c|=n \geq 0\}$ is accepted by the T-inputDFAwtl.

Figure 5 shows the graphical representation of the combo automata T-inputDFAwtl (T, A). T preprocesses input in form of $a^*\,w$. Otherwise we will have an x in input word. We can clearly see from A that there is no transition for letter x. Thus, all those words which contains x will be clearly not accepted by A.

T:



A:

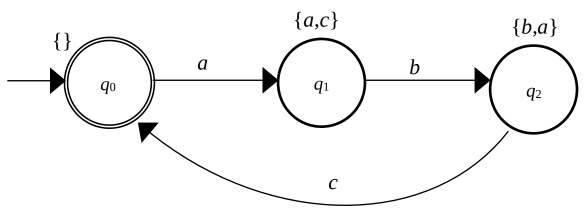


Figure 5. The T-inputNFAwtl for language L_{aw}.

Now A is already fulfilling condition 1, 3, and 4 from Definition 2 stated above. It means first, there is only 1 letter for which transition is allowed for each state. A has exactly one accepting state q_0 and it is processing every input letter in an accepting computation. Now, to fulfill the remaining condition 2, we need to modify DFAwtl A into "last letter normal form" A' such that when A' reads last letter then there is no translucency mapping.

To incorporate this important point in, we will index all the states $Q_{A'}$ by the set of n-tuples from the set IND= $\{(i_1,\ldots, i_n)|\ i_1,\ldots, i_n \in \{2,1,0,d\}\}$.

With a word $w \in \Sigma^*$ we associate an index vector IND(w)= $(i_1,\ldots, i_n)$ by taking

$$i_j = \begin{cases} 2 \text{ if } |w|_{a_j} \geq 2 \text{ i.e. } w \text{ contains atleast two occurences of letter } a_j \\ 1 \text{ if } |w|_{a_j} = 1 \text{ i.e. } w \text{ contains exactly one occurrence of letter } a_j \\ 0 \text{ if } |w|_{a_j} = 0 \text{ i.e. letter } a_j \text{doesn'toccur in word } w \end{cases}$$

for all $j = 1,\ldots,n$. As a first general rule A' cannot erase a letter a_s if $i_s = 0$ or $i_s =$ d holds.

On input $w \in \Sigma^*$, A' guesses a tuple IND' (w)= $(i_1,\ldots, i_n) \in \{2,1,0\}^n$ for one of the initial state of automata $q_0 \in Q_{A'}$. It attempts to erase the left most occurrence of letter a_s, changing the state to a state q_i in $\delta_{A'}(q, a)$ for all i=1,…,n. $\delta_{A'}$ is a transition relation for $A'$where $q \in Q_{A'}$. For instance, L_{aw} is a language over three letters a, b, and c. For initial state q_0 in Figure 5, it will associate an index 222 with an input word w which corresponds to the at least two occurrences of a, b, c in that state when the computation starts. A will read the leftmost occurrence of letter a_s such that $i_s \neq 0$ and will move to next state according to transition relation. Moreover, if i_s =2, then A' transforms the word $w = w_1\, a_s\, w_2$ into the word $w_1 w_2$. Now, IND($w_1 w_2$) either coincides with IND(w) or it is obtained from IND(w) by replacing i_s by the value $i'_s = 1$. For, instance in Figure 6 after reading a from state q_0^{222}, if there are at least 2 a's left in input word than this state will correspond to q_1^{222} otherwise it will correspond to q_1^{122} by replacing the index of a from 2 to 1. If i_s =1, then A' transforms the word $w = w_1\, a_s\, w_2$ into the word $w_1 w_2$. Here $|w_1 w_2|_{a_s} = 0$. For, instance in Figure 6 after reading a from state q_0^{111}, it will move to next state q_1^{011} by replacing index from 1 to 0.

After reading the last occurrence of a letter a_s, automaton moves to the next state - during a computation of certain input word w, the remaining input word now may have a leftmost last occurrence of letter b_s which we cannot read because of translucency mapping in original automata A. But in the "last letter normal form", we have no translucency at the said step and we read that letter on that state by using indicator d in index. Indicator "d" means that the current word w contains a single occurrence of the letter b_s, but it is already read in the corresponding to original computation.

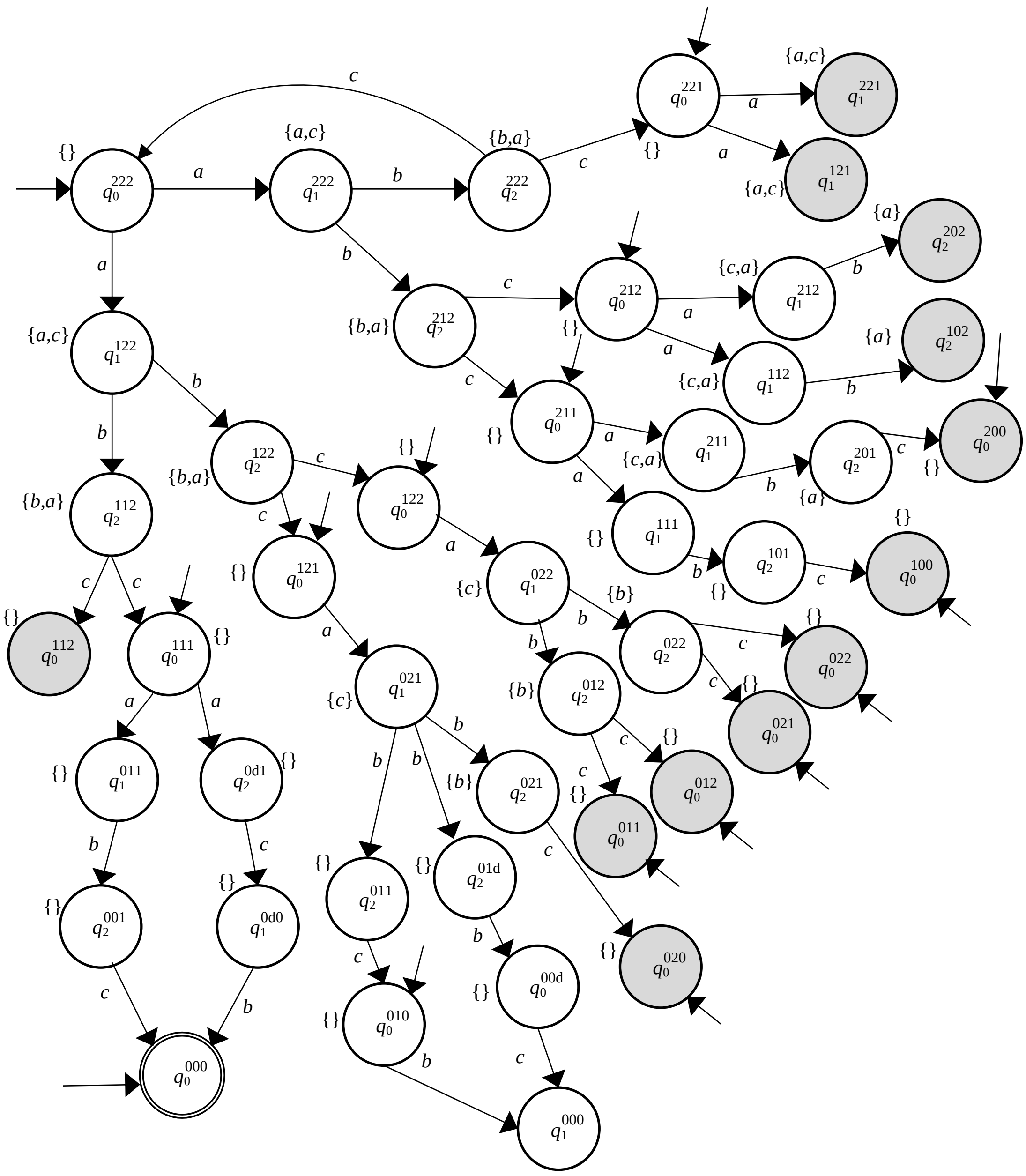


Figure 6. Partial representation of the "last letter normal form" NFAwtl for the language L_{aw}.

Let us consider the word *acb*. In original automata (Figure 5) at initial state q_0, DFAwtl A will read the first letter a and then move to next state q_1 with remaining word *cb*\$. At q_1, c is translucent and A will read b and reach to state q_2 with remaining input c\$. At state q_2, A will read last c.

Now, consider this word *acb* computation on NFAwtl in Figure 6 which is “last letter normal form” of A. At initial state q_0^{111} the automaton will read the first letter a, and goes to next state q_2^{0d1} where nothing is translucent and automaton has to read c. This specific step of computation is not possible in original automaton A because c becomes translucent after reading a. But in its last letter normal form, to read c, automaton will reach to state q_2^{0d1} after reading a and considering that letter b is already read by replacing its index 1 with d. Later in that computation d is replaced with 0 to show that the input is completely processed.

Please note that there is only one accepting state q_0^{000}. All other states with empty input word end in non-accepting computations e.g. q_1^{000}.

Figure 6 is a partial graphical representation of “last letter normal form”. All the paths which ends in a state with darker base color, it means that the specific path will end in a failed computation. Only one such complete path (which will end in failed computation) is shown to understand the automaton’s expected process in this case. For example, the path for word *aabbbcc* is from q_0^{222} to q_1^{000} but this word is not in language L_{aw} and q_1^{000} is not an accepting state. Hence, this ends in rejecting the word *aabbbcc.*

Now we further modify A’ in such a way that it also fulfills some additional properties which are very important to prove next property, closure under intersection with regular languages. We will call this form dual normal form.

Definition 3. An NFAwtl $A'' = (Q, \Sigma, \$, \tau, I, F, \delta)$ is in dual normal form if it fulfills the following conditions.

1. There is an NFAwtl $A= (Q_A, \Sigma_A, \$, \tau_A, I_A, F_A, \delta_A)$ in “last letter normal form” such that $L(A)= L(A'')$, moreover

2. The alphabet Σ of A is doubled. Formally, $\Sigma = \Sigma_A \cup \Sigma'_A$ where $\Sigma'_A = \{b' \mid b \in \Sigma_A\}$ and $\Sigma_A \cap \Sigma'_A = \emptyset$
3. The states of A are doubled, there is a state for transitions for an original input letter and there is also a copy with transitions of its marked version. Formally,

   $Q = Q_A \cup Q'_A$,where $Q'_A = \{q' \mid q \in Q_A\}$ and $Q_A \cap Q'_A=\emptyset$.

   This condition is not applicable on the accepting (final) state, as we will always have only one accpeting state (Condition 4 in Definition 2).
4. The translucency mapping for each state contains both the original and the marked version of the given letters. Formally,

   $a, a' \in \tau(q)$ if $a \in \tau_A(q)$

   $a, a' \in \tau(q')$ if $a \in \tau_A(q)$
5. The last input letter a' for every input word w will only belong to Σ'_A. Therefore, no duplication is allowed for the states through which we read last letter.

That is, in the dual normal form, in the NFAwtl, all accepting computations erase the entire input word and one can also be sure when the last letter of the input is processed.

Figure 9 shows the modified version of NFAwtl from Example 2 which is now in dual normal form NFAwtl. We have marked the copied states and input letters with a dash (‘) to distinguish the copy from the original states and input letters of *A.* Other points we need to keep in mind are that every marked letter as well as original letter must have a similar or analogous computation from one state to another reading an original letter and its marked version. For instance, if there is a transition from a state *p* to *q* reading letter *a*, then there must be a transition from *p* to *q*’ reading letter *a*, and from *p’* to *q*’ or *p*’ to *q* reading its marked version *a*’.

Now, all the states of *Q* will be indexed by n-tuple and *Q* = *Q*×IND as we did in “last letter normal form” stated above.

In Figure 9 those paths are removed which end in failed computation. Consider the path from q_0^{222} to q_1^{000} in Figure 6 which we removed in the dual normal form for the said reason.

Now we turn to another interesting closure property, namely we study intersection by regular languages.

4.3 DNT-inputFAwtl Closure under Intersection with Regular Languages

Theorem 5. The language class accepted by DNT-inputFAwtl is closed under intersection with regular languages.

Proof: Let *T* be a deterministic transducer, *A* is an NFAwtl, and *B* is a DFA. We need to consturct DNT-inputFAwtl (*T’*, *A*’) which accepts the intersection of the languages accepted by the NFAwtl (*T*,*A*) and by *B*. First, the “intersection” of *T* and *B* is constructed, i.e., the transducer *T*' and then, its output is forwarded to *A*' which is based on the dual normal form of the NFAwtl A.

The intersection of the two finite state automata *T* and *B* is done by the usual cross-product method, however, here one of the automaton is an accepting device while the other, and as well as the resulted automata, are transducers. In what follows, the accepting states of *B* must be encoded in the output allowing the NFAwtl *A*' to check also this condition. Thus, the output alphabet of *T* is doubled, and whenever, *B* is in accepting state (which is clearly identified since *B* is a *DFA*) in its process, a marked output letter (such as *a*') is written in the output tape (instead the original output letter *a*) allowing to distinguish the positions where the prefix of the input is also in the regular language defined by *B* or not.

However, since NFAwtl may proceed the input in a not usual left-to-right way, we need to be careful how to know that the input is in both of the languages of (*T*,*A*) and of *B*. This point is satisfied with condition 2 definded in Definition 2 Section 4.2. In this way the condition to be in the language *L*(*B*) can also be checked by *A*'. To ensure all these points, we built dual normal form *A*'.

Formally, let

$T = (Q_T, \Sigma, \Delta_T, q_0, \gamma_T)$

$A= (Q_A, \Delta_T, \$, \tau_A, I_A, F_A, \delta_A)$

$B= (Q_B, \Sigma, q_i, F_B, \delta_B)$ be given.

Further, let

$T' = (Q', \Sigma, \Delta, (q_0, q_i), \gamma)$ where,

$Q' = Q_T \times Q_B$

$\Delta = \Delta_T \cup \Delta'$ where $\Delta' = \{b' \mid b \in \Delta_T\}$ and $\Delta_T \cap \Delta' = \emptyset$

If $(p', x) = \gamma_T(p, a)$ and $q' = \delta_B(q, a)$ then $\gamma((p, q), a) = ((p', q'), y)$ where,

$p \in Q_T$, $q \in Q_B$, $a \in \Sigma$, and $y = \begin{cases} x \text{ if } q \notin F_B \\ x' \text{ if } q \in F_B \end{cases}$

Without loss of generality, we assume that *A* was given in "last letter normal form" and now we show the construction of *A*' in dual normal form.

$A' = (Q, \Delta, \$, \tau, I, F, \delta)$

$Q = Q_A \cup Q'_A$ where $Q'_A = \{q' \mid q \in Q_A\}$

Δ as described for *T'*

$I = I_A \cup I'_A$ where $I'_A = \{q'_i \mid q_i \in I\}$

$a, a' \in \tau(q)$ if $a \in \tau_A(q)$

$a, a' \in \tau(q')$ if $a \in \tau_A(q)$

$F = \{q_f\}$ the only accepting state where $q_f \in F_A$

$\delta: Q \times \Delta \to 2^Q$

$p, p' \in \delta(q, a)$ if $p \in \delta_A(q, a)$

$p, p' \in \delta(q', a')$ if $p \in \delta_A(q, a)$

In this way, (T', A') is a T-inputNFAwtl and L(T,A) ∩ L(*B*) = L(T', A').

The proof is finished.

4.4 Examples

This section provides the examples of intersetion with regular language and closure under union. First in Example 3 the intersection with regluar languages is explained and then the closure under union is explained in Example 4.

Example 3: Let us consider the lanuage $L_{aw=}\{a^n w \mid w \in \{b,c\}^*$ and $|b|=|c|= n\geq 0\}$ is accepted by the T-inputNFAwtl, from Example 2 and language L_n = a*b*c* is accepted by a DFA. Figure 5 shows the graphical representation of DNT-inputFAwtl. We present the graphical representation of DFA of L_n and transducer *T*' of combo automaton that accept the intersection of L_{aw} with L_n in Figure 7, and Figure 8 respectively.

B:

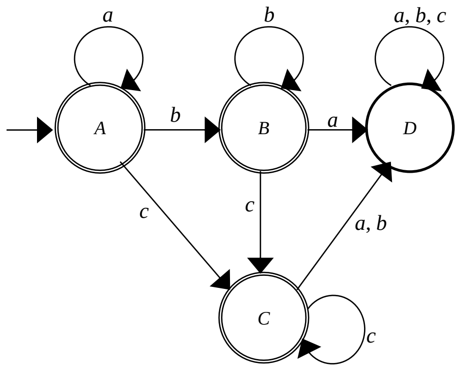


Figure 7. The DFA that accepts the language L_n.

T':

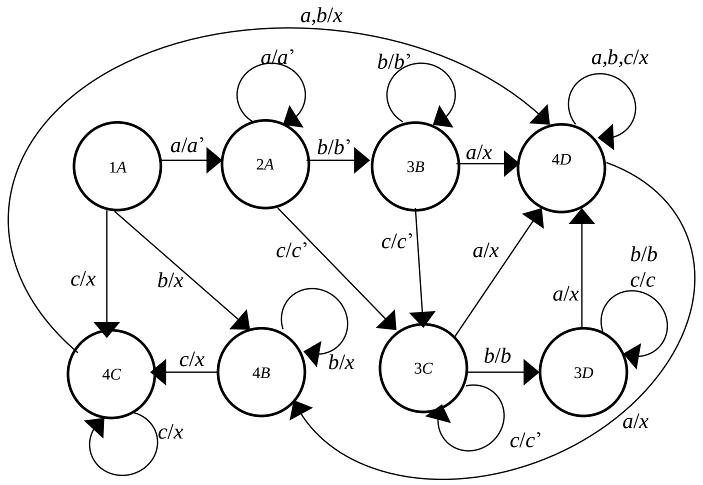


Figure 8. The transducer *T*'

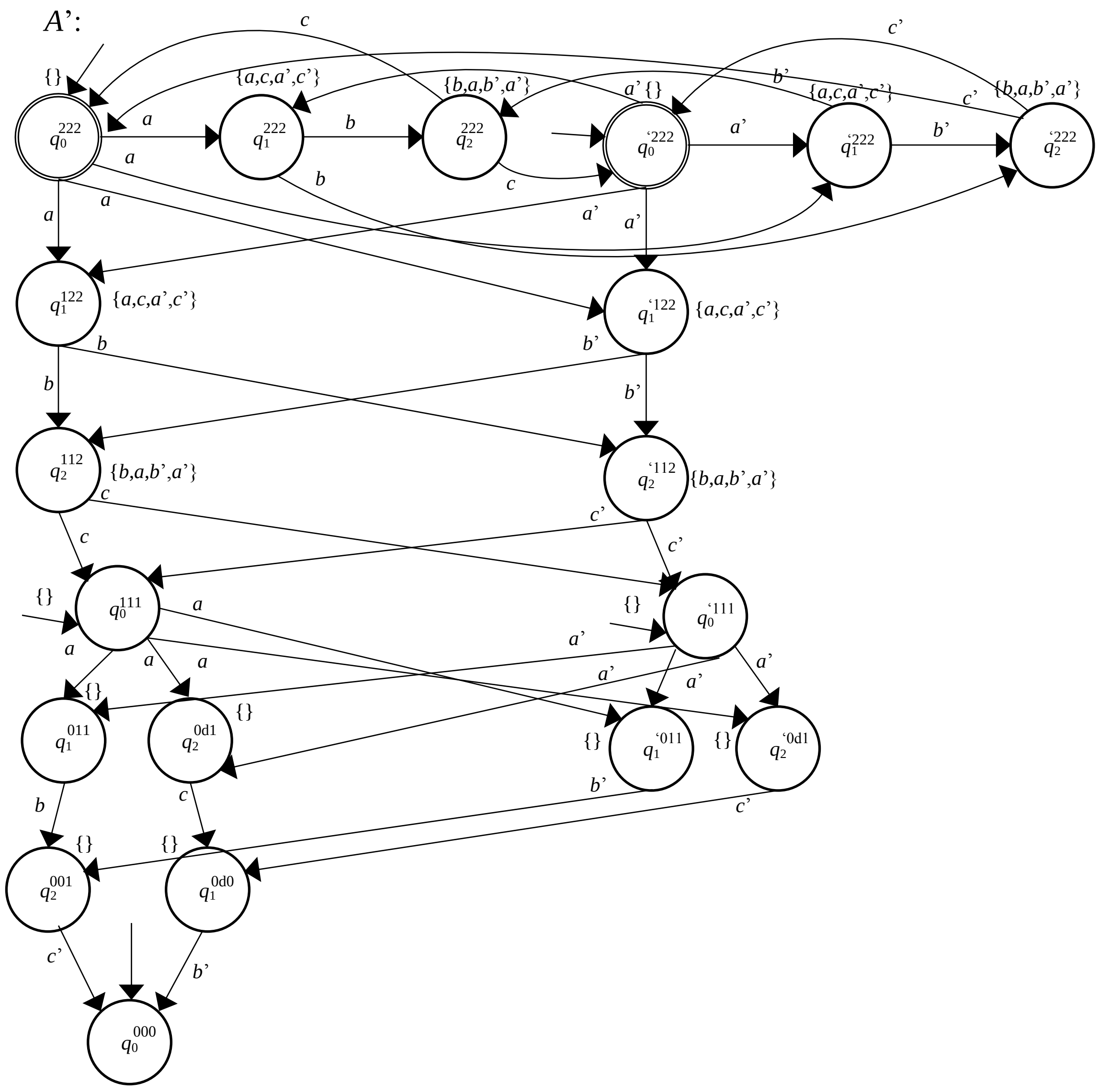


Figure 9. Partial graphical representation of the dual normal form NFAwtl for the language L_{aw}.

Now from Figure 9 we can clearly see how the dual normal form is working. It shows that we have set of initial states $I = \{q_0, q'_0\}$, with associating index we expand this set to $I = \{q_0^{222}, q_0^{111}, q'^{222}_0, q'^{111}_0, q_0^{000}\}$. Thus, the input word may only have one occurrence of each letter *a*, *b*, and *c* then the initial state will be q_0^{111} and if it has 2 or more occurrences of each letter in the input word then the initial state will be q_0^{222}. Both initial states $q_0^{222}$and q_0^{111} are copied to $q'^{222}_0$and q'^{111}_0 to allow to read the marked letter *a*' initially if *T*' provided such output.

Further, as we already discussed in Definition 3 that if the original state, let say q_0^{222} has a transition to read a letter a to state q_1^{122} and marked state q'^{222}_0 have transition for a' to q'^{122}_1 then there will also be a transition, for letter a from state q_0^{222} to state q'^{122}_1 and for letter a' from q'^{222}_0 to state q_1^{122} as well.

Let us consider the input word *aabbcc* is in the language. *T'* in Figure 5 will start transducing the input at state 1*A*, the letter *a* will be transduced to *a*' and state 2*A* will be reached. Then next *a* will be transduced to *a*' and it will remain on the same state. Then letter *b* will be transduced to *b*' and state 3*B* will be reached and again *b* will be transduced to *b*' and state will not be changed. From there letter *c* is transduced to *c*' and reached to 3*C*. By staying in the same state *T*' transduced next *c* to *c*' and produced an output *a*'*a*'*b*'*b*'*c*'*c*'.

Thus, the word *a*'*a*'*b*'*b*'*c*'*c*' is obtained and passed to *A*' with the end marker \$. In the initial state q'^{222}_0 nothing is translucent. *A'* will erase the first letter *a*' and after updating the index of *a*' it will move to next state q'^{122}_1 with remaining input word *a*'*b*'*b*'*c*'*c*'\$. At this state {*a*, *a*', *c*, *c*'}, are translucent, *A*' will erase first *b*' and will move to state q'^{112}_2 by updating the index of *b*.

The remaining input is *a*'*b*'*c*'*c*'\$. Now at this state q'^{112}_2, {*a*', *b*'} are translucent. *A*' will now read first *c* and will move to state q'^{111}_0 with remaining input is *a*'*b*'*c*'\$.

At this state as we are reading last occurrences of *a*', *b*' and *c*', translucency mapping is empty. *A*' reads first letter i.e. *a*' and come to state q'^{011}_1 with remaining input *b*'*c*'\$. *A*' will now read first *b*' and will reach to state q_2^{001} with remaining input *c*'\$. Then finally reading last input letter *c* and reaches to accepting state q_0^{000} with $ end marker. It means that the word *a*'*a*'*b*'*b*'*c*'*c*' is completely processed and accepted.

Let us consider another word *acabbc*. *T*' transduced this word as *a*'*c*'*xxxx*. This world clearly will not be accepted by *A*'. The word *acabbc* is not in the language *L*(*T'*, *A*'). Observe that *A*' cannot read any letter *x* because no transition is defined with letter *x* which we used as a failure symbol and this input word will be discarded.

Example 4: Let us consider the language $L_{abc} = \{a^n b^n c^n \mid n \geq 0\}$ is accepted by a T-inputDFAwtl from theorem 1 and the language $L_{abbc} = \{ a^n b^{2n} c^n \mid n \geq 0\}$ is accepted by T-inputDFAwtl is given in Figure 10. Then Figure 11 shows the graphical representation of union of L_{abc} and L_{abbc}.

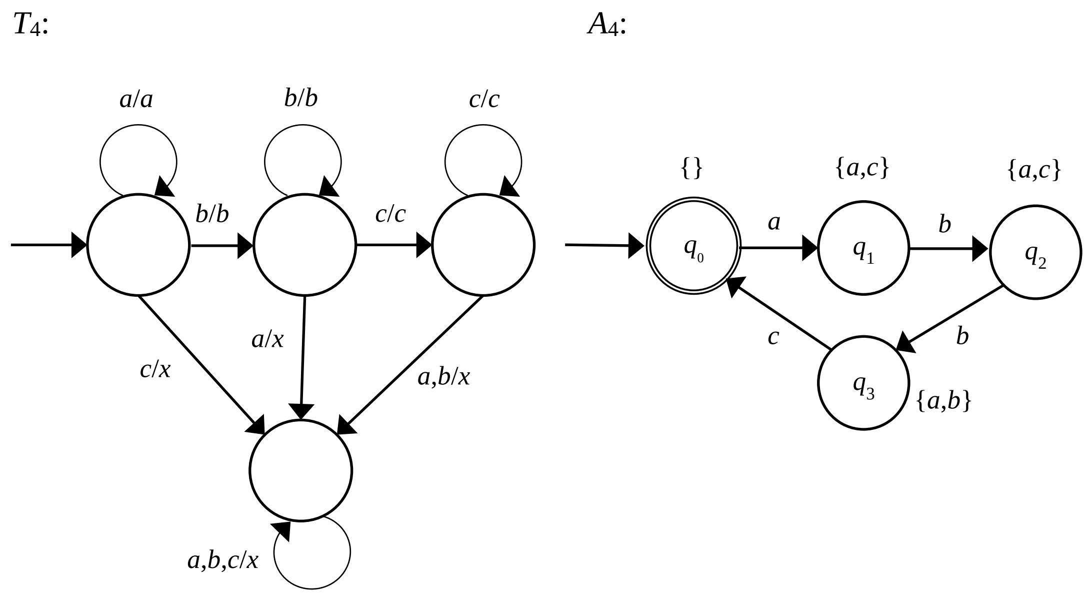


Figure 10. The T-inputDFAwtl that accepts the language L_{abbc}.

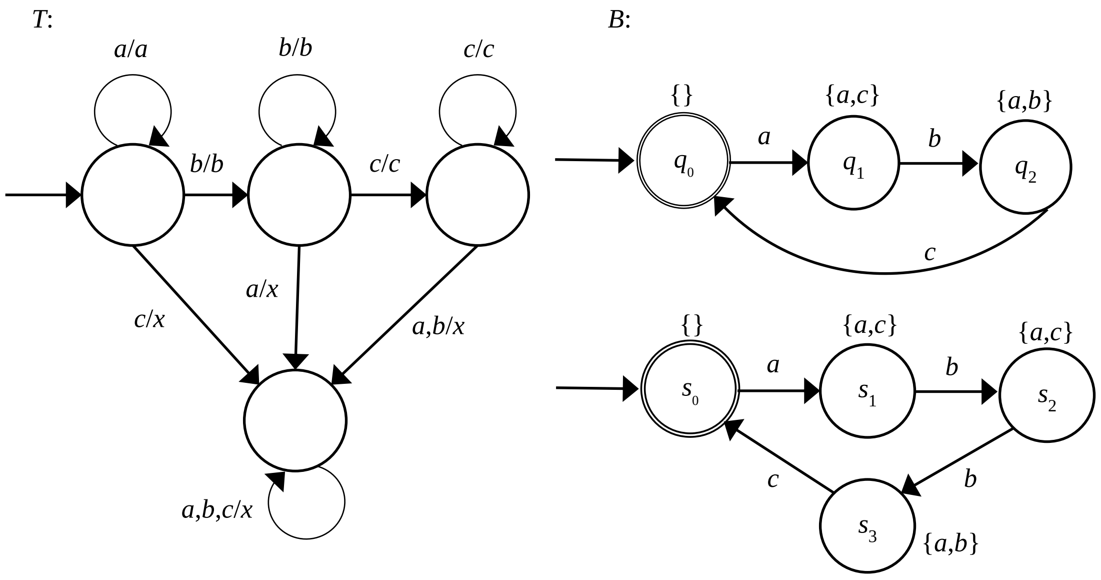


Figure 11. The T-inputNFAwtl that accepts the language $L_{abbc} \cup L_{abc}$.

In Figure 11 it is shown that the T-inputNFAwtl accepts the union of the langugages L_{abc} and L_{abbc} when same transducer is used. There is no intefrence of computations done by of A_1 and A_4 encoded in NFAwtl *B* presented in Figure 11, each of the accepting (and non-accepting) computations of A_1 and A_4 has a one-to-one correspondence with an accepting (non-accepting) computation of *B*, respectively.

Chapter 5

CONCLUSION

We have shown that a combination of deterministic finite state machines is very powerful, all the three important non-context-free mildly context-sensitive languages are accepted by this model. The language classes defined by these new models are interesting and we have stated some closure properties. Particularly, we have shown that the language class accepted by T-inputNFAwtl is closed under union with same signatures (where same signature means that the same transducer is used for preprocessing). Also, it was shown that the language class of T-inputNFAwtl is closed under intersection with regular languages. Further we intend to work on investigation of other closure properties with other cases of deterministic/non-deterministic versions of T-inputFAwtl with and without assuming the same signatures.

REFERENCES

[1] Nagy B., F. Otto (2011): Finite-State Acceptors with Translucent Letters. In Proc. BILC 2011: 1st Int. Workshop on AI Methods for Interdisciplinary Research in Language and Biology, 3-13.

[2] Nagy B., F. Otto (2012): On CD-systems of stateless deterministic R-automata with window size one. J. Computer and System Sci. 78 (2012), 780-806.

[3] Nagy B., F. Otto (2013): On Globally Deterministic CD-Systems of Stateless R-Automata with Window Size One. Int. J. Computer Mathematics 90 (2013), 1254-1277.

[4] Nagy B., F. Otto (2010): CD-Systems of Stateless Deterministic R(1)-Automata Accept all Rational Trace Languages. In Proc. LATA 2010: 4th Int. Conf. Language and Automata Theory and Applications, LNCS 6031 (2010), 463-474.

[5] Nagy B., L. Kovács (2013). Linguistic Applications of Finite Automata with Translucent Letters. In Proc. ICAART 2013: 5th International Conference on Agents and Artificial Intelligence, Barcelona, vol. 1, 461-469.

[6] Nagy B., L. Kovács (2014). Finite Automata with Translucent Letters applied in Natural and Formal Language Theory. LNCS Transactions on Computational Collective Intelligence 17, LNCS-TCCI XVII, LNCS 8790, 107-127.

[7] Jäger G., J. Rogers (2012). Formal language theory: refining the Chomsky hierarchy. Philos Trans R Soc Lond B Biol Sci. 367(1598): 1956-1970.

[8] Morawietz F. (2003). Two-Step Approaches to Natural Language Formalism. Studies in Generative Grammar 64. De Gruyter, Berlin.

[9] Gazdar G. (1982). Natural languages and context-free languages. Linguist. Philos. 4, 469-473.

[10] Kornai A. (1985). Natural languages and the Chomsky hierarchy. In Proceedings of EACL'85, 1-7.

[11] Wintner S. (2002). Formal language theory for natural language processing. In Proceedings of ACL'02, 71-76.

[12] Hopcroft J.E, R. Motwani, J.D Ullman (2006). Introduction to Automata Theory, Languages, and Computation. 3rd Edition. Addison-Wesley, Boston, MA, USA.

[13] Horváth, G., B. Nagy (2014). Formal Languages and Automata Theory, Typotex, Budapest, Hungary.

[14] Linz P. (2011) An Introduction to Formal Languages and Automata. 5th Edition, Jones & Bartlett Learning, USA.

[15] Yu S. (1997). Regular languages (Chaper 2). In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. vol. 1, pp. 41-110. Springer, Berlin.

[16] Mealy G.H. (1955). A Method for Synthesizing Sequential Circuits. Bell System Technical Journal 34, 1045-1079.

[17] Kutrib M., A. Malcher, M. Wendlandt (2015). Tinput-Driven Pushdown Automata. In Proc. MCU 2015: 7th Int. Conf. Machines, Computations, and Universality, LNCS 9288, 94-112.

[18] Kutrib M., A. Malcher, M. Wendlandt (2017). Tinput-Driven Pushdown, Counter, and Stack Automata. Fundam. Inform. 155(1-2): 59-88.

[19] Nagy B., M. Fatima (2020). Transduced-input automata with translucent letters. Comptes rendus de l'Académie bulgare des Sciences, Vol 73, No1, pp.33-39.

[20] Carroll J., D.E. Long Darrell (1989). Theory of Finite Automata with an Introduction to Formal Languages, Prentice-Hall, Inc., New Jersey.

[21] Angyal D., B. Nagy (2017). An extension of the LR parsing algorithm for two-head pushdown automata, NCMA 2017: Ninth Workshop on Non-Classical Models of Automata and Applications, Prague, 71-86.

[22] Sipser M. (1996). Introduction to the Theory of Computation (1st ed.). International Thomson Publishing. USA.

[23] McDermid J.A. (1991). Software Engineer's Reference Book. Butterworth-Heinemann, Newton, MA, USA.

[24] Khalili A., T. Armando (2014). Learning Nondeterministic Mealy Machines. International Conference on Grammatical Inference, in PMLR. 34:109–123.

[25] Mehdi M., A. Khan (2016). DNA Pattern Analysis using FA, Mealy and Moore Machines. International Journal of Computer Science and Information Security. 14:235-243.

[26] Bonsangue M., J. Ruten, A. Silva (2008). Coalgebraic Logic and Synthesis of Mealy Machines. Communications of The ACM. 231-245.

[27] Chomsky N. (1956). Three models for the description of language. IRE Transactions on Information Theory 2: 113–124.

[28] Joshi, A.K. (2010) Mildly Context-Sensitive Grammars. Technical Report; http://www.kornai.com/MatLing/mcsfin.pdf (12.11.2010)

[29] Mery, B., M. Amblard, I. Durand, C. Retor ́e (2006): A Case Study of the Convergence of Mildly Context-Sensitive Formalisms for Natural Language Syntax: from Minimalist Grammars to Multiple Context-Free Grammars. Rapport de recherche, nr.6042, INRIA Futurs, Parc Club Orsay Universit ́e, Orsay.