2-Head Pushdown Automata

Samson Ayodeji Awe

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

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

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

Prof. Dr. Nazim Mahmudov
Acting 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 Master of Science in Applied Mathematics and Computer Science.

Assoc. Prof. Dr. Benedek Nagy
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev _____
2. Prof. Dr. Rza Bashirov _____
3 Assoc. Prof. Dr. Benedek Nagy _____

ABSTRACT

Finite state automata recognize regular languages which can be used in text processing, compilers, and hardware design. 2-head finite automata accept linear context-free languages. In addition, pushdown automata are able to recognize context-free languages which can be used in programming languages and artificial intelligence. We distinguish between deterministic and nondeterministic finite automata, 2-head automata and also pushdown automata. The deterministic version of these machines is such that there is no choice of move in any situation while the non-deterministic version may have a choice of move. The present thesis describes 2-head pushdown automata which is more powerful than the pushdown automata and it is able to recognize some non-context-free languages as well. Throughout the thesis we try to focus on characterization of aforementioned machines.

Keywords: 2-head pushdown automata, non-context-free languages, deterministic automata, non-deterministic automata.

ÖZ

Sonlu otomata düzenli dilleri tanır ve metin işleme, derleyiciler ve donanım tasarımı icin kullanilabilir. Iki basli sonlu otomata doğrusal bağlamsiz dilleri kabul eder, ve ters otomata programlama dilleri ve yapay zeka konularinda kullanilabilen baglamsiz dilleri tanir. Sonlu otomat iki basli sonlu otomata ve asagi suruklemeli otomata'da oldugu gibi deterministik ve deterministik olmayan versiyonlara sahiptir. Bu otomata'larin deterministik versiyonlarinda hareket etme secimi yapilamaz iken, deterministik olmayan versiyonlarda hareket seçimi yapmak mumkundur. Bu tezde ters otomata'dan daha güçlü olan bunun yani sira bazi baglamli dilleri de tanimakta olan 2-basli asagi suruklemeli otomata tarif edilmiştir. Bu çalışmalar sırasında, temel olarak yapilan is bu otomata'lari karakterize etmektir.

Anahtar Kelimeler: 2-basli asagi suruklemeli otomata, bağlamli serbest diller, deterministik otomata, deterministik olmayan otomata.

ACKNOWLEDGMENT

My heart of gratitude goes to my supervisor Assoc. Prof. Dr. Benedek Nagy for his support throughout the period of working on this thesis, he was always available and he guides me through every process of this thesis. I will also like to give thanks to my friend Yetunde Adeuja for her countless assistance and inspiration while I was writing this thesis.

The dedication of this thesis goes to my family for their countless support morally and financially. I also dedicate this thesis to my friend Adeuja Yetunde.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

A, C, G &T	Adenine, Cytosine, Guanine, Thymine
D2HPDA	Deterministic 2-Head Pushdown Automata
DNA	Deoxyribonucleic Acid
ID	Instantaneous Description
LIFO	Last in First Out
PDA	Pushdown Automata
WK	Watson-Crick
WKA	Watson-Crick Automata
2HPDA	2-Head Pushdown Automata
3'→5'	Three Prime to Five Prime
5'→3'	Five Prime to Three Prime

Chapter 1

INTRODUCTION

Automata theory is the study of abstract computing devices or "machines" before there were computers [6]. In the field of automata theory; there are various kinds of automata recognizing various classes of languages [6]. The 2-head automata are equivalent to the WKA which was originally introduced as model of automata working on DNA strands as input. The DNA molecule encodes the genetic instructions used in the growth and behavior of all living organisms [9]. DNA has double stranded helical structure one strand has 5' to 3' structure, and another is reverse ordered. The DNA's two strands run in opposite directions to each other and they are anti-parallel. If we untwist the DNA, it will result in a double helix structure and from the study of molecular biology the DNA looks like two parallel strands in which each of the strands has a linear sequence of *Adenine, Cytosine, Guanine and Thymine* (*A, C, G, T*) respectively. The exact arrangement of the letters contain the instructions that are coded such that one strand is an image that complements the other i.e. *A* always pairs with *T*, and *C* always pairs with *G* as shown in Figure 1 below. By computational point of view, DNA molecules can be seen as double strings over the four letter alphabet. So if you know the sequence of one strand, you can work out the sequence of the other. Watson and Crick discovered the double helix structure of DNA and the fact that the two chains run in opposite directions [11]. WKA depicts an instance of mathematical model extracting biological properties for the purposes of computation [1]. The WKA are finite automata, their

two reading heads work on double stranded sequences. The two strands of the input are separately scanned by the read only heads and these read only heads are controlled by a common state. All regular languages can be accepted by the WKA [2]. If the two heads step together reading the same input letter at each transition then we simulate the work of a traditional finite state automata. But the accepting power of WKA is larger than one head finite automata.

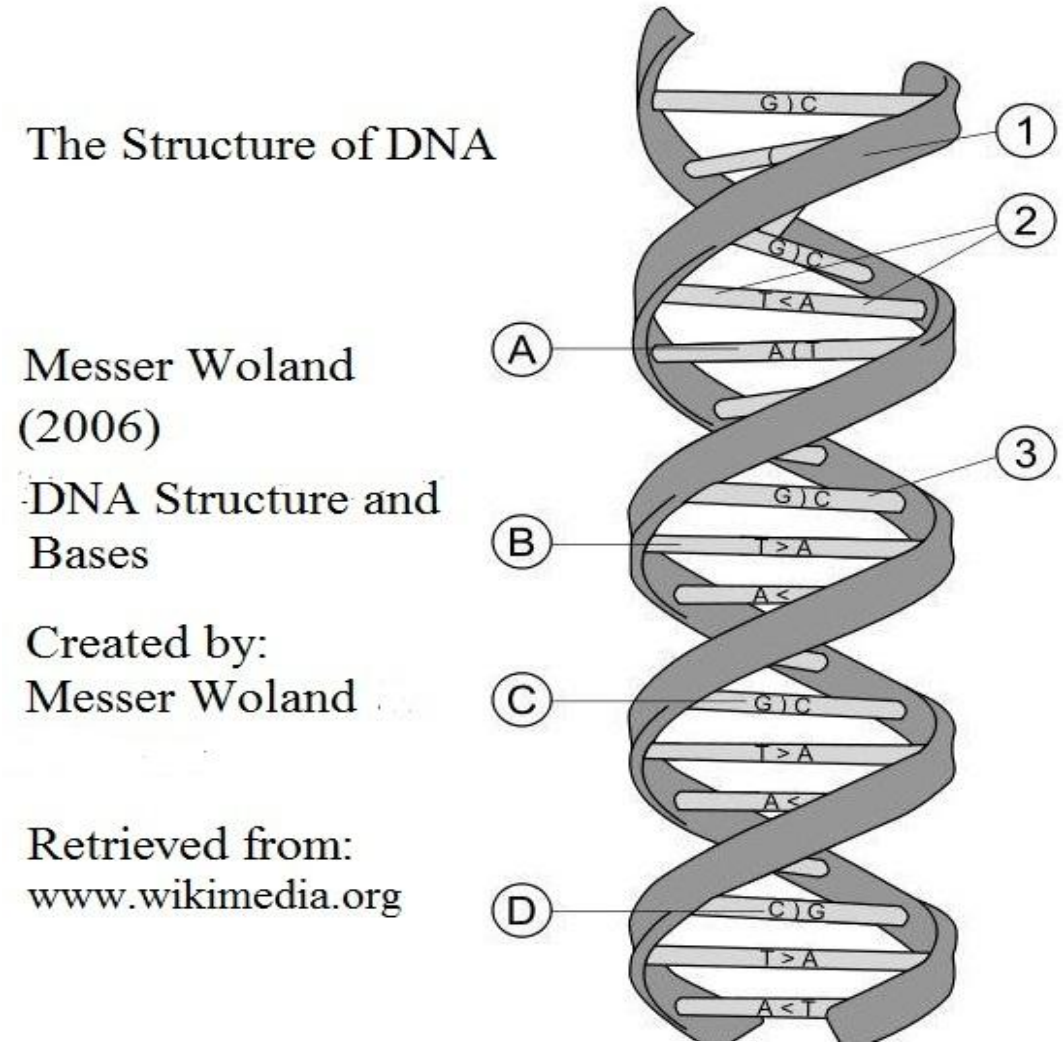

Figure 1. The Structure of DNA

The pushdown automata accept exactly the context free languages [5, 6]. Our new model, the 2HPDA accept some non-context free languages; we as well also consider

the subclass of the 2HPDA that is deterministic. The thesis is arranged as follows: the next section shows the fundamental definitions of the WKA, pushdown stack, the pushdown automaton. The informal description of the 2HPDA, the formal definition of the 2HPDA, the graphical notations, the instantaneous description, the languages of 2HPDA, the closure properties, the definition of deterministic version and example of deterministic version of the 2HPDA.

Chapter 2

LITERATURE REVIEW

Formal languages and automata theory are one of the essential foundation fields of theoretical computer science and they are rooted in the middle of the last century and most of the classical results are from the last century. Automata theories find important applications in other fields of computer science and information technology e.g. compiler technologies, operating systems etc. There are some new developments connected to various fields [6]. Some of the new developments are the modeling of DNA sequences, and 2D pictures. Basically in the formal language theory there has being a productive impact by the new model of computation caused by biological processes and these biological processes had being promoting the improvement of formal language theory. In 1987, the biological phenomenon of recombinant processes was mathematically formalized and was proposed by Tom Head [3]. Initiating a theoretical model of computation by using biological processes and an example of this is the WKA.

As the interest in using DNA in computation increased so did the need for automata which exploit the properties of DNA. The first such automata which exploited the DNA property were the traditional WKA [2]. WKA are examples of mathematical model, this remarkable model of automata abstract the biological features of DNA for the usage of computation. The WKA are finite automata with two reading heads controlled by one finite state control; these two reading heads are operational on

double stranded tapes (inputs), these heads separately scan from 5' end to 3' end. Over the time, several variants and restrictions of WKA were introduced and investigated, for example, stateless WK finite automata, WK two way finite automata, WKA with a Watson-Crick memory and Watson-Crick transducers [1]. Likewise, Simple and 1-limited WKA with bounded number of leaps between the two strands were investigated in [7]. While Watson-Crick ω-automata were discussed in [10] also 5'→3' Sensing WK Finite Automata was described in [8]. The main essence of these 2-head automata is not only that they could process the input twice faster than the traditional automata due to the two heads but they are accepting a wider class of languages [4]. One of the current trends in nanoengineering is to develop nanomachines which can parse molecules of DNA and perform a finite number of tasks, e.g., the development of artificial enzymes [11] in line with this; formal languages can be used to analyze the strings into logical syntactic components. The term "pushdown" means the stack can be considered as being "pushed down" similar to a tray dispenser used in a cafeteria. The operation of the pushdown stack majorly works on the top element of the stack.

The set of finite automaton (both deterministic and nondeterministic ones) recognizes the class of regular languages, while the set of 2-head finite automata accepts the class of the linear context free languages. The class of regular languages is a proper subset of the class of linear languages. The set of pushdown automata are able to recognize the class of context-free languages and the set of linear bounded automata accepts the class of context-sensitive language.

Chapter 3

PRELIMINARIES

3.1 5'→3' WKA

5'→3'WKA are finite state automata working on double-stranded tapes, that are initiated to investigate the ability of DNA molecules for computing [8]. They have two heads, and the two heads are moving in opposite directions starting from the two extremes of the input. The two heads of the 5'→3'WKA move in anti-parallel, i.e., in opposite directions. This is the main difference between the traditional WKA [2, 9] and the 5'→3' WKA [8]. Moreover in the 5'→3' WKA the process terminates when the two heads meet, and do not move on to the corresponding ends of the word.

Formally, a 5' → 3' WKA is a 7 tuple (Q, V, s_o, F, δ, \$, ¥) as shown below:

Q	=	the finite Set of States
V	=	the input (tape) Alphabets
\$, ¥	=	end-markers of the Input Strings such that (\$, ¥ ∉ V)
$s_o \in Q$	=	is the Initial state
$F \subset Q$	=	the set of final (accepting) state
δ	=	is the state transition, a mapping $((Q \times V^* \times V^*) \rightarrow 2^Q) \cup ((Q \times (\$, ¥)) \rightarrow 2^Q)$

3.1.1 Accepted Language of 5'→3' WKA

The 5'→3' WKA accepts linear languages. Linear languages are context-free but not all context-free languages are linear. However, every regular language is linear, and, thus, the set of 5'→3' WKA accepts all regular languages.

3.2 Pushdown Stack

The stack is a LIFO memory which has two operations, push and pop when we use the pop operation, we read the top letter of the stack and at the same time we delete it and when we use the push operation we add some symbols to the top of the stack. The presence of a stack means that an automaton using a pushdown stack can remember an infinite amount of information.

3.3 Pushdown Automaton

The *Pushdown Automaton* (PDA) is in essence a nondeterministic finite automaton with empty word transitions (*λ-transition*) permitted and one additional capability: a stack and on the stack it can store a string of "stack symbols" and the PDA accepts exactly the context-free language [6].

$P = (Q, T, Z, \delta, q_{o,} z_{o,} F)$

The meanings of the components are as shown below:

Q = finite set of *states*

T = finite set of *input symbols*

Z = finite *stack alphabet*

δ = the *transition function*, δ controls the behavior of the automaton, formally δ takes as argument a triple $\delta (q, a , X)$ where:

$q \in Q$ = q is a *state* in Q

a is either an input symbol in T or $a = \lambda$ the empty string which is assumed not to be an input symbol.

X is the stack symbol i.e. a member of Z

q_o = the initial state the PDA is in this state before making any transition

Z_o = the *stack symbol*, initially the PDA'S stack consist of one instance of this symbol and nothing else

F = the set of *accepting state* or *final state*.

3.3.1 Accepted Language of PDA

The Pushdown Automata (PDA) accept the class of languages called the context-free languages and, for instance, a language L = $\{a^n b^n \mid n \geq 0\}$ can be accepted by a PDA P such that $L(P) = \{a^n b^n \mid n \geq 0\}$.

Chapter 4

2-HEAD PUSHDOWN AUTOMATA

4.1 Informal Description of a 2HPDA

A 2HPDA is a theoretical device that has two reading heads that reads the *input string* (built by symbols) on a tape. The reading heads are anti-parallel to each other i.e. they move in opposite (anti parallel) directions. The 2HPDA has a "finite-state control" and a "pushdown stack".

We can view the 2HPDA informally as the device suggested in Figure 2 below. A "finite-state control" reads two input symbols at the same time, the 2HPDA is allowed to observe the symbol at the top of the stack or alternatively, it may make a spontaneous transition using "λ" as its input instead of an input symbol. It observes the symbol on top of the stack according to transition rules in the "finite-state control".

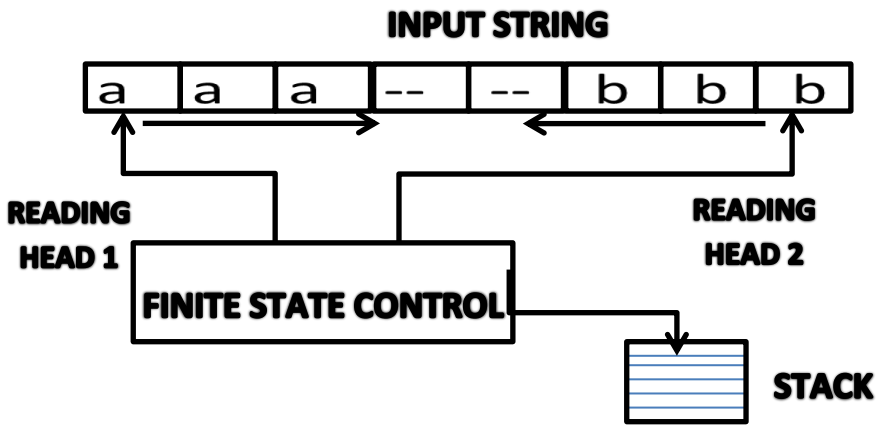

Figure 2. The schematic diagram of a 2HPDA

4.2 The Formal Definition of a 2HPDA

The formal notation for a two head pushdown automaton (2HPDA) involves seven components. The specification of a 2HPDA H is as follows:

M = $<Q, T, Z, \delta, z_o, q_{o,} F>$

Q = finite set of states

T = input symbols e.g. $\{a, b, c\}$

Z = finite stack alphabet (it's the set of symbols we are allowed to push onto the stack)

δ = it is the mapping from $Q \times (T \cup \{\lambda\})^2 \times Z \rightarrow 2^{Q \times Z^*}$

$q_o \in Q$ = it is the initial state

$z_o \in Z$ = initial stack symbol (initially the pushdown stack consisting of one instance of this symbol and nothing else)

F = set of accepting states or final states

4.3 A Graphical Notation for 2HPDA

The behavior of a given 2HPDA can easily be understood through a transition diagram. A transition diagram of a 2HPDA is explained below:

a) The nodes corresponds to the states of the 2HPDA

b) The arrow that is labeled *Start* indicates the initial state, and double circled states are the accepting states.

c) The arcs represent the transitions of the 2HPDA in the following way: an arc labeled $a, b, \beta / \alpha$ from state q to state p means that $\delta(q, (a, b), \beta)$ contains the pair (p, α) meaning that the arc label shows the inputs that are used and also shows the old and new tops of the stack.

4.5 Instantaneous Description of 2-Head Pushdown Automata

Intuitively the 2-head pushdown automata (machine) go from configuration to configuration in response to input symbols or sometimes no input symbol. The 2HPDA configuration includes the *state* and the content of the *stack* together with the *input symbols* and it has an instantaneous description. The standard step is when the 2HPDA reads its current state, current input letters, the top stack symbol then it finds an appropriate transition rule and changes its state. It moves to the next input letters and changes the top symbol of the *stack* to the appropriate word formally we say the 2HPDA can change its configuration from

$(q_1, aub, \beta\,\mu) \vdash (q_2, u, \alpha\,\mu)$

Here, $q_1, q_2 \in Q$, $u \in T^*$, $a, b \in (T \cup \{\lambda\})$, $\alpha \in Z^*$, $\beta \in Z$

q_1 = previous state

q_2 = new State

(a, b) = a pair of input symbols (any or both of them could also be the empty word)

α = new word on top of the stack

β = previous symbol on top of the stack

This move shows that by consuming (a, b) which may be (λ, b) (a, λ) (λ, λ) from the input and replacing β on top of the stack by α we consequently can go from the current state q_1 to state q_2.

Chapter 5

THE LANGUAGES OF A 2HPDA

5.1 Acceptance of an Input String

The word w is accepted if $(q_o, w, z_o) \vdash^* (p, \lambda, \mu)$, where

(q_o, w, z_o) is the so-called initial configuration, i.e.

q_o = initial state

w = the whole input string

z_o = initial stack symbol

$p \in F$ = an accepting state

$\mu \in Z^*$ = string of stack symbols

5.2 Various Acceptance Conditions

We have therefore guessed that a 2HPDA accepts its inputs when its input is consumed thereafter, entering an accepting state. We hence call this process "acceptance by final state" there is a second process to defining the language of a 2HPDA which means that we may also define for any 2HPDA the language 2HPDA the language "accepted by empty stack" i.e. the set strings that cause the 2HPDA to empty its stack, starting from the initial ID.

5.3 Accepted Language by Final State

Let $M = <Q, T, Z, \delta, Z_o, q_o, F>$ be a 2HPDA, then $L(M)$ is the language that is accepted by M by final state is:

$$\{w \mid (q_o, w, z_o) \vdash^* (p, \lambda, \mu)\}$$

For some state $p \in F$ and any stack string μ. That is starting in the initial ID with w waiting on the input; M consumes w from the input and enters an *accepting state*. The content of the stack at that time is irrelevant.

Example 5.1

For the language $L = \{a^n\, b^n\, c^n : n \geq 0\}$, consider the

2HPDA = ($\{q_o, q_1, q_2\}$,{ $a, b, c\}$,$\{a, Z_o\}$, q_o, Z_o, δ,$\{q_2\}$) having the following transition function:

$\delta\ (q_o, (a, c), Z_o) = \{(q_o, aZ_o)\}$

$\delta\ (q_o, (a, c), a) = \{(q_o, aa)\}$

$\delta\ (q_o, (b, \lambda), a) = \{(q_1, \lambda)\}$

$\delta\ (q_1, (b, \lambda), a) = \{(q_1, \lambda)\}$

$\delta\ (q_1, (\lambda, \lambda), Z_o) = \{(q_2, Z_o)\}$

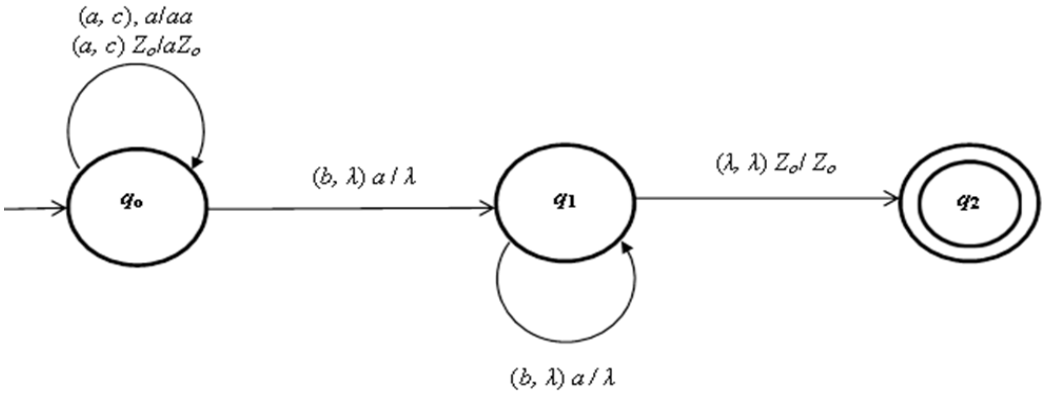

Figure 3. 2HPDA that accepts the language of Example 5.1 by a final state

Example 5.2

For the language $L = \{a^n\, b^n\, c^n\, d^n \mid n \geq 0\}$, consider the

2HPDA = ($\{q_o, q_1, q_2\}$, $\{a, b, c, d\}$, $\{a, b, Z_o\}$, q_o, Z_o, δ,$\{q_2\}$) having the following transition function:

$\delta\ (q_o, (a,\ d), Z_o) = \{(q_o,\ aZ_o)\}$

$\delta\ (q_o, (\ a,\ d), a) = \{(q_o,\ aa)\}$

$\delta\ (q_o, (\ b,\ c), a) = \{(q_1,\ \lambda)\}$

$\delta\ (q_1, (\ b,\ c), a) = \{(q_1,\ \lambda)\}$

$\delta\ (\ q_1, (\ \lambda,\ \lambda), Z_o) = \{(q_2,\ Z_o)\}$

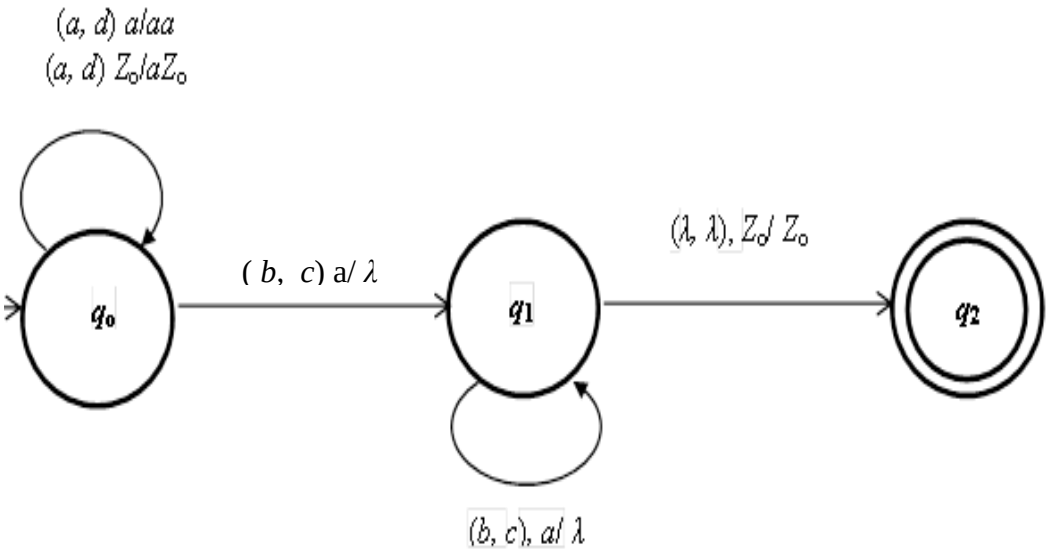

Figure 4. 2HPDA that accepts the language of Example 5.2 by a final state

5.4 Accepted Language by Empty Stack

Here the automaton does not have any final state and the word is accepted if it can read the whole word and the stack is empty when the two heads meet and the input string is consumed.

For each 2HPDA $M = (Q, T, Z, \delta, Z_o, q_{o,} F)$ we also define

$$N\ (M) = \{w \mid (q_{o,}\ w, Z_o)\ \vdash^*\ (p, \lambda, \lambda)\}$$

Now, because the set of accepting state is inapplicable, we shall sometimes discontinue the use of the last (seventh) component from the specification of a

2HPDA *M*, provided all we care about is the language that *M* accepts by empting its stack. Thus we could write M as a six-tuple.

$$2\text{HPDA}_{e} = (Q, T, Z, \delta, Z_o, q_o)$$

For any state *q*, i.e. *N* (*M*) is the set of inputs *w* that *M* can consume and at the same time empty its stack.

By empty stack L ($2HPDA_e$) = $\{p \mid p \in T^*, (q_o, p, Z_o) \vdash^* (p, \lambda, \lambda), p \in Q\}$

e.g., see the next example.

Example 5.3

Let $L = \{a^n b^n c^n \mid n \geq 0\}$, consider

$2\text{HPDA}_{e} = (\{q_o, q_1, q_2\}, \{a, b, c\}, \{a, Z_o\}, \delta, zo, q_o\})$ having the following transition function:

$\delta(q_o, (a, c), Z_o) = \{(q_o, aZ_o)\}$

$\delta(q_o, (a, c), a) = \{(q_o, aa)\}$

$\delta(q_o, (b, \lambda), a) = \{(q_1, \lambda)\}$

$\delta(q_1, (b, \lambda), a) = \{(q_1, \lambda)\}$

$\delta(q_1, (\lambda, \lambda), Z_o) = \{(q_2, \lambda)\}$

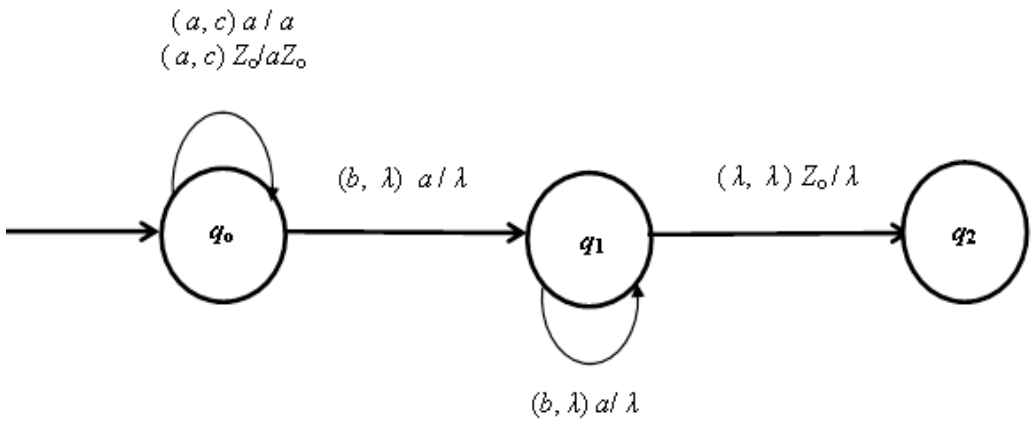

Figure 5. 2HPDA that accepts the language of Example 5.3 by empty stack

Consequently, with 2HPDA we may accept non-context-free languages and below is the diagram of an (extended) Chomsky Hierarchy with 2-head pushdown languages as proposed by this research.

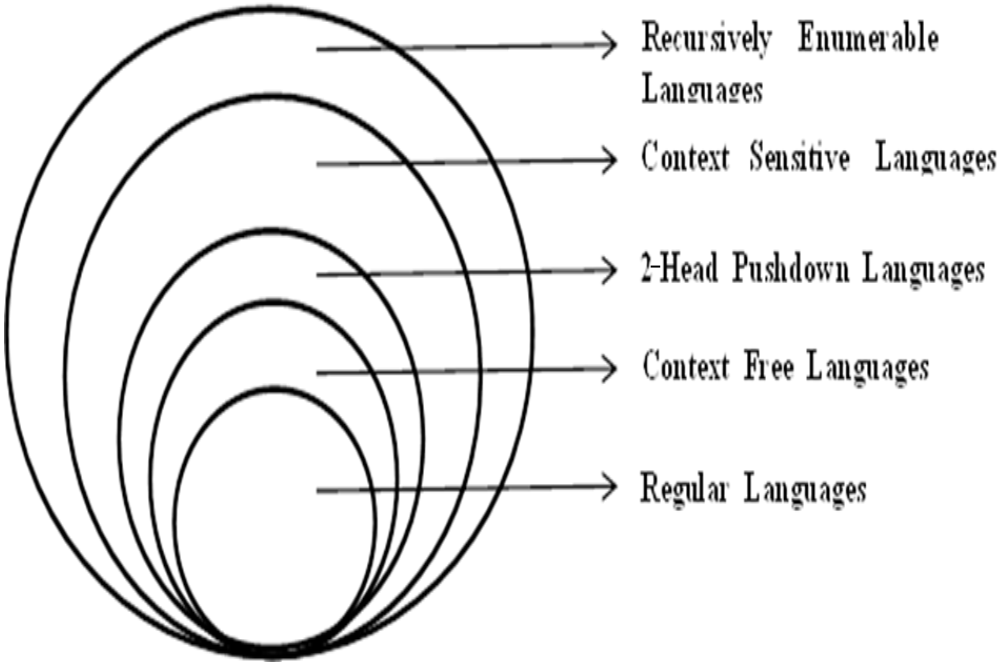

Figure 6. Extended Chomsky Hierarchy

5.5 Closure Properties

5.5.1 Closure under Union

The class of 2-Head Pushdown languages is closed under union and for this, we can use a proof that goes by construction of automata. Now we show it on an example. Suppose $2HPDA_1$ accepts language $L_1 = \{a^n b^n c^n \mid n \geq 0\}$ and $2HPDA_2$ accepts a language $L_2 = \{a^n b^n c^n d^n \mid n \geq 0\}$, then a $2HPDA_{1\cup 2}$ is a 2-head pushdown language.

The construction is as follows:

$2HPDA_{1\cup 2} = (\{q_o, q_1, q_1', q_2, q_2', q_3\}, \{a,b,c,d\}, \{a, Z_o\}, q_o, Z_o, \delta, \{q_3\})$ has the following transition function:

$\delta\,(q_o, (\lambda, \lambda), Z_o) = \{(q_1, Z_o)\}$

$\delta\,(q_1, (a, c), Z_o) = \{(q_1, aZ_o)\}$

$\delta\,(q_1, (a, c), a) = \{(q_1, aa)\}$

$\delta\,(q_1, (b, \lambda), a) = \{(q_2, \lambda)\}$

$\delta\,(q_2, (b, \lambda), a) = \{(q_2, \lambda)\}$

$\delta\,(q_2, (\lambda, \lambda), Z_o) = \{(q_3, Z_o)\}$

and

$\delta\,(q_o, (\lambda, \lambda), Z_o) = \{(q_1', Z_o)\}$

$\delta\,(q_1', (a, d), Z_o) = \{(q_1', aZ_o)\}$

$\delta\,(q_1', (a, d), a) = \{(q_1', aa)\}$

$\delta\,(q_1', (b, c), a) = \{(q_2', \lambda)\}$

$\delta\,(q_2', (b, c), a) = \{(q_2', \lambda)\}$

$\delta\,(q_2', (\lambda, \lambda), Z_o) = \{(q_3, Z_o)\}$

Thus, the new automaton can non-deterministically simulate any of the automata accepting the language L_1 and L_2.

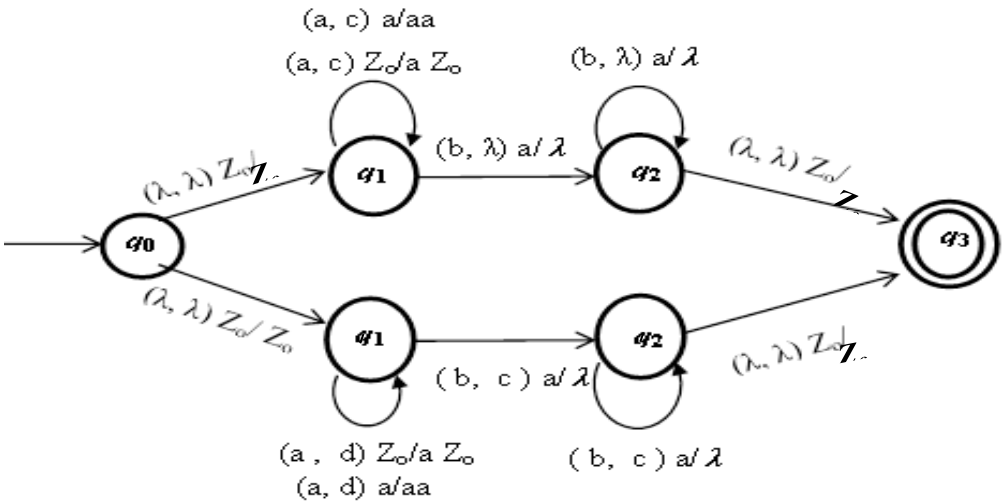

Figure 7. 2HPDA accepting the union of two 2HPDA languages

5.5.2 Closure under Reversal

The class of languages accepted by *2HPDA* is closed under reversal. This can also be proven by construction of automata. Now we show an example of such construction. Suppose, the 2*HPDA* accepts the language $L = \{a^n b^n c^n d^n \mid n \geq 0\}$. The construction is the following:

2HPDA= ($\{q_o, q_1, q_2\}$, $\{ a, b, c, d \}$, $\{ a, Z_o\}$, q_o, Z_o, δ, $\{q_2\}$) has the following transition function:

$\delta (q_o, (a, d), Z_o) = \{(q_o, aZ_o)\}$

$\delta (q_o, (a, d), a) = \{(q_o, aa)\}$

$\delta (q_o, (b, c), a) = \{(q_1, \lambda)\}$

$\delta (q_1, (b, c), a) = \{(q_1, \lambda)\}$

$\delta (q_1, (\lambda, \lambda), Z_o) = \{(q_2, Z_o)\}$

and

$2HPDA^R$= ($\{q_o, q_1, q_2\}$, $\{ a, b, c, d \}$, $\{a,Z_o\}$, q_o, Z_o, δ, $\{q_2\}$) has the following transition function:

$\delta (q_o, (d, a), Z_o) = \{(q_o, dZ_o)\}$

$\delta (q_o, (d, a), a) = \{(q_o, dd)\}$

$\delta (q_o, (c, b), a) = \{(q_1, \lambda)\}$

$\delta (q_1, (c, b), a) = \{(q_1, \lambda)\}$

$\delta (q_1, (\lambda, \lambda), Z_o) = \{(q_2, Z_o)\}$

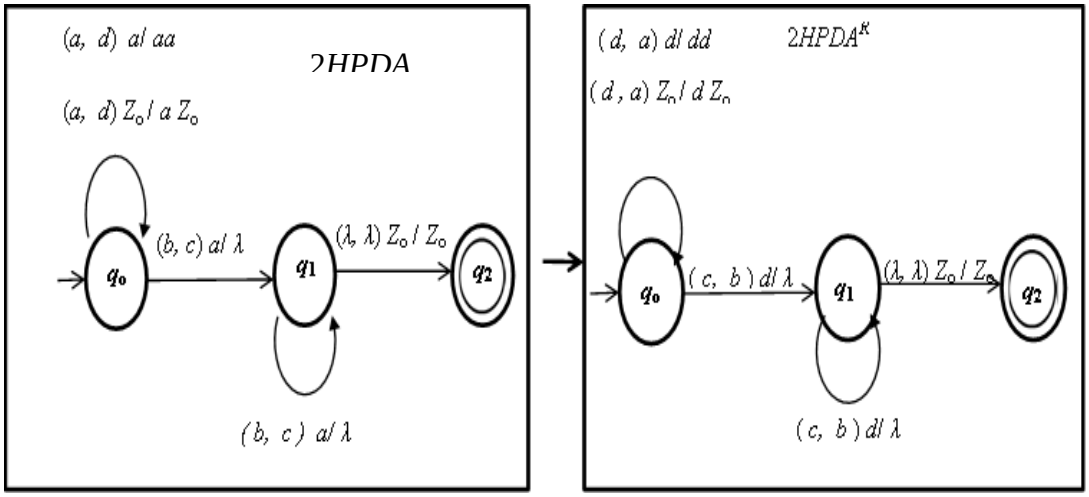

Figure 8. 2HPDA accepting the reversal of a 2HPDA language

5.6 The Formal Definition of a Deterministic 2HPDA

The 2-head pushdown automata 2HPDA $_d$ = (Q, T, Z, q_o, Z_o, δ, F) is deterministic if and only if

• For every $q \in Q$, every $z \in Z$, if $\delta(q, \lambda, z) \neq \emptyset$ then for every $a \in T$, $\delta(q, a, z) = \emptyset$, a λ move is possible from q with z on top only if no other move is possible.

• For every $q \in Q$, every $z \in Z$, every $a \in T \cup \{\lambda\}$, $|\delta(q, a, z)| \leq 1$. There is at most one allowed transition from any ID.

Q = finite set of states

T = input symbols e.g. $\{a, b, c\}$

Z = a finite stack alphabet (it's the set of symbols we are allowed to push onto the stack)

δ = it is the mapping from $Q \times (T \cup \{\lambda\})^2 \times Z \rightarrow Q \times Z^*$

$q_o \in Q$ = it is the initial state

$z_o \in Z$ = the stack symbol (initially the 2HPDA stack consisting of one instance of this symbol and nothing else)

F = accepting states

It holds that

{Languages accepted by D2HPDA}⊆{Languages accepted by 2HPDA}

Since every D2HPDA is also a 2HPDA the above statement is true.

5.6.1 Deterministic 2HPDA accepting a Non-Context-Free Language

We show that there exist a non-context-free language *L* which can be accepted by the deterministic version of the 2HPDA. Consider the next example.

Example 5.4

The deterministic 2HPDA below accepts the non-context-free language

$L(M) = \{a^n b^{2n} a^n \mid n \geq 0\}$.

Let the D2HPDA = $(\{q_o, q_1, q_2, q_3\}, \{a, b, c\}, \{a, z_o\}, q_o, Z_o, \delta, \{q_o, q_3\})$ with the following transition function:

$\delta\ (q_o,\ (a\ ,a), Z_o) = \{(q_1, aZ_o)\}$

$\delta\ (q_1,\ (a\ ,a), a) = \{(q_1,\ aa)\}$

$\delta\ (q_1,\ (b\ ,b), a) = \{(q_2,\ \lambda)\}$

$\delta\ (q_2,\ (b\ ,b), a) = \{(q_2,\ \lambda)\}$

$\delta\ (q_2,\ (\lambda\ ,\lambda), Z_o) = \{(q_3,\ Z_o)\}$

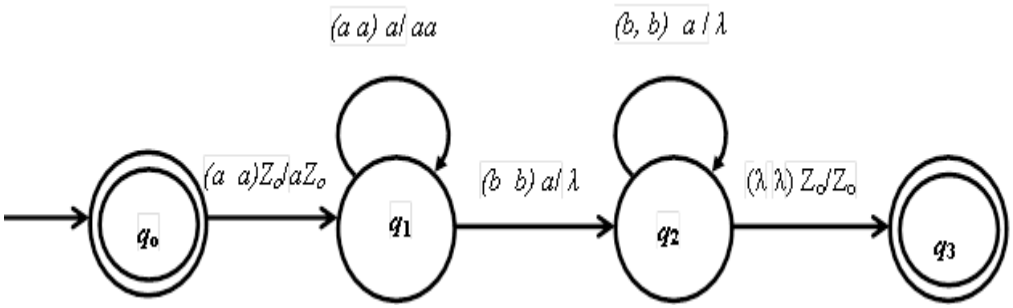

Figure 9. Deterministic 2HPDA accepting a non-context-free language

Chapter 6

CONCLUSION

We have defined the 2HPDA and have shown that the 2HPDA can accept some non-context free language and deterministic version of the 2HPDA has also being defined and it is also shown that this version can accept some non-context-free language. This research has successfully modeled the properties of a DNA and has combined it with a pushdown stack such that an acceptance requires that we have an empty stack at the end of the computation.

REFERENCES

[1] Czeizler, E. & Czeizler, E. A Short Survey on Watson Crick Automata. Bulletin of the European Association of Theoretical Computer Science 88 (2006) 104-119.

[2] Freund, R., Paun, Gh., & Rozenberg, G., A Watson-Crick Finite Automata, Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, (1997), 291- 329

[3] Head, T. Formal Language Theory and DNA : An Analysis of the Generative Capacity of Specific Recombinant Behavior, The Bulletin of Mathematical Science

[4] Herendi, T., Nagy, B. The Parallel Approach of Algorithms, Typotex, Budapest, 2014 http://www.interkonyv.hu/

[5] Horvath, G., Nagy, B. Formal Languages and Automata Theory, Typotex, Budapest, 2014

[6] Hopcroft, J. E., Motswani, R. & Ullman, J. D. Introduction to Automata Theory, Languages and Computation; (2nd Edition) Addison-Wesley 2001

[7] Martin-Vide, C., Paun, Gh. Normal Forms for Watson-Crick Finite Automata, F. Cavoto (Ed). The Complete Linguist: A Collection of Papers in Honor of Alexis Manaster Ramer: Lincom Europa. Munich (2000) 281 - 296

[8] Nagy, B. On 5'→3' Sensing Watson Crick Finite Automata: Proceedings of the 13th International Conference on DNA computing. LNCS Springer-Verlag Berlin, (2008) 256 – 262.

[9] Paun, Gh., Rozenberg, G & Salomaa, A. DNA Computing: New Computing Paradigms, Springer – Verlag, Berlin. (1998).

[10] Petre, E., Watson-Crick ω-Automata, J. Autom. Lang. Comb. 8 (2003), 59-70.

[11] Watson, J.D. & Crick, F. H. C. A Structure for Deoxyribose Nucleic Acid Nature 171 (1953) 737 – 738