Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 195 (2015) 2037 – 2046

Procedia Social and Behavioral Sciences

World Conference on Technology, Innovation and Entrepreneurship

2-Head Pushdown Automata

Awe Ayodeji Samson*

Department of Mathematics, Eastern Mediterranean University, Mersin 10 Turkey, Famagusta, North Cyprus

Abstract

Finite state automata recognize regular languages which can be used in text processing, compilers, and hardware design. Two 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. The finite automaton has deterministic and non-deterministic version likewise the two head finite automata and the 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 has a choice of move. In this research the 2-head pushdown automata are described which is more powerful than the pushdown automata and it is able to recognize some non-context free languages as well. During this work, the main task is to characterize these machines.

Peer-review under responsibility of Istanbul Univeristy.

Keywords: 2-Head Pushdown Automata; Non-Context-Free Languages; Deterministic 2-Head Pushdown Automata.

1. Introduction

Automata theory is the study of abstract computing devices or "machines" before there were computers (Hopcroft, Motswani and Ullman, 2001). In the field of automata theory; there are various kinds of automata recognizing various classes of languages (Hopcroft, Motswani and Ullman, 2001). The 2-head automata are equivalent to the Watson-Crick Automata (WKA) which was originally introduced as model of automata working on Deoxyribose Nucleic Acid (DNA) strands as input. The DNA molecule encodes the genetic instructions used in the growth and behavior of all living organisms (Paun, Rozenberg and Salomaa, 1998).

* Corresponding author. Tel.:+905338206572
E-mail address: aweolufemi@yahoo.com

Peer-review under responsibility of Istanbul Univeristy.
doi:10.1016/j.sbspro.2015.06.225

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*. 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 (Watson and Crick, 1953) WKA depicts an instance of mathematical model extracting biological properties for the purposes of computation computers (Czeizler and Czeizler, 2006). 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 (Freund, Paun and Rozenberg, 1997). 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. The pushdown automata accept exactly the context free languages (Horvath and Nagy, 2014 and Hopcroft, Motswani and Ullman, 2001). Our new model, the 2-Head Pushdown Automata 2HPDA accept some non-context free languages; we as well also consider the subclass of the 2HPDA that is deterministic.

2. Preliminaries

2.1 5'→3' WKA

A 5'→3'WKA, these are finite state automata working on double-stranded tapes, initiated to investigate the ability of DNA molecules for computing (Nagy, 2008) It has two heads in which the two heads are moving in opposite directions starting from the two extremes of the input. In the 5'→3'WKA, the 2 heads move in anti-parallel (opposite direction) and this is the difference between the traditional WKA (Freund, Paun and Rozenberg, 1997 and Paun, Rozenberg and Salomaa, 1998**).** the 5'→3' WKA (Nagy, 2008), moreover in the 5'→3' WKA the two heads terminate when they meet, and do not move on to the corresponding ends of the word.

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

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

2.2 Accepted Language of a 5'→3' WKA

The 5'→3' WKA accepts linear languages. Though linear languages are context free but not all context free languages are linear but every regular language is linear so a 5'→3' WKA accepts all regular languages.

2.3 Pushdown Stack

The stack is a LIFO (last in first out) 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.

2.4 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 computers (Hopcroft, Motswani and Ullman, 2001)

$P = (Q, T, Z, \delta, q_o, z_o, F)$

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 behaviour 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*.

2.5 Accepted Language of a PDA

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

$L(P) = \{a^n b^n \mid n \geq 0\}$

3. Results

3.1 Informal Description of a 2HPDA

A 2HPDA is a theoretical device that has 2 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 figure2 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".

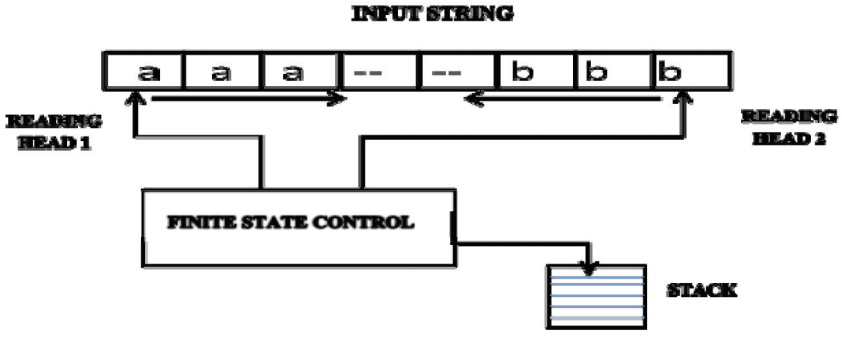

Fig.1. The Schematic Diagram of a 2HPDA

3.2 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 = accepting states

3.3 A Graphical Notation for 2HPDA

The behaviour 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 labelled *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 labelled $\alpha, \beta, X / \gamma$ from state q to state p means that $\delta(q, (\alpha, \beta), X)$ contains the pair (p, γ) meaning that the arc label shows the inputs that are used and also shows the old and new tops of the stack.

3.4 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.

3.5 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
$p \in F$ = accepting state
$\mu \in Z^*$ = string of stack symbols

3.6 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.

3.7 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.

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

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

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

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

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

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

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

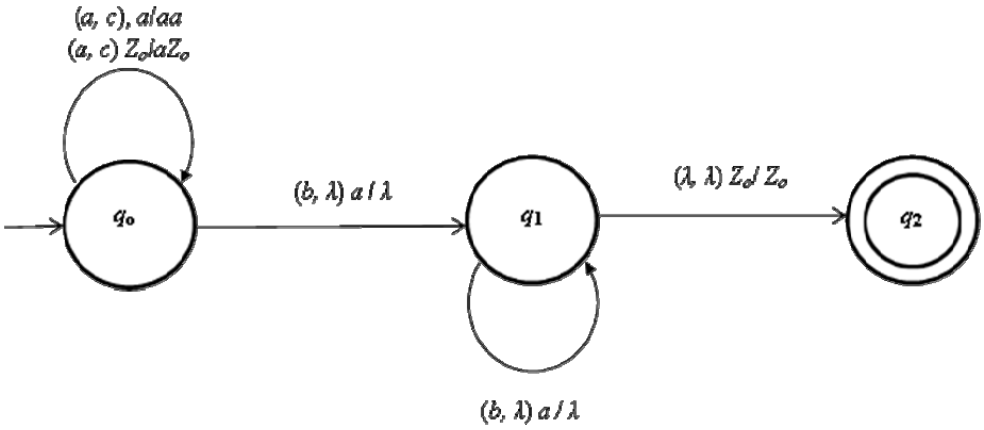

Fig.2. 2HPDA that accepting by final state

For the language $L = \{a^n b^n c^n d^n \mid n \geq 0\}$ Suppose a 2HPDA = $(\{q_o, q_1, q_2\}, \{a, b, c, d\}, \{a, b, Z_o\}, q_o, Z_o, \delta, \{q_2\})$

Has the following transition function:

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

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

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

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

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

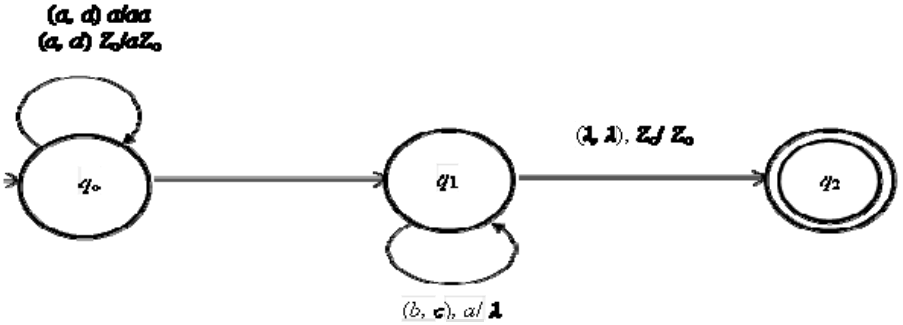

Fig.3. 2HPDA accepting by final state

3.8 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 end of the input word is reached.

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. $2HPDA_e = (Q, T, Z, \delta, Z_o, q_o)$

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. $L = \{a^n b^n c^n \mid n \geq 0\}$

Suppose a $2HPDA_e = (\{q_o, q_1, q_2\}, \{a, b, c\}, \{a, Z_o\}, \delta, Z_o, q_o\})$

Has the following transition function:

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

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

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

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

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

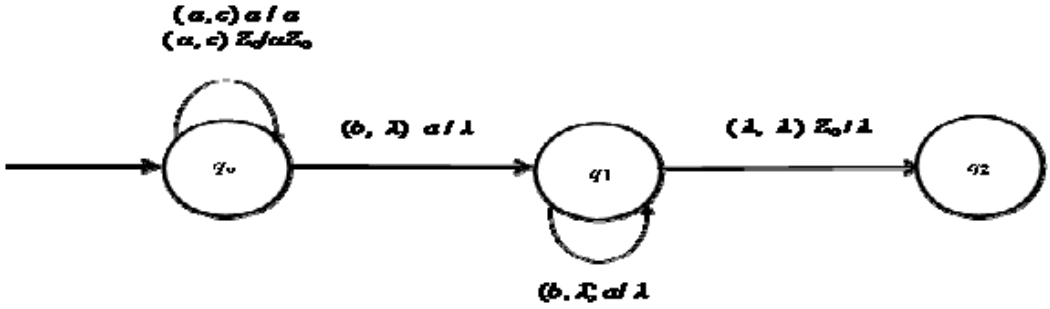

Fig.4. 2HPDA accepting by emptying the stack

Consequently, with 2HPDA we may accept non-context free language and below is the diagram of an (extended)

Chomsky Hierarchy with 2-head pushdown languages as proposed by this research.

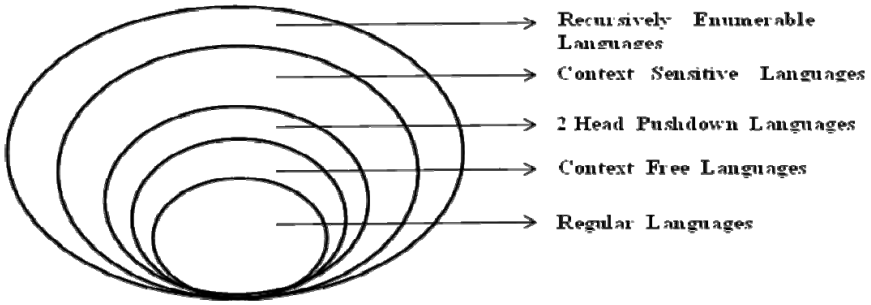

Fig.5 Extended Chomsky Hierarchy

3.9 Closure Properties

3.9.1 Closed Under Union

The *2HPDA* is closed under union and for this, we use a prove that goes by automata; suppose $2HPDA_1$ accepts language $L= \{a^n b^n c^n \mid n \geq 0\}$ and $2HPDA_2$ accepts a language $L = \{a^n b^n c^n d^n \mid n \geq 0\}$, then a $2HPDA_{1\cup2}$ is closed under union.

Proof by Construction

$2HPDA_{1\cup2} = (\{q_o, q_1, q_2, q_3\}, \{a,b,c,d\}, \{a, Z_o\}, q_o, Z_o, \delta, \{q_3\})$

Has the following transition function:

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

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

$\delta(q1, (a, c), a) = \{(q1, aa)\}$ $\quad$ $\delta(q_1, (a, d), a) = \{(q1, aa)\}$

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

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

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

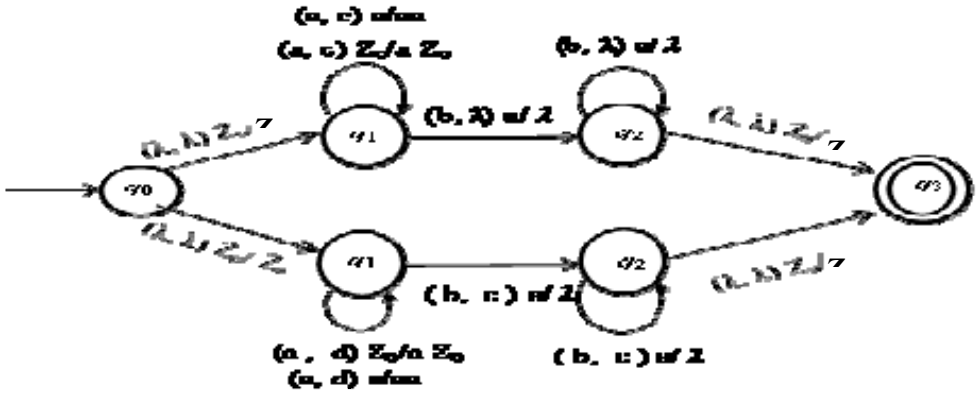

Fig.6.The Union of 2HPDA

3.9.2 Closed Under Reversal

The *2HPDA* is closed under reversal and for this, we use a prove that goes by automata; suppose 2*HPDA* accepts

language $L= \{a^n b^n c^n d^n \mid n \geq 0\}$ it is easily shown that it is closed under reversal.

Proof by Construction

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:
$\delta(q_o, (a, d), Z_o) = \{(q_o, aZ_o)\}$
δ (qo, (a, d), a) = {(qo, aa)}
$\delta(q_o, (b, c), a) = \{(q_1, \lambda)\}$
$\delta(q_1, (b, c), a) = \{(q_1, \lambda)\}$
$\delta(q_1, (\lambda, \lambda), Z_o) = \{(q_2, Z_o)\}$

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

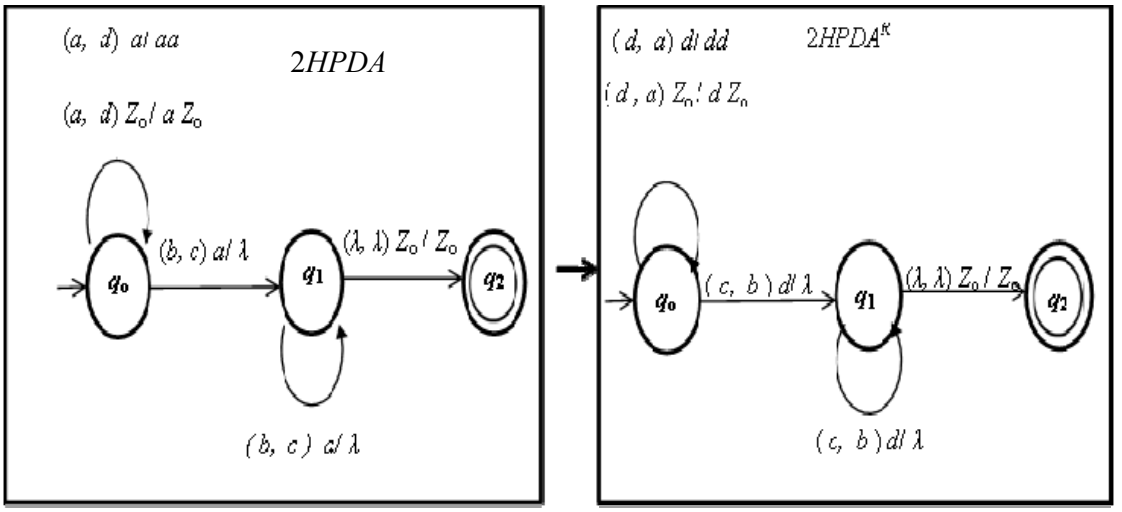

Fig.7.The Reversal of 2HPDA

3.10 Deterministic 2-Head Pushdown Automata

3.10.1 The Formal Definition of a Deterministic 2HPDA

$M = < Q, T, Z, \delta, Z_o, q_o, F >$
The 2-head pushdown automata $2HPDA_d = (Q, T, Z, q_o, Z_o, \delta, F)$ is deterministic if and only if
• For every q ∈ 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 ∈ 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 the: $\{(\text{D2HPDA})\} \subseteq \{(\text{2HPDA})\}$
Since every D2HPDA is also a 2HPDA we show that there exist a non context-free language L which can be accepted by the deterministic version of the 2HPDA.
A deterministic *2-head pushdown automata* accepts a non context free language and below is an example of a non context free language that can be accepted by a deterministic 2HPDA
$L(M) = \{a^n b^{2n} a^n \mid n \geq 0\}$
Suppose a D2HPDA = $(\{q_o, q_1, q_2, q_3\}, \{a, b, c\}, \{a, z_o\}, q_o, Z_o, \delta, \{q_o, q_3\})$
Have the following transition function:
$\delta\ (q_o,\ (a, a), Z_o) = \{(q_1, aZ_o)\}$
$\delta\ (q_1,\ (a, a), a) = \{(q_1,\ aa)\}$
$\delta\ (q_1,\ (b, b), a) = \{(q_2,\ \lambda)\}$
$\delta\ (q_2,\ (b, b), a) = \{(q_2,\ \lambda)\}$
$\delta\ (q_2,\ (\lambda, \lambda), Z_o) = \{(q_3,\ Z_o)\}$

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

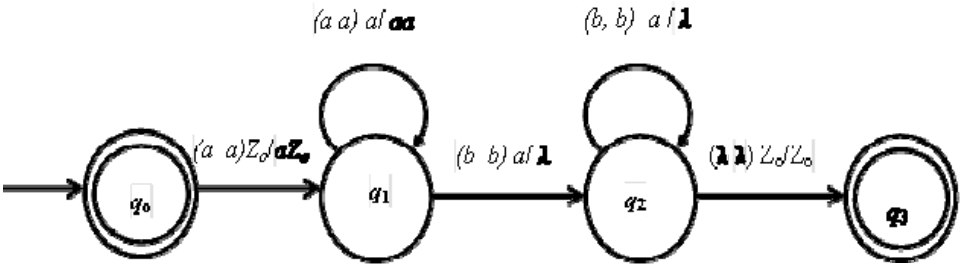

Fig.8. Deterministic 2HPDA accepting a non context free language

4. 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 modelled 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

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

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

Head, T. Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behavior. *The Bulletin of*

Mathematical Science

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

Horvath, G., Nagy, B. (2014) Formal Languages and Automata Theory, Typotex, Budapest,

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

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

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

Paun, Gh., Rozenberg, G. and Salomaa, A. (1998). *DNA Computing: New Computing Paradigms*, Springer – Verlag, Berlin.

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

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