Applied Mathematics, **2013, 4, 26-34**
http://dx.doi.org/10.4236/am.2013.410A1005 Published Online October 2013 (http://www.scirp.org/journal/am)

Equivalence of Subclasses of Two-Way Non-Deterministic Watson Crick Automata

Kumar Sankar Ray, Kingshuk Chatterjee, Debayan Ganguly
Electronics and Communication Science Unit, ISI, Kolkata, India
Email: ksray@isical.ac.in, kingshukchaterjee@gmail.com, debayan3737@gmail.com

Received June 5, 2013; revised July 5, 2013; accepted July 12, 2013

ABSTRACT

Watson Crick automata are finite automata working on double strands. Extensive research work has already been done on non deterministic Watson Crick automata and on deterministic Watson Crick automata. Parallel Communicating Watson Crick automata systems have been introduced by E. Czeziler *et al*. In this paper we discuss about a variant of Watson Crick automata known as the two-way Watson Crick automata which are more powerful than non-deterministic Watson Crick automata. We also establish the equivalence of different subclasses of two-way Watson crick automata. We further show that recursively enumerable (RE) languages can be realized by an image of generalized sequential machine (gsm) mapping of two-way Watson-Crick automata.

Keywords: Non-Deterministic Watson Crick Automata; Two-Way Non-Deterministic Watson Crick Automata; RE Languages

1. Introduction

The tremendous progress in biotechnology has resulted in decoding of DNA sequences, synthesizing and manipulating DNA, which lead to its usage in computation by computer scientists. As a result sticker systems, splicing systems and carving systems came into existence [1]. Many of the NP-complete problems were solved efficiently using DNA computing. The first, the Adleman experiment was done in 1994 [2]. 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 Watson-Crick automata [3] which are the automata counterpart of the Sticker Systems. Essentially Watson-Crick automata are finite automata having two independent heads working on double strands where the characters on the corresponding positions of the two strands are connected by a complementarity relation similar to the Watson-Crick complementarity relation.

The movement of the heads although independent of each other is controlled by a single state.

Details of several variants of non-deterministic Watson-Crick automata have been explored in [4].

Deterministic Watson-Crick automata and their variants have been explicitly handled in [5,6]. Parallel Communicating Watson-Crick automata were introduced in [7] and further investigated in [8]. A survey of Watson-Crick automata can be found in [9]. The effect of the complementarity relation on the computing power of Watson Crick automata is discussed in [10].

Two-way finite automaton (FA) is an abstract machine, a generalized version of the finite automaton which can revisit characters already processed. As in FA, in two-way FA there are finite number of states with transitions between them based on the current character; but each transition is also labeled with a value indicating whether the machine will move its reading head to the left, right, or stay at the same position. Equivalently, 2FAs can be seen as read-only Turing machines with no work tape; only a read-only input tape. The accepting condition is that when the reading head falls off the right end of the tape and the state in which the machine is at that time is final state then the input word is accepted. A two-way Watson Crick automaton (2AWK) is similar in concept to a two-way finite automaton. The only difference between them is that in two-way Watson Crick automata the input tape is double stranded. The idea of two-way Watson Crick automata were introduced in [4] but no comparison of its power with respect to AWK was discussed. The importance of 2AWK is that unlike two-way FA which is equal in power to a FA, 2AWK are more powerful than AWK which we establish in this paper.

In this paper, we give a general description of non-deterministic Watson Crick automata and its different subclasses in Sections 2 and 3. In Section 4 we describe the twin shuffle language and state the relation of twin shuffle language with RE languages. In the following section we state the rules governing two-way non-deterministic Watson Crick automata. In Section 6 we give the definition of the different classes (variants) of 2AWK and investigate the relationship between classes of 2AWK automata. We show that 2AWK = 2SWK = 21WK = 2FWK = 2FSWK = 2F1WK similar to the case of Watson Crick automata. We further show the family of languages accepted by 2AWK is context sensitive. In Section 7 we show that two-way non-deterministic Watson Crick automata are more powerful than non-deterministic Watson Crick automata. In Section 9 we further show that recursively enumerable (RE) languages can be realized by an image of generalized sequential machine (gsm) mapping of two-way Watson-Crick automata.

2. Basic Terminology for Watson-Crick Automata

V is a finite alphabet. V^* denotes the set of all finite words over V, including the empty word λ. $V^+ = V^* - \lambda$. For $w \in V^*$, $|w|$ denotes the length of *w*. Let $u \in V^*$ and $v \in V^*$ be two words and if there is some word $x \in V^*$, such that $v = ux$, then *u* is the prefix of *v*, denoted by $u \le v$. Two words, *u* and *v* are prefix comparable denoted by $u \sim_p v$ if *u* is a prefix of *v* or vice versa.

Given two alphabets *V* and *U* a mapping *h*: $V \to U^*$, extended to s: $V^* \to U^*$ by $h(\lambda) = \{\lambda\}$ and $h(x_1, x_2) = h(x_1)h(x_2)$ for $x_1, x_2 \in V^*$, is called a morphism. If $\lambda \notin h(a)$, for each $a \in V$, then *h* is a λ free morphism.

A morphism *h*: $V^* \to U^*$ is called a coding if $h(a) \in U$ for each $a \in V$ and a weak coding if $h(a) \in U \cup \{\lambda\}$ for each $a \in V$. If $h : (V_1 \cup V_2)^* \to V_1^*$ is the morphism defined by $h(a) = a$ for $a \in V_1$, and $h(a) = \lambda$ otherwise, then we say that h is a projection (associated with V_1) and we denote it by pr_{V1}.

For $x, y \in V^*$ we define their shuffle by

$$x \text{ш} y = \{x_1 y_1 \cdots x_n y_n \mid x = x_1 \cdots x_n,$$
$$y = y_1 \cdots y_n, x_i, y_i \in V^*, 1 <= i \langle = n,\ n \rangle = 1\}$$

A generalized sequential machine (gsm) is a sequential transducer. Such a device is a system $g = (Q, V_1, V_2, q_0, F, \delta)$. where Q is the set of states, V_1, V_2 are the alphabets(input and output alphabets) of the automaton, $q_0 \in Q$ is the initial state, $F \in Q$ is the set of final states and $\delta : Q \times V_1 \to P_f(V_2^* \times Q)$ is the transition mapping.

The definitions of morphism, gsm and shuffle are stated in [1].

A Watson-Crick automaton is a 6-tuple of the form $M = (V, \rho, Q, q_0, F, \delta)$ where V is an alphabet set, Q is a set of states, $\rho \subseteq V \times V$ is the complementarity relation similar to Watson Crick complementarity relation and q_0 is the initial state and $F \subseteq Q$ is the set of final states. The function δ contains a finite number of transition rules of the form $q\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to q'$, which denotes that the machine in state *q* parses w_1 in upper strand and w_2 in lower strand and goes to state q' where $w_1, w_2 \in V^*$. $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ is different from $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$. $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ is just a pair of strings written in that form instead of (w_1, w_2) whereas in $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ the two strands are of same length *i.e.* $|w_1| = |w_2|$ and the corresponding symbols in two strands are complementarity in the sense of relation ρ.

$$\begin{bmatrix} V \\ V \end{bmatrix}_\rho = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \middle| a, b \in V,\ (a, b) \in \rho \right\} \text{ and}$$

$$WK_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*.$$

A transition in a Watson-Crick finite automaton can be defined as follows:

For, $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \in \begin{pmatrix} V^* \\ V^* \end{pmatrix}$ such that

$\begin{bmatrix} x_1 u_1 w_1 \\ x_2 u_2 w_2 \end{bmatrix} \in WK_\rho(V)$ and $q,\ q' \in Q$,

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} q \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} q' \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ iff there is transition rule $q\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \to q'$ in δ.

$\overset{*}{\Rightarrow}$ denotes the transitive and reflexive closure of $\Rightarrow$.

The language accepted by Watson-Crick Automata is

$$L(M) = \left\{ w_1 \in V^* \middle| q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* q \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, \right.$$
$$\left. \text{with } q \in F, w_2 \in V^*, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V) \right\}$$

For a Watson-Crick Automaton *M*, with input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)$ where w_1 is any string in V^* and $q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} q_f$ where q_0 is the initial state and q_f is

a final state. Then $q_0\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} q_f$ is a computation in *M* denoted by σ.

Another important language associated with Watson-Crick automaton is defined taking into consideration the transitions and not the language recognised.

For a Watson-Crick Automaton $M(V, \rho, Q, q_0, F, \delta)$ consider a labeling

$e: \delta \to Lab$, of rules in δ with elements in a set Lab. For computation

$\sigma: q_0 w \Rightarrow^* w\, q_f, w \in WK_\rho(V),\ q_f \in F,$ denoted by $e(\sigma)$ the control word of σ, that is the sequence of labels of transition rules used in σ. In this way the language is obtained.

$$L_{ctr}(M) = \{e(\sigma) | \sigma: q_0 w \Rightarrow^* w q_f,\ w \in WK_\rho(V),\ q_f \in F\}.$$

The definition of $L_{ctr}(M)$ is stated in [4].

3. Subclasses of Non-Deterministic Watson-Crick Automata (AWK)

Depending on the type of states and transition rules there are four types or subclasses of Watson-Crick Automata. Watson-Crick Automaton $M = (V, \rho, Q, q_0, F, \delta)$ is

1) *stateless* (*NWK*): If it has only one state, *i.e.* $Q = F = \{q_0\}$;

2) *all-final* (*FWK*): If all the states are final, *i.e.* $Q = F$;

3) *simple* (*SWK*): If at each step the automaton reads either from the upper strand or from the lower strand, *i.e.* for any transition rule

$$q\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to q',\ \text{either } w_1 = \lambda \text{ or } w_2 = \lambda;$$

4) 1-*limlited* (1*WK*): If for any transition rule *q*

$q\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to q'$, we have $|w_1 w_2| = 1$.

Theorem 1: Simple and 1 limited Watson-Crick automata accept the same family of languages as the family of languages accepted by Watson-Crick automata with arbitrary transition rules.

The proof of Theorem 1 is in [4].

Theorem 2: Non-deterministic Watson-Crick automata are equivalent with non-deterministic simple Watson-Crick automata.

Corollary 1: Non-deterministic Watson-Crick automata are equivalent with non-deterministic 1-limited Watson-Crick automata.

The proof of Theorem 2 and Corollary 1 are given in [4].

4. Twin-Shuffle Language

Consider an alphabet *V* and its barred variant, $\bar{V} = \{\bar{a} | a \in V\}$. The language

$$TS_V = \bigcup_{x \in V^*} (x \amalg \bar{x})$$

is called the twin-shuffle language over V. (For a string $x \in V^*, \bar{x}$ denotes the string obtained by replacing each symbol in x with its barred variant).

For the morphism *h*: $(V \cup \bar{V})^* \to V^*$ defined by

$h(a) = \lambda$, for $a \in V$ and

$h(\bar{a}) = a$, for $a \in V$.

Clearly the equality is $TS_V = EQ(h, pr_V)$.

Theorem 3: Each recursively enumerable language $L \subseteq T^*$ can be written in the form $L = pr_T(TS_V \cap R)$, where V is an alphabet and R is a regular language.

In this representation, the language TS_V depends on the language *L*. This can be avoided in the following way:

Let a coding be $f: V \to \{0,1\}^*$, for instance, $f(a_i) = 01^i0$, where a_i is the i^{th} symbol of V in a specified ordering. The language $f(R)$ is regular. A generalized sequential machine (gsm) can simulate the intersection with a regular language, the projection pr_T as well as the decoding of elements in $f(TS_V)$. Thus we obtain:

Corollary 2: For each recursively enumerable language L there is a gsm g_L such that $L = g_L(TS_{\{0,1\}})$.

Therefore, by using a sequential transducer which can be a deterministic one, we can obtain all recursively enumerable language, starting from the unique language $TS_{\{0,1\}}$. Proofs of Theorem 3 and Corollary 2 are in [1].

5. Two-Way Non-Deterministic Watson Crick Automata (2AWK)

Two-way non-deterministic Watson Crick automata system is a 6 tuple, $M = (V \cup \{\#, \$\}, \rho, Q, q_0, F, \delta)$ where V is a set of alphabet, $\#, \$ \notin V$ are the beginning and the end marker respectively; that is the word *w* to be recognized is provided as an input to the automaton in the form $\#w\$$. Q is a set of states, $\rho_1 \subseteq V \times V$ and $\rho_2 = \{(\#, \#), (\$, \$)\}$ and $\rho = \rho_1 \cup \rho_2$ is the complementarity relation and q_0 is the initial state and $F \subseteq Q$ is the set of final states. δ is the finite number of transition rules;

1) either of the form $q\begin{pmatrix} w_1, dir_1 \\ w_2, dir_2 \end{pmatrix} \to q'$, which denotes that the machine in state *q* parses w_1 in upper strand in dir_1 direction and w_2 in lower strand in dir_2 direction and goes to state q' where $w_1, w_2 \in V^*, dir_1, dir_2 \in \{L, R, 0\}$ where *L* signifies that the head is reading the word in the left direction, *R* signifies that the head is reading the word in right direction and if a head reads the empty word λ it remains in its

current position denoted by 0.

2) or of the forms $q\begin{pmatrix} x_1, dir_1 \\ x_2, dir_2 \end{pmatrix} \to q'$, where

$x_1, x_2 \in \{\#, \lambda\}$ and $dir_1, dir_2 \in \{L, R, 0\}$ with restrictions that when x_1 or $x_2 = \lambda$ the corresponding dir_1 or $dir_2 = L$ and when x_1 or $x_2 = \lambda$ the corresponding dir_1 or $dir_2 = 0$. Moreover, there cannot be transition rules having the form $q\begin{pmatrix} w_1, dir_1 \\ w_2, dir_2 \end{pmatrix} \to q'$ where $w_1 = w_1'\#$ where $w_1' = V^*$ and $dir_1 = L$ or $w_2 = w_2'\#$ where $w_2' = V^*$ and $dir_2 = L$ or both. These rules ensure that the reading heads do not go past the input word on the left side or the heads do not move when it reads empty word. Moreover once a head goes past the right end of the tape it cannot comeback.

Accepting conditions

W_1 is accepted by M if, starting in state q_0 (initial state) with $\begin{bmatrix} \#w_1\$ \\ \#w_2\$ \end{bmatrix}$ and $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)$ on the double stranded input tape and the two heads at the left end of $\#w_1\$$ and $\#w_2\$$ respectively, M eventually enters a final state at the same time both the heads fall off the right hand side of the double stranded input tape.

The word w_1 is rejected if one of the following 3 conditions occurs:

1) The two-way WK automaton goes into a loop which is identified in a similar way as loops in two-way FAs are identified.

2) When both the heads fall off the right hand side of the input tape and the machine is in a non final state.

3) If the machine comes to a halt (*i.e.* there are no transition rules that can be applied for that particular state in which the machine is) before the heads fall off the right hand side of the input tape.

i.e. mathematically

$$L(M) = \left\{ w_1 \in V^* \middle| q_0 \begin{bmatrix} \#w_1\$ \\ \#w_2\$ \end{bmatrix} \overset{*}{\Rightarrow} q \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, \text{ with } q \in F, w_2 \in V^*, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V) \right\}.$$

6. Subclasses of Two-Way Non-Deterministic Watson-Crick Automata (2AWK)

Depending on the type of states and transition rules there are four types or subclasses of two-way Watson-Crick Automata similar to Watson Crick automata.

2-way Watson-Crick Automaton $M = (V, \rho, Q, q_0, F, \delta)$ is

1) *stateless* (2*NWK*): If it has only one state, *i.e.* $Q = F = \{q_0\}$;

2) *all-final* (2*FWK*): If all the states are final, *i.e.* $Q = F$;

3) *simple* (2*SWK*): If at each step the automaton reads either from the upper strand or from the lower strand, *i.e.* for any transition rule $q\begin{pmatrix} w_1, dir_1 \\ w_2, dir_2 \end{pmatrix} \to q'$ either $w_1 = \lambda$ or $w_2 = \lambda$;

4) 1-*limlited* (21*WK*): If for any transition rule $q\begin{pmatrix} w_1, dir_1 \\ w_2, dir_2 \end{pmatrix} \to q'$, we have $|w_1 w_2| = 1$.

Many combinations of these classes can also be obtained such as all-final simple two-way WK automata (2FSWK), all final 1 limited two-way WK automata (21FWK), stateless 1 limited two-way WK automata (21NWK) etc.

Theorem 4: Simple and 1 limited two-way Watson Crick automata accept the same family of languages as the family of languages accepted by two-way Watson Crick automata with arbitrary transition rules.

The proof of theorem is similar to the proof done in [4] for Theorem 1.

Let $M = (V, \rho, Q, q_0, F, \delta)$ be a non-deterministic two-way Watson Crick automaton. We introduce a 1 limited two-way Watson Crick automaton $M' = (V, \rho, Q', q_0, F, \delta')$. For each transition rule *t* of the form $q\begin{pmatrix} w_1, dir_1 \\ w_2, dir_2 \end{pmatrix} \to q'$ in δ where $w_1 = a_1 a_2 \cdots a_n$ where $|w_1| = n$ and $w_2 = b_1 b_2 \cdots b_m$ where $|w_2| = m$ and $n + m \geq 2$, the condition $n + m \geq 2$ is imposed because rules with $m + n < 2$ is already in the 1-limited form and no further modification is required for them. We introduce new rules in δ' of the form

$$q\begin{pmatrix} a_1, dir_1 \\ \lambda, 0 \end{pmatrix} \to q_{t,1}$$

$$q_{t,i}\begin{pmatrix} a_{i+1}, dir_1 \\ \lambda, 0 \end{pmatrix} \to q_{t,i+1},\ 1 \leq i \leq n-1,$$

$$q_{t,n}\begin{pmatrix} \lambda, 0 \\ b_1, dir_2 \end{pmatrix} \to q'_{t,1}$$

$$q'_{t,j}\begin{pmatrix} \lambda, 0 \\ b_{j+1}, dir_2 \end{pmatrix} \to q'_{t,j+1},\ 1 \leq j \leq m-2,$$

$$q'_{t,m-1}\begin{pmatrix} \lambda, 0 \\ b_m, dir_2 \end{pmatrix} \to q'.$$

All the new states are introduced in *Q*' along with states in *Q*. From the construction of M' which is obtained from M it is obvious that both M' and M recognize the same language. So 2AWK are subset of 21WK and from the definition of 21WK and AWK we know

that 21WK are subset of 2AWK. So 2AWK and 21WK are equivalent *i.e.* they accept the same family of languages. A similar proof can also be established for 2SWK. Therefore we can say, 2AWK = 2SWK = 21WK.

Theorem 5: All final two-way Watson Crick automata accept the same family of languages as the family of languages accepted by two-way Watson Crick automata with arbitrary transition rules.

Let $M=(V,\rho,Q,q_0,F,\delta)$ be a two-way non-deterministic Watson Crick automaton. We introduce an all final two-way Watson Crick automaton $M'=(V,\rho,Q',q_0,\delta')$. Each transition rule *t* of the form $q\begin{pmatrix}w_1,dir_1\\w_2,dir_2\end{pmatrix}\to q'$ in δ where $w_1=a_1a_2\cdots a_n$ where $|w_1|=n$ and $w_2=b_1b_2\cdots b_m$ where $|w_2|=m$ falls under one of the five classes. The classes are defined as follows:

Class 1: Transition rules of the form $q\begin{pmatrix}w_1,dir_1\\w_2,dir_2\end{pmatrix}\to q'$ in δ where $w_1=a_1a_2\cdots a_n$ where $|w_1|=n$ and $w_2=b_1b_2\cdots b_m$ where $|w_2|=m$ and a_n and $b_n\neq\$$, *i.e.* w_1 and w_2 do not have \$ at their ends.

Class 2: Transition rules of the form $q\begin{pmatrix}w_1,R\\w_2,R\end{pmatrix}\to q'$ in δ where $w_1=a_1a_2\cdots a_n$ where $|w_1|=n$ and $w_2=b_1b_2\cdots b_m$ where $|w_2|=m$, and $a_n,b_n=\$$, *i.e.* w_1 and w_2 both have \$ at their ends.

Class 3: Transition rules of the form $q\begin{pmatrix}w_1,R\\w_2,dir_2\end{pmatrix}\to q'$ in δ where $w_1=a_1a_2\cdots a_n$ where $|w_1|=n$ and $w_2=b_1b_2\cdots b_m$ where $|w_2|=m, a_n=\$$ and $b_n\neq\$$ *i.e.* w_1 has \$ at its end and w_2 does not have \$ at its end.

Class 4: Transition rules of the form $q\begin{pmatrix}w_1,dir_1\\w_2,R\end{pmatrix}\to q'$ in δ where $w_1=a_1a_2\cdots a_n$ where $|w_1|=n$ and $w_2=b_1b_2\cdots b_m$ where $|w_2|=m, a_n\neq\$$ and $b_n=\$$ *i.e.* w_1 does not have \$ at its end and w_2 has \$ at its end.

Class 5: Either transition rules of the form $q\begin{pmatrix}\$,L\\\lambda,0\end{pmatrix}\to q'$ in δ or transition rules of the form $q\begin{pmatrix}\$,L\\\$,L\end{pmatrix}\to q'$ in δ or transition rules of the form $q\begin{pmatrix}\lambda,0\\\$,L\end{pmatrix}\to q'$ in δ.

The transition rules of M are modified as follows to form the transition rules of M'.

Transition rules of M which fall in class 1 and class 5 are kept same in M'.

For transition rules of M which belong to class 2 two instances can occur;

case 1: For transition $q\begin{pmatrix}w_1,R\\w_2,R\end{pmatrix}\to q'$, where q' is a final state. In this case the transition rules are kept same in M'.

case 2: For transition $q\begin{pmatrix}w_1,R\\w_2,R\end{pmatrix}\to q'$, where q' is a non final state. In this case the transition rules of M are modified as follows for M'.

For each transition rule $q\begin{pmatrix}w_1,R\\w_2,R\end{pmatrix}\to q'$ in M belonging to class 2 where q' is a non final state, $q\begin{pmatrix}w_1',R\\w_2',R\end{pmatrix}\to q''$ where $w_1=w_1'\$$ and $w_2=w_2'\$$ are introduced in M' and there is no transition from q'' in M'. These new rules in M' ensure that if the heads go off the right end of the tape in M when M is in a non final state then M' would go to state q'' and would not accept the string as there is no transition from q'' *i.e.* the above stated rules ensure the heads do not fall off the right end of the tape for M' when M does not accept the word. As M' is all final if the heads go off the right end of the tape it will accept the given string.

For transition rules of M which belong to class 3 the following modifications are needed. Class 3 also has two instances similar to class 2.

case 1: For transition $q\begin{pmatrix}w_1,R\\w_2,dir_2\end{pmatrix}\to q'$, where q' is a final state. In this case the transition rules are kept same in M'.

case 2: For transition $q\begin{pmatrix}w_1,R\\w_2,dir_2\end{pmatrix}\to q'$, where q' is a non final state. In this case the transition rules of M are modified as follows for M'.

For each transition rule $q\begin{pmatrix}w_1,R\\w_2,dir_2\end{pmatrix}\to q'$ in M belonging to class 3 where q' is a non final state, $q\begin{pmatrix}w_1',R\\w_2,dir_2\end{pmatrix}\to q'_{u\$}$ where $w_1=w_1'\$$ is introduced in M' where $q'_{u\$}$ denotes that the head on the upper strand has gone past the right end marker \$ in the original machine M on application of the above transition rule.

Only rules having λ on the upper strand are applied to $q'_{u\$}$ because in the actual machine M if the above rules of class 3 are applied then the upper head would have gone past the right end of the tape. So only rules

having λ on the upper head can be applied to the machine M. As M' replicates M similar thing is done in M' too.

Thus, all the transition rules that can be applied to q' in M with λ on the upper strand and $w_2 = a_1 a_2 \cdots a_n$ and $a_n \neq \$$ in the lower strand can also be applied to $q'_{u\$}$ in M'. Rules having λ on the upper strand and $w_2 = a_1 a_2 \cdots a_n$ and $a_n = \$$ in the lower strand where the transition goes to a final state are applied to $q'_{u\$}$. Finally for rules with λ on the upper strand and $w_2 = a_1 a_2 \cdots a_n$ and $a_n = \$$ in the lower strand where the transition goes to a non final state, the rules of the form $q'_{u\$}\begin{pmatrix}\lambda,0\\ w'_2,R\end{pmatrix} \to q_{ul\$}$ where $w_2 = w'_2\$$ are introduced in M' and there are no transition rules from $q_{ul\$}$. These rules ensure that when *M* reaches the end of the string on a non final state then M' goes to $q_{ul\$}$. and M' does not accept the string as there is no transition from $q_{ul\$}$. *i.e.* the above stated rules ensure the heads do not fall off the right end of the tape for M' when heads off *M* fall off the right end and the state to which *M* goes is non final.

Class 4 rules are handled in a similar way to class 3 rules.

It is obvious from the transition rules introduced in M' that M' accepts the same family of languages as M.

Thus, 2FWK = 2AWK.

Theorem 6: All final 1 limited two-way Watson Crick automata accept the same family of languages as the family of languages accepted by 1 limited two-way Watson Crick automata with arbitrary transition rules.

Let $M = (V, \rho, Q, q_0, F, \delta)$ be a two-way 1 limited non-deterministic Watson Crick automaton. We introduce an all final 1 limited two-way Watson Crick automaton $M' = (V, \rho, Q', q_0, \delta')$. Each transition rule t of the form $q\begin{pmatrix}w_1, dir_1\\ w_2, dir_2\end{pmatrix} \to q'$ in δ where $|w_1 w_2| = 1$ falls under one of the four classes. The classes are defined as follows:

Class 1: Transition rules of the form $q\begin{pmatrix}w_1, dir_1\\ w_2, dir_2\end{pmatrix} \to q'$ in δ where $w_1 \neq \$$ and $w_2 \neq \$$.

Class 2: Transition rules of the form $q\begin{pmatrix}\$, R\\ \lambda, 0\end{pmatrix} \to q'$ in δ.

Class 3: Transition rules of the form $q\begin{pmatrix}\lambda, 0\\ \$, R\end{pmatrix} \to q'$ in δ.

Class 4: Transition rules of the form $q\begin{pmatrix}\$, R\\ \lambda, 0\end{pmatrix} \to q'$ in δ or transition rules of the form $q\begin{pmatrix}\lambda, 0\\ \$, R\end{pmatrix} \to q'$ in δ

The transition rules of M are modified as follows to form the transition rules of M'.

Transition rules of M which fall in class 1 and class 4 are kept same in M'.

For transition rules of M which belong to class 2 have two instances.

case 1: For transition $q\begin{pmatrix}\$, R\\ \lambda, 0\end{pmatrix} \to q'$, where q' is a final state. In this case the transition rules are kept same in M'.

case 2: For transition $q\begin{pmatrix}\$, R\\ \lambda, 0\end{pmatrix} \to q'$, where q' is a non final state. In this case the transition rules of M are modified as follows for M'.

For each transition rule $q\begin{pmatrix}\$, R\\ \lambda, 0\end{pmatrix} \to q'$ in M belonging to class 2 where q' is a non final state, $q\begin{pmatrix}\$, L\\ \lambda, 0\end{pmatrix} \to q'_{bu\$}$ and $q'_{bul\$}\begin{pmatrix}x, R\\ \lambda, 0\end{pmatrix} \to q'_{u\$}$ where $x \in V$ are introduced in M'. $q'_{u\$}$ denotes that the head on the upper strand has gone past the right end marker \$ in the original machine M.

Only rules having λ on the upper strand can be applied to $q'_{u\$}$ (for reasons similar to reasons stated in proof of Theorem 5). Thus, all the transition rules that can be applied to q' with λ on the upper strand and $w_2 \neq \$$ in the lower strand are applied to $q'_{u\$}$. For rules having λ on the upper strand and $w_2 = \$$ in the lower strand where the transition goes to a final state are applied to $q'_{u\$}$. Finally for rules with λ on the upper strand and $w_2 = \$$ in the lower strand where the transition goes to a non final state in M, the rules of the form $q'_{u\$}\begin{pmatrix}\lambda, 0\\ \$, L\end{pmatrix} \to q'b_{ul\$}$ and $q'_{bul\$}\begin{pmatrix}\lambda, 0\\ x, R\end{pmatrix} \to q'_{ul\$}$, where $x \in V$ are introduced in M'. These rules ensure that when M reaches the end of the string on a non final state, M' does not accept the string as there are no transitions from state $q'_{ul\$}$.

Class 3 is handled in a similar way as class 2.

It is obvious from the transition rules introduced in M' that M' accepts the same family of languages as M.

Thus, 21FWK = 21WK.

Corollary 3: All final 1 limited two-way Watson Crick automata accept the same family of languages as the family of languages accepted by arbitrary two-

way Watson Crick automata with arbitrary transition rules.

Proof: From Theorem 4 we know 2AWK = 21WK and from Theorem 6 we obtain 21FWK = 21WK. Thus combining both the results we get 21FWK = AWK.

Thus from the above Theorems we can state that 2AWK = 21FWK = 21WK = 2SWK = 2FSWK = 2FWK.

7. Power of Two-Way Non-Deterministic WK Automata

In this section we first show that AWK are subset of 2AWK. Then we further show that this subset relation is proper *i.e.* 2AWK are more powerful than AWK.

Theorem 7: AWK $\subseteq$ 2AWK.

The theorem says that non-deterministic Watson Crick automata are subset of two-way non-deterministic Watson Crick automata.

Proof:

Let $M=(V,\rho,Q,q_0,F,\delta)$ be a non-deterministic Watson Crick automaton where V is a set of alphabet, Q is a set of states, $\rho \subseteq V \times V$ is the complementarity relation and q_0 is the initial state and $F \subseteq Q$ is the set of final states. δ is the finite number of transition rules of the form $q\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to q'$, where $w_1, w_2 \in V^*$.

We introduce a two-way non-deterministic Watson Crick automaton $M'=\left(V \cup \{\#,\$\}, \rho', Q', q', q_f, \delta'\right)$ where V is a set of alphabet, $\#,\$ \notin V$ are the beginning and the end marker respectively, that is, the word w to be recognized is provided as an input to the automaton in the form $\#w\$$. $Q' = Q \cup \{q_0', q_f\}$, $\rho' = \rho \cup \{(\#,\#),(\$,\$)\}$ is the complementarity relation and q_0' is the initial state and q_f is a final state. δ' is the finite number of transition rules of the form

1) For each rule $q\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to q'$ in δ introduce $q\begin{pmatrix} w_1, R \\ w_2, R \end{pmatrix} \to q'$ in δ'.

2) $q_0'\begin{pmatrix} \# \\ \# \end{pmatrix} \to q_0$.

3) For each state $x \in F$ in M introduce $x\begin{pmatrix} \$, R \\ \$, R \end{pmatrix} \to q_f$ in δ'.

From the construction of M' it is evident that all that will be accepted by M will be accepted by M'.

Theorem 8: One-Way Two headed finite automata are equivalent to AWK

An informal proof of this theorem is in [4].

Example 1

Let $M=\left(V \cup \{\#,\$\}, \rho, Q, q_0, F, \delta\right)$ be a 2AWK where $V=\{a,b\}, \#,\$ \notin V$ are the beginning and the end marker respectively, that is, the word w to be recognized is provided as an input to the automaton in the form $\#w\$$. Q is a set of states, $Q=\{q_0, q_1, q_2, q_3, q_4\}$, ρ is the identity complementarity relation and q_0 is the initial state and $F=\{q_4\}$ is the set of final states. δ is the finite number of transition rules. In this example the mirror language $L=\{w \mid w \in \{a,b\}^*$ and $L=\left\{w \middle| w \in \{a,b\}^* \text{ and } w=w^R\right\}$ where w^R denotes the reverse of w is accepted using two-way Watson Crick automaton.

The transition rules of M are as follow

$$q_0\begin{pmatrix} \#, R \\ \#, R \end{pmatrix} \to q_1,$$

$$q_1\begin{pmatrix} x, R \\ \lambda, 0 \end{pmatrix} \to q_1,\ x \in V,$$

$$q_1\begin{pmatrix} \$, L \\ \lambda, 0 \end{pmatrix} \to q_2,$$

$$q_2\begin{pmatrix} x, L \\ x, R \end{pmatrix} \to q_2,\ x \in V,$$

$$q_2\begin{pmatrix} \#, R \\ \$, L \end{pmatrix} \to q_3,$$

$$q_3\begin{pmatrix} x, R \\ \lambda, 0 \end{pmatrix} \to q_3,\ x \in V,$$

$$q_3\begin{pmatrix} \$, R \\ \$, R \end{pmatrix} \to q_4.$$

Theorem 9: One-way finite automata with 2 heads cannot accept the mirror language.

The above theorem is stated in [11].

Theorem 10: 2AWK are more powerful than AWK *i.e.* AWK $\subset$ 2AWK.

Proof. From Theorem 8 we know that AWK is equivalent to 1-way two headed finite automata and from Theorem 9 we know that 1-way two headed finite automata cannot recognize the mirror language. Thus AWK cannot recognize the mirror language. But in Example 1 we have shown that two-way AWK can accept the mirror language and in theorem 7 we have shown that AWK $\subseteq$ 2AWK *i.e.* 2AWK accepts all the family of languages which are accepted by AWK. Moreover it also accepts the mirror language which AWK cannot accept. Thus 2AWK accepts at least one language more than AWK. Hence we conclude that the accepting power of two-way AWK is more than AWK. Mathematically AWK $\subset$ 2AWK, *i.e.* the subset relation is proper.

Theorem 11: Family of languages accepted by WK automata is context sensitive.

A linear bounded Turing machine (LBA) can simulate the actions of two-way Watson Crick automaton. As the language accepted by LBA is context sensitive so the

family of languages accepted by two-way Watson Crick automaton is also context sensitive.

8. Characterization of Recursively Enumerable (RE) Languages in Terms of 2AWK Automata

In this section we discuss 2AWK in the light of the RE languages. We show each language in the family of RE is the image of a gsm mapping of a language in 2 AWK.

Theorem 12: $TS_V \in AWK(ctrl)$

The proof of this theorem is in [4].

Theorem 13: For each recursively enumerable language L there is a gsm g_L such that $L = g_L$ (2AWK (ctrl)).

Proof: We have already shown in Theorem 7 AWK $\in$ 2AWK and it is stated in [4] that $TS_V \in$ AWK(ctrl) and from corollary 2 we know that each language in the family of RE is the image of a gsm mapping of a language in $TS_{\{0,1\}}$. As $TS_V \in$ AWK(ctrl) and AWK $\in$ 2AWK, so we can state, each language in the family of RE is the image of a gsm mapping of a language in 2AWK(ctrl).

9. Conclusion

In this paper, we discuss about the power of a variant of non-deterministic Watson Crick automata known as 2AWK. We describe their structure and accepting conditions. We introduce different subclasses of 2AWK similar to AWK and show the equivalence of some of those subclasses. We further establish the fact that 2AWK are more powerful than AWK. Based on the relation between AWK and 2AWK we show that a gsm mapping of 2AWK results in the generation of each language in the family of the recursively enumerable languages.

REFERENCES

[1] L. C. S. Calude and G. Paun, "Computing with Cells and Atoms: An Introduction to Quantum, DNA and Membrane Computing," Taylor & Francis Publishers, London, 2001.

[2] L. M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, *New Series*, Vol. 226, No. 5187, 1994, pp. 1021-1024. http://dx.doi.org/10.1126/science.7973651

[3] R. Freund, G. Paun, G. Rozenberg and A. Saloma, "A, Watson-Crick Finite Automata," *Proceedings of the* 3*rd DIMACS Workshop on DNA Based Computers*, Philadelphia, 1997, pp. 297-328.

[4] G. Paun, G. Rozenberg and A. Salomaa, "DNA Computing: New Computing Paradigms," Springer-Verlag, Berlin, 1998.

[5] E. Czeizler, E. Czeizler, L. Kari and K Salomaa, "Watson-Crick Automata: Determinism and State Complexity," *Proceeding of*: 10*th International Workshop on Descriptional Complexity of Formal Systems*, *DCFS*, 16-18 July 2008, pp. 121-133.

[6] E. Czeizler, E. Czeizler, L. Kari and K. Salomaa, "On the Descriptional Complexity of Watson-Crick Automata," *Theoretical Computer Science*, Vol. 410, No. 35, 2009, pp. 3250-3260. http://dx.doi.org/10.1016/j.tcs.2009.05.001

[7] E. Czeizler, E. Czeizler," Parallel Communicating Watson-Crick Automata Systems," In: Z. Fulop Z. Esik (Ed.), Proceedings of 11th International Conference, AFL 2005, 2005, pp. 83-96.

[8] E. Czeizler, "On the Power of Parallel Communicating Watson-Crick Automata Systems," *Theoretical Computer Science*, Vol. 358, No. 1, 2006, pp. 142-147.

[9] E. Czeizler, "A Short Survey on Watson-Crick Automata," *Bulletin of the EATCS*, Vol. 88, 2006, pp. 104-119.

[10] D. Kuske and P. Weigel, "The Role of the Complementarity Relation in Watson-Crick Automata and Sticker Systems," *Developments in Language Theory*, Vol. 3340, Lecture Notes in Computer Science, Springer, Berlin, 2004, pp. 272-283.

[11] M. Holzer, M. Kutrib and A. Malcher, "Multi-Head Finite Automata: Characterizations, Concepts and Open Problems," *Proceedings International Workshop on the Complexity of Simple Programs*, Cork, 6-7 December 2008, pp. 93-107.

Abbreviations

AWK: non-deterministic Watson-Crick automata.

NWK: stateless non-deterministic Watson-Crick automata.

FWK: all final non-deterministic Watson-Crick automata.

SWK: simple non-deterministic Watson-Crick automata.

1WK: 1-limited non-deterministic Watson-Crick automata.

2AWK: two way non-deterministic Watson-Crick automata.

2NWK: two way stateless non-deterministic Watson-Crick automata.

2FWK: two way all final non-deterministic Watson-Crick automata.

2SWK: two way simple non-deterministic Watson-Crick automata.

21WK: two way 1-limited non-deterministic Watson-Crick automata.

TS_V: twin-shuffle language.

RE: recursive enumerable.

gsm: generalized sequential machine.

