Title: Reversible Watson-Crick Automata

Authors: Kingshuk Chatterjee\(^1\), Kumar Sankar Ray(corresponding author)\(^2\)

Affiliations: \(^{1,2}\)Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata-108

Address: Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata-108

Telephone Number: +918981074174

Fax Number: 033-25776680

Email: ksr@isical.ac.in
Reversible Watson-Crick Automata

Kingshuk Chatterjee¹, Kumar Sankar Ray²

Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata-108.
¹kingshukchatterjee@gmail.com²ksray@isical.ac.in

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. In this paper, we introduce a new model of Watson-Crick automata which is reversible in nature named reversible Watson-Crick automata and explore its computational power. We show even though the model is reversible and one way it accepts all regular languages and also analyze the state complexity of the above stated model with respect to non-deterministic block automata and non-deterministic finite automata and establish its superiority. We further explore the relation of the reversible model with twin-shuffle language and recursively enumerable languages.

Keywords: non-deterministic Watson-Crick automata, deterministic Watson-Crick automata, reversible finite automata, reversible multihead finite automata, reversible Watson-Crick automata, state complexity.

I. INTRODUCTION

The first study on reversible automata was done in the seventies [1], when the concept of reversible Turing machine was introduced and it was shown that for every Turing machine that accepts a language there is a reversible Turing machine that accepts the same language i.e. the computational power of reversible Turing machine is same as that of the Turing machine. This result encouraged researchers to explore the reversibility property in more restricted automata and in this endeavor Morita [2] explored the concept of two way multihead reversible automata and showed that it has the same computational power as two way deterministic multihead automata. Kutrib et al. explored the computational power of one way multihead reversible automata [3] and reversible pushdown automata [4]. Pin et al. applied algebraic theory to reversible automata to explore its computability [5]. The interest in reversible automata comes from the fact that we are interested in information preserving machines as it helps us to better analyze the way an automaton behaves. Moreover also the physical observation that a loss of information results in heat dissipation [1] increases the interest in lossless reversible automata.

Watson-Crick automata [6] 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 (AWK) have been explored in [7]. Deterministic Watson-Crick automata and their variants have been explicitly handled in [8]. Equivalence of subclasses of two-way Watson-Crick automata is discussed in [9]. A survey of Watson-Crick automata can be found in [10]. Research work regarding state complexity of Watson-Crick automata is reported in [11] and [12].

Work on reversibility of Watson-Crick automata has been done by Sempere [13]. He defined different kinds of regular reversibility in the model. Mainly, they explored regular reversibility in the upper (lower) strand and in the double strand. No formal reversible model for Watson-Crick automata was introduced by Sempere nor was the computational and state complexity aspects thoroughly discussed. In this paper we introduce the reversible model of Watson-Crick automata and explore in details its computational and state complexity. We show that unlike one way reversible finite automata, deterministic reversible Watson-Crick automata can accept all the regular languages along with some non-regular languages. We also show that similar to non-deterministic Watson-Crick automata; reversible Watson-Crick automata have the same state complexity advantages over non-deterministic finite automata, even though it is deterministic and reversible. We also compare the computational powers of variants of reversible Watson-Crick automata.

The paper is arranged in the following manner. In Section 2, 3 we introduce the preliminary concepts of Watson-Crick automata and its various subclasses. In Section 4, definitions regarding deterministic variants of Watson-Crick automata are stated. In Section 5, we introduce our new model of reversible Watson-Crick automata along with its variants and subclasses. In the following section, the computational complexity of reversible Watson-Crick automata is discussed. In Section 7, complexity measures are defined. In the following section, we explore the state complexity of reversible Watson-Crick automata and establish its superiority over non-deterministic finite automata and block automata. In Section 9, we conclude our work.

II. BASIC TERMINOLOGY

The symbol V denotes a finite alphabet. The set of all finite words over V is denoted by $V^\ast$, which includes the empty word $\lambda$. The symbol $V^+ = V^\ast - \{\lambda\}$ denotes the set of all non-empty words over the alphabet V. For $w \in V^+$, the length of w is denoted by $|w|$. 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 a prefix of v, denoted by $u \preceq v$. Two words, u and v are prefix comparable denoted by $u \sim v$ if u is a prefix of v or vice versa.

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, set of states is denoted by $Q$, $\rho \subseteq V \times V$ is the complementarity relation similar to Watson-Crick complementarity relation, $q_0$ is the initial state and $F \subseteq Q$
is the set of final states. The function $\delta$ contains a finite number of transition rules of the form $q\left(\frac{w_1}{w_2}\right)\rightarrow q'$, which denotes that the machine in state $q$ parses $w_1$ in the upper strand and $w_2$ in the lower strand and goes to state $q'$ where $w_1$, $w_2\in V^*$. The symbol $\left(\frac{w_1}{w_2}\right)$ is different from $\left(\frac{w_3}{w_4}\right)$. While $\left(\frac{w_5}{w_6}\right)$ is just a pair of strings written in that form instead of $(w_1, w_2)$, the symbol $\left[\frac{w_1}{w_2}\right]$ denotes that the two strands are of the same length i.e., $|w_1|=|w_2|$ and the corresponding symbols in two strands are complementarity in the sense given by the relation $\rho$. The symbol $\left[\frac{V}{V}\right]_\rho = \{ [\alpha, \beta] \mid \alpha, \beta \in V, (\alpha, \beta) \in \rho \}$ and $\text{WK}_\rho (V) = \left[\frac{V}{V}\right]_\rho$ denotes the Watson-Crick domain associated with $V$ and $\rho$.

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

For $\left(\frac{x_1}{x_2}\right)\left(\frac{u_1}{u_2}\right)\left(\frac{w_1}{w_2}\right)\left(\frac{v_1}{v_2}\right)\left(\frac{q}{q'}\right)\left(\frac{w_3}{w_4}\right)\left(\frac{v''}{v'}\right)\in \left(\frac{V}{V}\right)$ such that $\left(\frac{x_1u_1w_1}{x_2u_2w_2}\right)\in \text{WK}_\rho (V)$ and $q, q' \in Q$, $\left(\frac{x_1}{x_2}\right)\left(\frac{u_1}{u_2}\right)\left(\frac{w_1}{w_2}\right) \Rightarrow \left(\frac{x_1}{x_2}\right)\left(\frac{u_1}{u_2}\right) q' \left(\frac{w_4}{w_5}\right)$ if there is a transition rule $q\left(\frac{u_1}{u_2}\right)\rightarrow q'$ in $\delta$ and $\Rightarrow$ denotes the transitive and reflexive closure of $\Rightarrow$. The language accepted by a Watson-Crick automaton $M$ is $L(M) = \{ w_1 \in V' | q_0 \left(\frac{w_1}{w_2}\right) \Rightarrow q \left(\frac{[q]}{q}\right), \text{with } q \in F, w_2 \in V^*, \left[\frac{w_3}{w_4}\right] \in \text{WK}_\rho (V) \}$.

### III. SUBCLASSES OF NON-DETERMINISTIC WATSON-CRICK AUTOMATA

Depending on the type of states and transition rules there are four types or subclasses of Watson-Crick automata. A non-deterministic Watson-Crick automaton $M=\left(V, p, q_0, Q, F, \delta\right)$ is:

1) stateless (NWK): If it has only one state, i.e. $Q=\{ q_0 \}$;
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 $q\left(\frac{w_1}{w_2}\right)\rightarrow q'$, either $w_1=\lambda$ or $w_2=\lambda$;
4) 1-limited (1-limited WK): If for any transition rule $q\left(\frac{w_1}{w_2}\right)\rightarrow q'$, we have $|w_1|=|w_2|=1$.

### IV. DETERMINISTIC WATSON-CRICK AUTOMATA AND THEIR SUBCLASSES

The notion of determinism in Watson-Crick automata and a discussion on its complexity were first considered in [8]. In [8] different notions of determinism were suggested as follows:

1) weakly deterministic Watson-Crick automata (WDWK): Watson-Crick automaton is weakly deterministic if in every configuration that can occur in some computation of the automaton, there is a unique possibility to continue the computation, i.e. at every step of the automaton there is at most one way to carry on the computation.
2) deterministic Watson-Crick automata (DWK): deterministic Watson-Crick automaton is Watson-Crick automaton for which if there are two transition rules of the form $q\left(\frac{w_1}{w_2}\right)\rightarrow q'$ and $q\left(\frac{w_1'}{w_2'}\right)\rightarrow q''$ then $w_1w_1'=v_1v_1'$ or $v_1v_1'=w_2w_2'$.
3) strongly deterministic Watson-Crick automata (SDWK): strongly deterministic Watson-Crick automaton is a deterministic Watson-Crick automaton where the Watson-Crick complementarity relation is injective.

Similar to non-deterministic Watson-Crick automata, deterministic Watson-Crick automata can be stateless (NDWK), all final (FDWK), simple (SiDWK) and 1-limited (1-limited DWK).

### V. REVERSIBLE WATSON-CRICK AUTOMATA

In this section, we introduce and describe the model of reversible Watson-Crick automata along with its variants and subclasses. Reversible Watson-Crick automata are defined using the concept of deterministic Watson-Crick automata in the same manner as reversible finite automata are described in [3,5] using the concept of deterministic finite automata, i.e. given a Watson-Crick automaton where the transitions obey the constraints imposed on transitions in the definition of deterministic Watson-Crick automaton and when the transitions are reversed the new automaton so formed is deterministic in nature in terms of the transitions involved with no restrictions on the sets of initial and final states then we say that the given Watson-Crick automaton is reversible. Similar to the different notions of determinism in [8], here we define different notions of reversibility as follows:

1) Weakly deterministic reversible Watson-Crick automaton (WDRWK): Given a Watson-Crick automaton where the transitions obey the constraints imposed on transitions in the definition of weakly deterministic Watson-Crick automaton and after reversal of transitions the new automaton is weakly deterministic in terms of the restrictions on the transitions and the complementarity relation then the given automaton is a weakly deterministic
reversible Watson-Crick automaton.

2) Deterministic reversible Watson-Crick automata (DRWK): Given a Watson-Crick automaton where the transitions obey the constraints imposed on transitions in the definition of deterministic Watson-Crick automaton and after reversal of transitions the new automaton is deterministic in terms of the restrictions imposed on the transitions and the complementarity relation then the given automaton is a deterministic reversible Watson-Crick automaton.

3) Strongly deterministic reversible Watson-Crick automata (SDRWK): Given a Watson-Crick automaton where the transitions obey the constraints imposed on transitions in the definition of strongly deterministic Watson-Crick automaton and after reversal of transitions the new automaton is strongly deterministic in terms of the restrictions imposed on the transitions and the complementarity relation then the given automaton is a deterministic reversible Watson-Crick automaton.

Similar to deterministic Watson-Crick automaton, reversible Watson-Crick automaton can be stateless, all final, simple and 1-limited.

Inspired by the definition of reversible multihead finite automata in [3], we modify the accepting condition of reversible Watson-Crick automata in the following manner: We state that a reversible Watson-Crick automaton accepts a string $w$ if the automaton halts in a final state. If the automaton halts in a non-final state then the input string is rejected.

A reversible Watson-Crick automaton halts if the following two conditions occur:

1) When both the heads have gone past the right end of their respective input strings.

2) When the transition function is not defined on the current situation.

As the transitions employed in reversible Watson-Crick automaton are deterministic in nature at each application of the transition rule at least one character from the input string is consumed and as reversible Watson-Crick automaton is one way therefore it always halts after at most $2n$ steps. Where $n$ is the length of the input word. The language accepted by a reversible Watson-Crick automaton $M$ is $L(M)$={$w$ | $M$ accepts $w$}.

State complexity of a deterministic finite automaton is defined in [11].

In general, when we have two classes of generative (or recognizing) mechanisms, $G_1$, $G_2$ such that $G_1 \subseteq G_2$, and a complexity measure $CM$ defined on $G_2$ and extended in the natural way to languages, then

$$CM_{G_1}(L) \geq CM_{G_2}(L),$$

for all languages generated (recognized) by devices in $G_1$. Stronger forms of this relation can be considered.

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L) > CM_{G_2}(L),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

Clearly $\supseteq$ implies $\supset$ for each $i=2,3,4$.

The above definitions of state, transition complexity and succinctness are given in [11].

VI. COMPLEXITY MEASURES FOR AUTOMATA

The basic complexity measure of finite automata is the number of states, which was investigated in [14], [15], and [16]. Another parameter which estimates the size of a non-deterministic finite automaton is the number of transition rules (Trans). These complexity measures have also been employed on non-deterministic Watson-Crick automata [11].

Formally, if $Q$ and $\delta$ are the set of states and transition rules respectively of a given finite automaton $M$ or a Watson-Crick automaton $M$, then we denote $\text{State}(M)=\text{card}(Q)$, $\text{Trans}(M)=\text{card}(\delta)$, where $\text{card}(X)$ denotes cardinality of set $X$.

These measures have been extended to languages in the following way;

$$\text{State}_{NFA_1}(L) = \min\{\text{State}(M)|L=L(M), M \in NFA_1\},$$

where $NFA_1$ is a class of non-deterministic finite automata.

In general, $\text{State}_x(L) = \min\{\text{State}(M)|L=L(M), M \in x\}$, where $x$ can be any class of finite automata.

Succinctness

In general, when we have two classes of generative (or recognizing) mechanisms, $G_1$, $G_2$ such that $G_1 \subseteq G_2$, and a complexity measure $CM$ defined on $G_2$ and extended in the natural way to languages, then

$$CM_{G_1}(L) \geq CM_{G_2}(L),$$

for all languages generated (recognized) by devices in $G_1$. Stronger forms of this relation can be considered.

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L) > CM_{G_2}(L),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

$$G_1 \supseteq G_2 \Rightarrow CM_{G_1}(L_n) > CM_{G_2}(L_n),$$

Clearly $\supseteq$ implies $\supset$ for each $i=2,3,4$.

The above definitions of state, transition complexity and succinctness are given in [11].

VII. STATE COMPLEXITY OF REVERSIBLE WATSON-CRICK AUTOMATA
In this section, we are going to compare non-deterministic finite automata with deterministic reversible Watson-Crick automata with respect to state measure and look for relations \( \lambda_i \) as above with as large \( i \) as possible. We will show that deterministic reversible Watson-Crick automata similar to non-deterministic and deterministic Watson-Crick finite automata have an advantage over non-deterministic finite automata in terms of state complexity even though they are deterministic and reversible; which is contrary to the popular notion that the state complexity of non-deterministic automata is usually less than deterministic reversible automata.

**Theorem 1:** For every non-deterministic finite block automaton in which we can parse blocks of characters in each state, we can find an equivalent deterministic reversible Watson-Crick automaton which recognizes the same language \( L \) such that \( \text{State}_{\text{RK}}(L) \leq \text{State}_{\text{NFA}}(L) \) where \( \text{NFA}_A \) is a class of non-deterministic finite block automata.

Proof: Let us consider a non-deterministic finite block automaton \( M=(V, Q, q_0, F, \delta) \) without any \( \lambda \) transitions. This can be assumed, as for every non-deterministic finite block automaton with \( \lambda \) transitions we can obtain another non-deterministic finite block automaton without \( \lambda \) transitions having the same number of states and recognizing the same language. We can construct an equivalent deterministic reversible Watson-Crick automaton \( M'=(V', Q, q_0, F, \delta') \) with equal number of states, by the following method:

For every element \( 'x' \) in \( V \) and for each state \( q_i \) if there are \( p_1 \) transitions coming out of \( q_i \) where the block on which transition is taking place is \( 'x' \) then \( TS_{q_i}(x)=p_1 \).

For every element \( 'x' \) in \( V \) and for each state \( q_i \) if there are \( p_2 \) transitions coming out of \( q_i \) where the block on which transition is taking place begins with \( 'x' \) but does not end with \( 'x' \) then \( TB_{q_i}(x)=p_2 \).

For every element \( 'x' \) in \( V \) and each state \( q_i \) if there are \( p_3 \) transitions coming out of \( q_i \) where the block on which transition is taking place ends with \( 'x' \) but does not begin with \( 'x' \) then \( TE_{q_i}(x)=p_3 \).

For every element \( 'x' \) in \( V \) and each state \( q_i \) if there are \( p_4 \) transitions coming out of \( q_i \) where the block on which transition is taking place begins and ends with \( 'x' \) then \( TEB_{q_i}(x)=p_4 \).

Now for a particular element \( 'x' \) in \( V \) if the transition complexity for blocks different states are \( TS_{q_i}(x), TS_{q_2}(x), \ldots, TS_{q_n}(x) \) where \( n \) is the total number of states in \( M \), we define \( \text{CompS}(x)=\max(TS_{q_i}(x), TS_{q_2}(x), \ldots, TS_{q_n}(x)) \). Then for each \( 'x' \) in \( V \) new elements of the form \( x, x_2, \ldots, x_n \) where \( j=\text{CompS}(x) \) are introduced in \( V' \) along with \( 'x' \).

Now for a particular element \( 'x' \) in \( V \) if the transition complexity for blocks beginning with \( 'x' \) but not ending with \( 'x' \) in different states are \( TB_{q_1}(x), TB_{q_2}(x), \ldots, TB_{q_n}(x) \) where \( n \) is the total number of states in \( M \), we define \( \text{CompB}(x)=\max(TB_{q_1}(x), TB_{q_2}(x), \ldots, TB_{q_n}(x)) \). Then for each \( 'x' \) in \( V \) new elements of the form \( x, x_2, \ldots, x_n \) where \( k=\text{CompB}(x) \) are introduced in \( V' \) along with \( 'x' \).

Now for a particular element \( 'x' \) in \( V \) if the transition complexity for blocks ending with \( 'x' \) but not beginning with \( 'x' \) in different states are \( TE_{q_1}(x), TE_{q_2}(x), \ldots, TE_{q_n}(x) \) where \( n \) is the total number of states in \( M \), we define \( \text{CompE}(x)=\max(TE_{q_1}(x), TE_{q_2}(x), \ldots, TE_{q_n}(x)) \). Then for each \( 'x' \) in \( V \) new elements of the form \( x, x_2, \ldots, x_n \) where \( l=\text{CompE}(x) \) are introduced in \( V' \) along with \( 'x' \).

Now for a particular element \( 'x' \) in \( V \) if the transition complexity for blocks beginning and ending with \( 'x' \) in different states are \( TEB_{q_1}(x), TEB_{q_2}(x), \ldots, TEB_{q_n}(x) \) where \( n \) is the total number of states in \( M \), we define \( \text{CompEB}(x)=\max(TEB_{q_1}(x), TEB_{q_2}(x), \ldots, TEB_{q_n}(x)) \) and for each \( 'x' \) in \( V \) new elements of the form \( x_{1u}, x_{2u}, \ldots, x_{nu} \) and \( x_{1d}, x_{2d}, \ldots, x_{nd} \) where \( m=\text{CompEB}(x) \) are introduced in \( V' \) along with \( 'x' \).

The complementarity relations in deterministic Watson-Crick automaton for an element \( 'x' \) in \( V \) are of the form \( (x, x), (x, x_{1u}), (x, x_{2u}), \ldots, (x, x_{nu}), (x, x_{1d}), (x, x_{2d}), \ldots, (x, x_{nd}) \) where \( z=V \) without any \( \lambda \) transitions. For a particular transition \( q_i(x) \rightarrow q_j \) which has a order number say 1 introduce the transition rule \( q_i(x) \rightarrow q_j \) in \( \delta' \). If there is another transition \( q_i(x) \rightarrow q_k \) and has a order number 5 then the transition rule \( q_i(x) \rightarrow q_k \) is introduced in \( \delta' \).

Now suppose for an element \( 'x' \) in \( V \) there are \( n_1 \) transitions going out of state \( q_i \) whose blocks='x' where \( n_1 \leq \text{Comp}(x) \), then number the transitions of blocks='x' from 1 to \( n_1 \) in any particular order. Now for a particular transition \( q_i(x) \rightarrow q_j \) which has a order number say 1 introduce the transition rule \( q_i(x) \rightarrow q_j \) in \( \delta' \). If there is another transition \( q_i(x) \rightarrow q_k \) and has a order number 5 then the transition rule \( q_i(x) \rightarrow q_k \) is introduced in \( \delta' \).
not beginning with \( x \) from 1 to \( n \) in any particular order. Now for a particular transition \( q_i(zx)\to q_i' \) where \( z=aw \) where \( a\neq x \), which has a order number say 1 introduce the transition rule \( q_i(zx)\to q_i' \) in \( \delta' \). If there is another transition \( q_i(zx)\to q_k \) where \( z=ax \) and has a order number 4 then the transition rule \( q_i(zx)\to q_k \) is introduced in \( \delta' \).

Now suppose for an element \( x \) in \( V \) there are \( n \) transitions going out of state \( q_i \) whose blocks begins and ends with \( x \) but does not begin with \( x \) and block size is greater than 1 where \( n_i\leq \text{Comp}(x) \), then number the transitions of blocks beginning and ending with \( x \) from 1 to \( n \) in any particular order. Now for a particular transition \( q_i(zx)\to q_k \) where \( z=aV \), which has a order number say 1 introduce the transition rule \( q_i(zx)\to q_k \) in \( \delta' \). If there is another transition \( q_i(zx)\to q_k \) where \( z=V \) and has a order number 4 then the transition rule \( q_i(zx)\to q_k \) is introduced in \( \delta' \).

This method is repeated for each state of \( M \) with respect to all elements in \( V \). From the construction of \( M' \) from \( M \) we see that \( M' \) contains the same number of states and transitions as \( M \) and recognizes the same language \( L \).

Example 1
Consider the non-deterministic finite block automaton which recognizes the language \( (aab+bab)*aa \). Let it be \( M \). Here we will show how to obtain a deterministic reversible Watson-Crick automaton \( M' \) which recognizes the same regular language and has the same number of states as non-deterministic finite block automaton \( M \).

Let, \( M=(V, Q, q_0, F, \delta) \) where \( Q=\{q_0, q_1, q_2\} \), \( V=\{a,b\} \), \( q_0 \) is the start state, \( F=\{q_1\} \) and \( \delta: q_0(a)\to q_0, q_0(b)\to q_0, q_0(aa)\to q_0, q_1(b)\to q_0, q_2(a)\to q_1 \).

The equivalent deterministic reversible Watson-Crick automaton \( M' \) using the above mentioned procedure is,

\[
M'=(V', \rho, Q, q_0, F', \delta'):(a, a),(a, a_{1s}), (a, a_{1s}) \to (a, a_{1s}) (a, a_{1d}) \), (b, b), (b, b_{1d}), \}
\]

where \( a_{1s} \) and \( a_{1d} \) are introduced in \( \delta' \). If there is another transition \( q_i(zx)\to q_k \) where \( z=V \) and has a order number 4 then the transition rule \( q_i(zx)\to q_k \) is introduced in \( \delta' \).

Theorem 2: For every non-deterministic finite automaton in which only one character is passed in each state we can find an equivalent deterministic reversible Watson-Crick automaton which recognizes the same language \( L \) such that \( \text{State}_{\text{DRWK}}(L)\leq \text{State}_{\text{NFA}}(L) \).

Proof: Proof of Theorem 2 follows from Theorem 1 as a non-deterministic finite automaton is a special case of non-deterministic block automaton.

Example 2
Consider the non-deterministic finite automaton which recognizes the regular language \( (a+b)^*ab \). Let it be \( M \). Here we will show how to obtain a deterministic Watson-Crick automaton \( M' \) which recognizes the same regular language and has the same number of states as non-deterministic finite automaton \( M \).

Let, \( M=(V, Q, q_0, F, \delta) \) where \( Q=\{q_0, q_1, q_2\} \), \( V=\{a,b\} \), \( q_0 \) is the start state, \( F=\{q_2\} \) and \( \delta: q_0(a)\to q_0, q_0(b)\to q_0, q_0(aa)\to q_0, q_2(b)\to q_2 \).

The equivalent deterministic Watson-Crick automaton \( M' \) using the above mentioned procedure is,

\[
M'=(V', \rho, Q, q_0, F', \delta'):(a, a),(a, a_{1s}), (a, a_{1s}) \to (a, a_{1s}) (a, a_{1d}) \), (b, b), (b, b_{1d}), \}
\]

where \( a_{1s} \) and \( a_{1d} \) are introduced in \( \delta' \). If there is another transition \( q_i(zx)\to q_k \) where \( z=V \) and has a order number 4 then the transition rule \( q_i(zx)\to q_k \) is introduced in \( \delta' \).

For some integer \( k\geq 1 \), let us consider the language

\[ L_k = \{x^n \mid n \geq 2, x \in \{a,b\}^k\} \]

Lemma 1. \( \text{State}_{\text{DRWK}}(L_k) \leq 3 \).

Proof. The language \( L_k \) can be recognized by the deterministic reversible Watson-Crick automaton using 3 states in the following manner.

\[ M= (\{a,b,c\}, \{(a,a)(b,b),(a,c),(b,c)\}, \{q_0, q_1, q_2\}, q_0, \delta) \]

Thus to recognize strings whose lengths are multiple of \( k \) by the above deterministic Watson-Crick automaton the complementarity relation used should be the ‘c’ for the first and last \( k \) elements. For e.g. to recognize strings of length \( k=3 \), the input word should be of the form \( (aaabaaba)_{ccabacec} \) i.e. the first and last \( k \) letters have a complementarity relation which is not the identity relation.

Lemma 2: \( \text{State}_{\text{NFA}}(L_k) \geq 2^k + 1 \).

Proof. The proof of this lemma is in [11].

Theorem 3: \( \text{NFA}_0 > 4 \text{DRWK} \) and \( \text{NFA}_1 > 4 \text{DRWK} \).

Proof: We can arrive at this proof using Lemma 1, Lemma 2, Theorem 1, Theorem 2 and the definitions given in Section...
VI.

VIII. COMPUTATIONAL COMPLEXITY OF REVERSIBLE WATSON-CRICK AUTOMATA

Theorem 4: Strongly deterministic reversible Watson-Crick automata can accept all unary regular languages.

Proof: In [3] it is stated that a one-way multihead reversible finite automata with 2 heads (1RMFA(2)) accepts all unary regular language. From the definition of strongly deterministic reversible Watson-Crick automata it is evident that both have the same structure thus have the same computational power, as a result strongly deterministic reversible Watson-Crick automata also accepts all unary languages.

Theorem 5: Deterministic reversible Watson-Crick automata can accept all regular languages.

Proof: This result is a direct consequence of Theorem 2.

Theorem 6: Strongly deterministic reversible Watson-Crick automaton cannot accept the language $L=(a^*b^*)b^n$.

Proof: Strongly deterministic reversible Watson-Crick automaton has the same computational power as multihead reversible finite automaton with two heads which is evident from the description of strongly deterministic Watson-Crick automaton. In [3] it is stated that multihead reversible finite automaton with two heads cannot accept the language $L=(a^*b^*)b^n$. Thus strongly deterministic Watson-Crick automaton also cannot accept $L$.

Example 3: A strongly deterministic Watson-Crick automaton $M=(Q, V, \delta, q_0, F, \rho)$ can accept $L=(a^*b^*)b^n$ in the following manner.

$Q=\{q_0, q_a, q_b, q_s, q_f\}, V=\{a,b,\}$, $F=\{q_f\}$, $\rho$ is injective complementarity relation.

$\delta: q_0 (a) \rightarrow q_0, q_0 (b) \rightarrow q_0, q_0 (\$) \rightarrow q_s, q_0 (\lambda) \rightarrow q_2, q_0 (a) \rightarrow q_2, q_0 (b) \rightarrow q_{bb}, q_0 (\$) \rightarrow q_{\lambda}, q_0 (\lambda) \rightarrow q_f, q_0 (\$) \rightarrow q_f$.

The automaton $M$ works in the following manner each instance of the first 'b' which comes after 'a' in upper strand is matched to an instance of 'b' after $\$ in the lower strand.

Theorem 7: $SDRWK \subset SDWK$

Proof: Proof follows from Example 3 and Theorem 6

Example 4: A deterministic reversible Watson-Crick automaton $M=(Q, V, \delta, q_0, F, \rho)$ with non-injective complementarity relation can accept $L=(a^*b^*)b^n$ $n \geq 1$ in the following manner.

$Q=\{q_0, q_a, q_b, q_s, q_f\}, V=\{a,b,\}$, $F=\{q_f\}$, $\rho=\{(a,a), (a,a_s), (a,a_{bb}), (b,b), (b,b), ($,$)\}$

$\delta: q_0 (a) \rightarrow q_0, q_0 (b) \rightarrow q_0, q_0 (\$) \rightarrow q_s, q_0 (\lambda) \rightarrow q_2, q_0 (a) \rightarrow q_2, q_0 (b) \rightarrow q_{bb}, q_0 (\$) \rightarrow q_{\lambda}, q_0 (\lambda) \rightarrow q_f, q_0 (\$) \rightarrow q_{nf}$.

This automaton will accept strings from the language $(a^*b^*)b^n$, only when the first ‘a’ symbol at the beginning of the string will have as its complement ‘a$$. ‘a’ following a ‘b’ will have $a_b$ as its complement and b following a ‘a’ will have b$ as its complement. All other ‘a’ s and ‘b’ s have injective complementarity relation. If the string does not belong to $(a^*b^*)b^n$ then the automaton will not accept the string no matter what complementary string is used. The automaton $M$ works in the following manner each instance of the first 'b' which comes after 'a' in lower strand is matched to an instance of 'b' after $\$ in the upper strand. Thus the automaton $M$ accepts the language $L=(a^*b^*)b^n$. From the transitions in $\delta$ we see that $M$ is a deterministic reversible Watson-Crick automaton.

Theorem 8: Strongly deterministic reversible Watson-Crick automata are a proper subset of deterministic reversible Watson-Crick automata.

Proof: Every language accepted by strongly deterministic reversible Watson-Crick automaton, we have a deterministic reversible Watson-Crick automaton which accepts the same language which is evident from both their definitions as every strongly deterministic reversible Watson-Crick automaton is a deterministic reversible Watson-Crick automaton. But there is a language
L=(a^*b)*Sb* not accepted by any strongly deterministic Watson-Crick automaton. From Example 4, we see that L is accepted by a deterministic reversible Watson-Crick automaton. Thus strongly deterministic reversible Watson-Crick automata are a proper subset of deterministic reversible Watson-Crick automata.

**Theorem 9:** One way multichip reversible automata with two heads are a proper subset of deterministic reversible Watson-Crick automata.

Proof: The proof follows from Theorem 6, 8 and Example 4.

**Example 5:** L = \{#w_1x_1#,...,#w_nx_n|n\geq0, w_i \in \{a,b\}^*, x_i \in \{a,b\}^*, \exists \exists_j : w_i=x_j, x_i\neq x_j\} is accepted by a deterministic reversible Watson-Crick automaton with non-injective complementarity relation.

Let, \(M=(V,\rho,Q,q_0,F,\delta)\) be a Watson-Crick automaton, where \(V=\{a,b,v_1, v_2, #\}, \rho=\{(a,a),(\#,\#),(\#,v_1),(\#,v_2),(\#,\#)\}, Q=\{q_0, q_1, q_2, q_3\}, F=\{q_3\}\), and we have the following transitions:

\[\begin{align*}
q_0(a) &\rightarrow q_0, q_0(b) \rightarrow q_0, q_0(#) \rightarrow q_0, q_0(v_1) \rightarrow q_1, q_0(v_2) \rightarrow q_2, q_0(#) \rightarrow q_3, q_1(a) \rightarrow q_1, q_1(b) \rightarrow q_1, q_1(#) \rightarrow q_1, q_1(v_1) \rightarrow q_1, q_1(v_2) \rightarrow q_3, q_2(a) \rightarrow q_2, q_2(b) \rightarrow q_2, q_2(#) \rightarrow q_2, q_2(v_1) \rightarrow q_3, q_2(v_2) \rightarrow q_2, q_3(#) \rightarrow q_3, q_3(v_1) \rightarrow q_3, q_3(v_2) \rightarrow q_3.
\end{align*}\]

L is not accepted by any k-head deterministic finite automaton [17]. As a result, L is not accepted by any strongly deterministic Watson-Crick automaton. In Example 5, L is accepted by deterministic Watson-Crick automaton by using its non-injective complementarity relation property. ‘v_1’ and ‘v_2’ are used as complements of ‘#’ to guess the two words in the input string which have their w parts equal but x parts not equal.

Then the two guessed words are compared and if they don’t match at any position in their “x” parts but match in their “w” parts then the input string is accepted. If there is no two words in the input string such that there “w” parts are equal and “x” parts are not then no matter whether ‘v_1’ and ‘v_2’ are placed in place of ‘#’ it will never be accepted.

**Theorem 10:** \(L_{DRWK} \cup L_{SDWK} \neq \emptyset\) where \(L_{DRWK}\) is the set of languages accepted by deterministic reversible Watson-Crick automata and \(L_{SDWK}\) is the set of languages accepted by strongly deterministic Watson-Crick automata.

Proof: From Example 5 we see that there exists a deterministic reversible Watson-Crick automaton which accepts the language L = \{#w_1x_1#,...,#w_nx_n|n\geq0, w_i \in \{a,b\}^*, x_i \in \{a,b\}^*, \exists \exists_j : w_i=x_j, x_i\neq x_j\}, moreover as stated in [17], L is not accepted by any k-head deterministic automata thus L cannot be accepted by any strongly deterministic Watson-Crick automaton. Thus the proof follows from the above observation.

**Example 6:** Twin-shuffle language with end marker $ is accepted by a strongly deterministic Watson-Crick automaton.

Let \(M=(V,\rho,Q,q_0,F,\delta)\) be a Watson-Crick automaton, where \(V=\{a, b, \overline{a}, \overline{b}, S\}, \rho\) is injective complementarity relation, \(Q=\{q_0, q_1, q_2\}, F=\{q_0\}\), and we have the following transitions:

\[\begin{align*}
q_0(a) &\rightarrow q_0, q_0(b) \rightarrow q_0, q_0(S) \rightarrow q_0, q_0(\overline{a}) \rightarrow q_0, q_0(\overline{b}) \rightarrow q_0, q_0(S) \rightarrow q_0, q_1(a) \rightarrow q_1, q_1(b) \rightarrow q_1, q_1(\overline{a}) \rightarrow q_1, q_1(\overline{b}) \rightarrow q_1, q_1(S) \rightarrow q_1, q_2(a) \rightarrow q_1, q_2(b) \rightarrow q_1, q_2(\overline{a}) \rightarrow q_2, q_2(\overline{b}) \rightarrow q_2, q_2(S) \rightarrow q_2.
\end{align*}\]

The idea behind this automaton M is that we exploit the property of twin-shuffle language that is the order of the unbarred characters are same as order of the barred characters. So in the upper strand we look for the unbarred character and then look for the corresponding barred version in the lower strand.

**Theorem 11:** Strongly deterministic Watson-Crick automaton can accept twin-shuffle language.

Proof: Proof of the theorem follows from Example 6.

**Theorem 12:** There is no deterministic reversible Watson-Crick automaton or strongly deterministic reversible Watson-Crick automaton that accepts the twin-shuffle language.

Proof: To accept the twin-shuffle language using any two head one way finite automaton we need to identify a unbarred character in the upper strand and then look for its barred version in the lower strand or we can do the opposite i.e. for a barred character in the upper strand we look for a unbarred character in the lower strand. Both barred and unbarred versions cannot be compared on the same strand. Without loss of generality, we assume the first case. In assuming the first case, whenever we find an unbarred character the control shifts to the lower head and when the corresponding barred character is found then the control again shifts to the upper head to look for the next unbarred character in the input string. Thus to do this switch of control from upper head to lower head on unbarred character and then back to upper head on finding its barred version to look for the next unbarred character, there must be a state in the automaton which is responsible for this switch of head from upper to lower and to which the control returns when the corresponding barred variant is found. The control must return to this above mentioned state because the number of unbarred symbols in the input string is unknown so we cannot use a different state for
each unbarred character encountered in the input string and thus there must be a state which has a self loop for the barred characters and which switches the control to the lower head on reading an unbarred character and to which the control returns on finding the corresponding barred version. Thus to accept the twin-shuffle language using a deterministic Watson-Crick automaton it must have a state $q$ with transitions of the form $q(\lambda) \rightarrow q$, $q(\lambda) \rightarrow q$ from $q$ and transitions of the form $s(\lambda) \rightarrow q$, $r(\lambda) \rightarrow q$ coming into $q$ where $r$ and $s$ are states in $M$. Thus when we check for reversibility of the automaton which accepts the twin shuffle-language we find that due to the presence of state $q$ the automaton cannot be deterministic reversible Watson-Crick automaton because when the transitions coming into $q$ are reversed we find $q$ have transitions of the form $\lambda(\lambda)$, $\lambda(\lambda)$, $\lambda(\lambda)$, coming out of it. Thus the given automaton on reversal of transitions cannot satisfy the constraints in transition rules of deterministic Watson-Crick automata.

**Theorem 13:** $L_{SDWK} \neq L_{DRWK}$ where $L_{SDWK}$ is the set of languages accepted by strongly deterministic Watson-Crick automata and $L_{DRWK}$ is the set of languages accepted by strongly deterministic Watson-Crick automata.

Proof: Proof follows from Theorem 12.

**Theorem 14:** DRWK and SDWK are incomparable.

Proof: Proof of the theorem follows from Theorem 13 and Theorem 10.

**Theorem 15:** Strongly deterministic reversible Watson-Crick automata and CFL are incomparable.

Proof: The proof of the theorem follows from the fact that it has been shown in [3] that 1RMFA(2) can accept some context sensitive languages and as strongly deterministic Watson-Crick automata and 1RMFA(2) are same in terms of their definition strongly deterministic Watson-Crick automata also accepts some context sensitive languages. Moreover non-deterministic Watson-Crick automata cannot accept the mirror language which is context free [18] so neither can strongly deterministic reversible Watson-Crick automata as a result strongly deterministic reversible Watson-Crick automata and CFL are incomparable.

**Theorem 16:** DRWK $\subseteq$ AWK

Proof: DRWK is a subset of AWK is evident from the definition of AWK and DRWK and the subset relation is proper due to the fact that AWK can accept the twin-shuffle language but DRWK cannot.

**IX. CONCLUSION**

In this paper we have discussed about the state complexity and computational complexity of reversible Watson-Crick automata and compared its state complexity in terms of accepting regular languages with non-deterministic finite automata (NFA$_1$) and non-deterministic finite block automata (NFA$_b$). We have also shown that in spite of deterministic reversible Watson-Crick automata being deterministic and reversible in nature they hold similar advantages over non-deterministic finite automata as non-deterministic Watson-Crick automata. We have further shown that reversible Watson-Crick automata in spite of being reversible can accept all regular languages. We have also explored and compared the computational power of different variants of reversible Watson-Crick automata. We have further compared the computational power of reversible multihead automata with two heads with reversible Watson-Crick automata and established the superiority of deterministic reversible Watson-Crick automata in terms of computability.

**REFERENCES**

[1] C.H Bennet, Logical reversibility of computation. IBM J. Res. Dev. 17, 525–532, 1973.

[2] K.Morita, Two-way reversible multi-head finite automata. Fund. Inform. 110, 241–254, 2011.

[3] M. Kutrib, A. Malcher, One-Way Reversible Multi-head Finite Automata, Reversible Computation, Lecture Notes in Computer Science Volume 7581, pp 14-28, 2013.

[4] M. Kutrib, A. Malcher, reversible pushdown automata, Journal of Computer and System Sciences, Volume 78, Issue 6, Pages 1814 - 1827, November 2012.
[5] J.E Pin. On reversible automata, Simon. Proceedings of the first LATIN conference, Sao-Paulo, Brazil. Springer, pp.401-416, Lecture Notes in Computer Science 583, 1992.

[6] R.Freund, G.Păun, G.Rozenberg, A.Salomaa, Watson-Crick Finite Automata, Proc 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 297-328, 1997.

[7] G. Păun, G. Rozenberg, A. Salomaa, DNA Computing: New Computing Paradigms, Springer-Verlag, Berlin, 1998.

[8] E.Czeizler, E.Czeizler, L.Kari, K.Salomaa, On the descriptional complexity of Watson-Crick automata, Theoretical Computer Science, Volume 410, Issue 35, Pages 3250–3260, 28 August 2009.

[9] K. S Ray, K. Chatterjee, D. Ganguly, Equivalence of Subclasses of Two-Way Non-Deterministic Watson-Crick Automata, Applied Mathematics,Vol.4, No.10A, October 2013.

[10] E. Czeizler, E. Czeizler, A Short Survey on Watson-Crick Automata, Bull. EATCS 88 104-119, 2006.

[11] A.Păun, M.Păun,State and transition complexity of Watson-Crick finite automata, Fundamentals of Computation Theory,Lecture Notes in Computer Science, Volume 1684, pp 409-420,1999.

[12] K. S Ray, K. Chatterjee, D. Ganguly, State complexity of deterministic Watson–Crick automata and time varying Watson–Crick automata, Natural Computing, February 2015.

[13] J.M. Sempere, Exploring regular reversibility in Watson-Crick finite automata, The Thirteenth International Symposium on Artificial Life and Robotics 2008(AROB 13th ’08), B-Con Plaza, Beppu, Oita, Japan, January 31-February 2, 2008.

[14] S.Yu, Regular languages, in Handbook of Formal Languages,vol.1,chap.2(Springer),pp.41-110,1997.

[15] C.Campeanu, N.Santean, S.Yu, Minimal cover automata for finite languages, International workshop on Implementing automata, WIA 98, Rouen, 32-42, 1998.

[16] K.Salomaa, S.Yu,Q.Zhuang,The state complexities of some basic operations on regular languages, Theoretical Computer Science, 125, 315-328, 1994.

[17] A.C. Yao, R.L. Rivest, k + 1 Heads Are Better than k, Journal of the ACM (JACM), Volume 25 Issue 2, Pages 337-340, April 1978

[18] 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, Ireland, 6–7th volume 1 of EPTCS, pages 93–107, December 2008.