DCFS 2008 (Draft)

Descriptional Complexity of Formal Systems
Deadline for submissions: April 7, 2008
Final versions: June 25, 2008

# Watson-Crick automata: determinism and state complexity

Elena Czeizler[(A)] Eugen Czeizler[(A)] Lila Kari[(A)] Kai Salomaa[(B)]

[(A)]Department of Computer Science, University of Western Ontario,
London, Ontario N6A 5B7, Canada
elenac@csd.uwo.ca eugenc@csd.uwo.ca lila@csd.uwo.ca

[(B)]School of Computing, Queen's University,
Kingston, Ontario K7L 3N6, Canada,
ksalomaa@cs.queensu.ca

**Abstract.** Watson-Crick automata are finite state automata working on double-stranded tapes, introduced to investigate the potential of DNA molecules for computing. In this paper, we continue the investigation of descriptional complexity of Watson-Crick automata initiated in [9]. In particular, we show that any finite language as well as any unary regular language can be recognized by a Watson-Crick automaton with only two and respectively three states. Also, we formally define the notion of determinism for these systems. Contrary to the case of non-deterministic Watson-Crick automata, we show that for deterministic ones, the complementarity relation plays a major role in the acceptance power of these systems.

**Keywords:** Watson-Crick automata, state complexity, determinism

## 1 Introduction

One of the current trends in nanoengineering is to develop nanomachines which can parse molecules of DNA and perform a finite number of tasks, e.g., the development of artificial enzymes [12], or smart drug design [13]. One of the first theoretical abstractions for such nanomachines are *Watson-Crick automata* [2], which are based on the idea of finite automata running on complete DNA-molecules. Formally, these machines are finite automata, with two independent reading heads, working on double stranded sequences. The two strands of the input are separately scanned from left to right by read-only heads controlled by a common state. One of the main features of these automata is that characters on corresponding positions from the two strands of the input are related by a complementarity relation similar with the Watson-Crick complementarity of the DNA nucleotides. Several variants of these systems were investigated, e.g., in [8] and [11]; for a comprehensive presentation we refer to both Chapter 5 from [10] as well as to [1] for a recent survey.

In this paper, we continue the study of the descriptional complexity of Watson-Crick automata initiated in [9]. Since, at each step, Watson-Crick automata can read blocks of more than one letter, a special feature of these systems is that for a given number of states, even two or three, one can already define an infinite number of distinct

automata. This is different from most of the usually considered models, such as ordinary finite automata or Turing machines. Thus, Watson-Crick automata allow, in some sense, to encode state information in the finite but unbounded number of transitions and this makes it essentially more difficult to prove lower bounds for the number of states. In particular, we prove that several intricate families of languages can be accepted by Watson-Crick automata with a small, constant, number of states. For instance, we show that any finite language and any unary regular language can be recognized by a Watson-Crick automaton with two and respectively three states. Also, we provide a family of languages which generates an infinite hierarchy from the point of view of the state complexity of the block automata recognizing them. Then, we show that this hierarchy collapses when we consider the case of Watson-Crick automata, that is, we prove that three states are enough when recognizing any language from this family. Recall that *block automata*, or *block-NFA*, are finite automata which, similarly to the case of Watson-Crick automata, can read an arbitrarily long finite sequence of characters at a time.

Also, we formally define the notion of deterministic Watson-Crick automata and investigate their properties. Although determinism is a well established notion in automata theory, it has never been considered yet in relation with Watson-Crick automata. In this paper, we define the notion of determinism using a syntactic property of the rewriting rules of the automaton. We also consider a weaker operational definition of determinism, namely weak determinism, and show that it is undecidable whether a given non-deterministic Watson-Crick automaton is weakly deterministic. For non-deterministic Watson-Crick automata it was proved in [7] that we can always suppose the complementarity relation to be the identity. Hence, a natural question is whether the structure of the complementarity relation plays an active role in the deterministic case. Thus, we define the notion of strong determinism, embedding both the deterministic feature and the fact that the complementarity relation is the identity. We prove that these three levels of abstraction, i.e., weak determinism, determinism, and strong determinism, are all distinct from each other. Moreover, we also show that non-deterministic Watson-Crick automata are strictly stronger than strongly deterministic ones.

The paper is organized as follows. In the next section, we fix our terminology and recall some known results. In Section 3, we investigate some properties of deterministic Watson-Crick automata, on the three levels of abstraction mentioned above. Also, we look at the relation between the acceptance power of these three variants of deterministic Watson-Crick automata. In Section 4, we investigate the state complexity of both deterministic and non-deterministic Watson-Crick automata.

## 2 Preliminaries

Let $V$ be a finite alphabet. We denote by $V^*$ the set of all finite words over $V$, by $\lambda$ the empty word, and $V^+ = V^* \backslash \{\lambda\}$. For a word $u \in V^*$, we denote by $|u|$ its *length*, i.e., the number of letters occurring in it; in particular, $|\lambda| = 0$. We say that $u \in V^*$ is a *prefix* of a word $v$, and denote it by $u \leq v$, if there exists some $t \in V^*$ such that

$v = ut$. Two words $u$ and $v$ are *prefix comparable*, denoted by $u \sim_p v$, if one of them is a prefix of the other. For a word $u = u_1 \dots u_n$, with $u_1, \dots, u_n \in V$, we denote by $u^R = u_n \dots u_1$ its reverse. Then, we say that $u$ is *palindrome* if $u = u^R$.

Let now $\rho \subseteq V \times V$ be a symmetric relation, called *the Watson-Crick complementarity relation on* $V$. As suggested by the name, this relation is biologically inspired by the Watson-Crick complementarity of nucleotides in the double stranded DNA molecule. In accordance with the representation of DNA molecules, viewed as two strings written one on top of the other, we write $\binom{V^*}{V^*}$ instead of $V^* \times V^*$ and an element $(w_1, w_2) \in V^* \times V^*$ as $\binom{w_1}{w_2}$.

We denote $\begin{bmatrix} V \\ V \end{bmatrix}_\rho = \{\begin{bmatrix} a \\ b \end{bmatrix} \mid a, b \in V,\ (a, b) \in \rho\}$ and $WK_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*$. The set $WK_\rho(V)$ is called *the Watson-Crick domain* associated to $V$ and $\rho$. An element $\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \dots \begin{bmatrix} a_n \\ b_n \end{bmatrix} \in WK_\rho(V)$ can be also written in a more compact form as $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, where $w_1 = a_1 a_2 \dots a_n$ and $w_2 = b_1 b_2 \dots b_n$.

The essential difference between $\binom{w_1}{w_2}$ and $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ is that $\binom{w_1}{w_2}$ is just an alternative notation for the pair $(w_1, w_2)$, whereas $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ implies that the strings $w_1$ and $w_2$ have the same length and the corresponding letters are connected by the complementarity relation.

A *(non-deterministic) Watson-Crick finite automaton* is a 6-tuple $\mathcal{M} = (V, \rho, Q, q_0, F, P)$, where: $V$ is the (input) alphabet, $\rho \subseteq V \times V$ is the complementarity relation, $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $P$ is a finite set of transition rules of the form $q \binom{w_1}{w_2} \to q'$, denoting the fact that if the automaton is in a state $q$ and parses $w_1 \in V^*$ on the upper strand and $w_2 \in V^*$ on the lower strand, then it enters the state $q'$.

A *configuration* of a Watson-Crick automaton is a pair $(s, \binom{u}{v})$ where $s$ is the current state of the automaton and $\binom{u}{v}$ is the part of the input word which has not been read yet. Now, a *transition* between two configurations is defined as follows. For $\binom{u_1 v_1}{u_2 v_2} \in \binom{V^*}{V^*}$ and $q, q' \in Q$ we write $q \binom{u_1 v_1}{u_2 v_2} \Rightarrow q' \binom{v_1}{v_2}$ if and only if $q \binom{u_1}{u_2} \to q'$. If we denote by $\Rightarrow^*$ the reflexive and transitive closure of the relation $\Rightarrow$, then the *language accepted by a Watson-Crick* automaton is:

$$L(\mathcal{M}) = \{w_1 \in V^* \mid\ q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* s, \text{ with } s \in F,\ w_2 \in V^*, \text{ and } \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}.$$

Hence, a word $w_1$ is accepted by $\mathcal{M}$ if there exists a complementary word $w_2$ such that starting from the initial state, after parsing the whole input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ the automaton

is in a final state. By convention, as suggested also in [9], we consider two languages differing only by the empty word as identical.

Although the notion of determinism is well established in automata theory, it has never been considered in relation with Watson-Crick automata. Intuitively, this notion suggests that during a computation, in each state we have only one option to continue. Here, we propose three variants for this concept, each on a different level of abstraction. The first definition that we suggest illustrates the intuitive idea we presented above.

**Definition 1.** We say that a Watson-Crick automaton is *weakly deterministic* if in every configuration that can occur in some computation of the automaton there is at most one possibility to continue the computation.

Note that the previous definition does not provide a clear description of the structure of the transition rules of a deterministic automaton. Thus, we introduce our second definition.

**Definition 2.** We call a Watson-Crick automaton *deterministic* if whenever we have two rewriting rules of the form $q \begin{pmatrix} u \\ v \end{pmatrix} \to q'$ and $q \begin{pmatrix} u' \\ v' \end{pmatrix} \to q''$, then $u \nsim_p u'$ or $v \nsim_p v'$.

Clearly, the deterministic constraint is stronger than the weakly one. However, unexpectedly, Example 9 shows that a weakly deterministic automaton need not be deterministic. With the previous two definitions, for a given pair of words from the domain, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)$, the computation is unique. However, depending on the complementary relation $\rho$, the automaton can chose various words $w_2$ on the lower strand. Thus, in order to eliminate this selection, we introduce our third definition.

**Definition 3.** We call a Watson-Crick automaton *strongly deterministic* if it is deterministic and the Watson-Crick complementarity relation is the identity.

Note that the notion of strong determinism does not change if $\rho$ is allowed to be a non-identity one-to-one function. As shown later by Theorem 8, strongly deterministic Watson-Crick automata are less powerful than deterministic ones.

Depending on the type of the states and of the rewriting rules, there are four subclasses of Watson-Crick automata. We say that a Watson-Crick automaton $\mathcal{M}$ is

- *stateless* if it has only one state, i.e., $Q = F = \{q_0\}$;
- *all-final* if all the states are final, i.e., $Q = F$;
- *simple* if for any rewriting rule $s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to s'$, either $w_1 = \lambda$ or $w_2 = \lambda$;
- *1-limited* if for any rewriting rule $s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to s'$ we have $|w_1 w_2| = 1$.

Recently, in [7], it was proved that for non-deterministic Watson-Crick automata we can always suppose the complementarity relation $\rho$ to be the identity, denoted, from now on, by $\iota \subseteq V \times V$. However, this is not true anymore for the deterministic case, as shown later by Theorem 8.

## 3 Properties of deterministic Watson-Crick automata

One of the basic properties of non-deterministic Watson-Crick automata is that they are equivalent with simple and 1-limited ones, respectively, see [10]. The following two results show that this property still holds when we look at their deterministic variants.

**Theorem 4.** *Deterministic Watson-Crick automata are equivalent with deterministic simple Watson-Crick automata.*

*Proof*: (Sketch) Let $\mathcal{M} = (V, \rho, Q, q_0, F, P)$ be a deterministic Watson-Crick automaton. We construct an equivalent deterministic Watson-Crick automaton $\mathcal{M}' = (V, \rho, Q', q_0, F, P')$, where for any state $q \in Q'$ all rewriting rules from $q$, $q \begin{pmatrix} u_i \\ v_i \end{pmatrix} \to q_i$ with $1 \leq i \leq n$, satisfy exactly one of the following conditions:

$$\text{either } u_i = \lambda \text{ for all } 1 \leq i \leq n \text{ and } v_j \nsim_p v_k \text{ for any } 1 \leq j \neq k \leq n, \tag{1}$$

$$\text{or } v_i = \lambda \text{ for all } 1 \leq i \leq n \text{ and } u_j \nsim_p u_k \text{ for any } 1 \leq j \neq k \leq n. \tag{2}$$

Thus, we introduce some new intermediate states and transform the rewriting rules to achieve the above constraint. Since this construction is long and quite technical, we present here only one illustrative case, the others being similar.

If for some $q \in Q$ we have a rule $q \begin{pmatrix} u \\ v \end{pmatrix} \to q'$, where either $u_i = \lambda$ or $v_i = \lambda$, then the rule remains unchanged in $\mathcal{M}'$. Now, suppose that there exists a state $q \in Q$ such that we have some rules of the form $q \begin{pmatrix} u_i \\ v_i \end{pmatrix} \to q_i$ with $1 \leq i \leq n$ and all $u_i, v_i \neq \lambda$, and, moreover, $u_1 \leq u_i$ for all $i \geq 2$. We introduce a new state $s$ and we transform all the rules as follows. First, we introduce the rule $q \begin{pmatrix} u_1 \\ \lambda \end{pmatrix} \to s$. Then, each rule $q \begin{pmatrix} u_i \\ v_i \end{pmatrix} \to q_i$ is replaced by $s \begin{pmatrix} u_i' \\ v_i \end{pmatrix} \to q_i$, where $u_i = u_1 u_i'$ with $u_i' \in V^*$ and $|u_i'| < |u_i|$. Moreover, for this newly introduced state $s$, the words on the upper strands of the rules initiating in $s$ are strictly shorter than before. By König's lemma it follows that the process of adding new rules terminates with a finite number of rules.

Since each newly introduced state acts just as an intermediate, the language accepted by $\mathcal{M}'$ is exactly $L(\mathcal{M})$. □

Using a similar technique, we can transform a deterministic simple Watson-Crick automaton into a deterministic 1-limited one. Thus we can state the following result.

**Corollary 5.** *Deterministic Watson-Crick automata are equivalent with deterministic 1-limited Watson-Crick automata.*

As stated in [10], a 1-limited Watson-Crick automaton can be interpreted as a one way two headed automaton where the two strands are interrelated through the complementarity relation. Furthermore, as arbitrary Watson-Crick automata are equivalent with 1-limited ones, see [10], we can also state that they are equivalent with one way two headed automata. Moreover, by Corollary 5, deterministic Watson-Crick automata

using the identity complementarity relation are equivalent with deterministic one way two headed automata. Thus, we can prove the following result.

**Theorem 6.** *Non-deterministic Watson-Crick automata are more powerful than strongly deterministic ones.*

*Proof*: Let $L = \{w \in V^* \mid w = w^R\}$ be the set of palindrome words and $L' = V^* \setminus L$ be its complement. It is known, see [5] and [14], that $L'$ can be recognized by a non-deterministic one way two headed automaton but not by a deterministic one. Thus, $L'$ can be recognized by a nondeterministic Watson-Crick automaton but not by a strongly deterministic one. □

The next example shows that if we use a non-injective complementarity relation $\rho$, then we can construct a deterministic Watson-Crick automaton accepting the language $L'$ from the previous result.

**Example 7.** Let $\mathcal{M} = (V, \rho, Q, q_0, F, P)$ be a Watson-Crick automaton, where $V = \{a, b, v_a, v_b, c\}$, $\rho = \{(a,a), (a,v_a), (v_a,a), (b,b), (b,v_b), (v_b,b), (c,a), (a,c)(c,b), (b,c)\}$, $Q = \{q_0, q_1, q_a, q_b\}$, $F = \{q_1\}$, and we have the following transition rules:

- $q_0 \begin{pmatrix} \lambda \\ x \end{pmatrix} \to q_0$, $q_0 \begin{pmatrix} \lambda \\ v_x \end{pmatrix} \to q_x$, with $x \in \{a, b\}$,
- $q_x \begin{pmatrix} y \\ z \end{pmatrix} \to q_x$, with $x, y, z \in \{a, b\}$,
- $q_x \begin{pmatrix} zy \\ c \end{pmatrix} \to q_1$, with $x, y, z \in \{a, b\}, x \neq y$,
- $q_1 \begin{pmatrix} x \\ \lambda \end{pmatrix} \to q_1$, with $x \in \{a, b\}$.

It is easy to see that $\mathcal{M}$ is deterministic. Let $w \in \{a, b\}^*$, $w = w_1 w_2 \dots w_n$ with $w_i \in \{a, b\}$. If $w \neq w^R$, then there exists a position $k$ on the first half of $w$ such that $w_k \neq w_{n-k}$. The automaton $\mathcal{M}$ accepts the word $w$ only when we chose as its complement the word $w_1 \dots w_{k-1}\ v_{w_k}\ w_{k+1} \dots w_{n-1} c$. On the other hand, if $w$ is a palindrome word, then it will not be accepted, regardless of what complement we use; thus, $L(\mathcal{M}) = \{w \in \{a, b\}^+ \mid w \neq w^R\}$.

Thus, we can formulate the following result showing that the complementarity relation plays an active role for deterministic Watson-Crick automata, contrary to the non-deterministic case, see [7].

**Theorem 8.** *Strongly deterministic Watson-Crick automata are strictly weaker than deterministic ones.*

Now, a natural and interesting question, which remains still open, is whether non-deterministic Watson-Crick automata are equivalent with deterministic ones, clearly, using a non-injective complementarity relation.

As we already stated in the previous section, the deterministic constraint is stronger than the weakly deterministic one. Moreover, the following example presents a weakly deterministic Watson-Crick automaton which is not deterministic.

**Example 9.** Let us consider the non-regular language $L = \{a^n b^n \mid n \geq 1\} \cup \{b^n a^n \mid n \geq 1\}$. Then, let $\mathcal{M} = (V, \iota, Q, q_0, F, P)$, where $V = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $F = \{q_3, q_4\}$, and $P$ contains the following productions:

- $q_0 \begin{pmatrix} a \\ \lambda \end{pmatrix} \to q_1$ and $q_0 \begin{pmatrix} \lambda \\ b \end{pmatrix} \to q_2$,
- $q_1 \begin{pmatrix} a \\ \lambda \end{pmatrix} \to q_1$, $q_1 \begin{pmatrix} b \\ a \end{pmatrix} \to q_3$, $q_3 \begin{pmatrix} b \\ a \end{pmatrix} \to q_3$, $q_3 \begin{pmatrix} \lambda \\ b \end{pmatrix} \to q_3$,
- $q_2 \begin{pmatrix} \lambda \\ b \end{pmatrix} \to q_2$, $q_2 \begin{pmatrix} b \\ a \end{pmatrix} \to q_4$, $q_4 \begin{pmatrix} b \\ a \end{pmatrix} \to q_4$, $q_4 \begin{pmatrix} a \\ \lambda \end{pmatrix} \to q_4$.

Clearly, the language recognized by this automaton is $L$. The only non-deterministic choice can be made in $q_0$ at the beginning of a computation. However, given an input, this choice becomes uniquely determined. Thus, $\mathcal{M}$ is a weakly deterministic Watson-Crick automaton which is not deterministic, due to the first two rules in $q_0$.

It is not yet known whether weakly deterministic Watson-Crick automata recognize more languages than deterministic ones.

Although strongly deterministic Watson-Crick automata are weaker than non-deterministic ones, they still prove to be more powerful than finite automata, even if we look at their stateless versions. In [10] it was proved that if $L$ is the language accepted by a stateless Watson-Crick automaton $\mathcal{M}$, then $L = L^+$. For the case of strongly deterministic automata, we refine this result using prefix codes.

We say that a language $L$ is a *prefix code* if no two (distinct) words of the language are prefix comparable. That is, for any two words $w_1, w_2 \in L$ such that $w_1 \neq w_2$, we have $w_1 \not\sim_p w_2$. For more detailed information on prefix codes we refer to [6].

**Theorem 10.** *For any strongly deterministic stateless Watson-Crick automaton $\mathcal{M}$ there exists a prefix code $L$ such that $L(\mathcal{M}) = L^*$.*

*Proof*: Let $L \subseteq L(\mathcal{M})$ be the language obtained from $L(\mathcal{M})$ by taking all those non-empty words whose proper prefixes are not accepted by the automaton $\mathcal{M}$, i.e.,

$$L = \{w \in L(\mathcal{M}) \mid w \neq \lambda \text{ and for all } v \in L(\mathcal{M}) \setminus \{\lambda\} \text{ if } v \leq w \text{ then } v = w\}.$$

Clearly, $L$ is a prefix code and $L^* \subseteq L(\mathcal{M})$. We show that if $v \in L(\mathcal{M})$, then $v \in L^*$.

Let $q_0$ be the state of the automaton $\mathcal{M}$. Then, all the rewriting rules from $\mathcal{M}$ are of the form $q_0 \begin{pmatrix} u_i \\ v_i \end{pmatrix} \to q_0$ for $1 \leq i \leq n$, where moreover, for all $1 \leq i \neq j \leq n$ either $u_i \not\sim_p u_j$ or $v_i \not\sim_p v_j$ (or both). Let us consider now a non-empty word $v$ recognized by the automaton $\mathcal{M}$, i.e., $v \in L(\mathcal{M})$, such that $v \notin L$. Then, there must exist a word $w \in L$, such that $w \leq v$; let $v = ww'$ for some $w'$. Let $q_0 \begin{pmatrix} u_{i_1} \\ v_{i_1} \end{pmatrix} \to q_0$ and $q_0 \begin{pmatrix} u_{j_1} \\ v_{j_1} \end{pmatrix} \to q_0$ be the first rewriting rules applied when recognizing $w$ and $v$, respectively. Since $w \leq v$ we conclude that $u_{i_1} \sim_p u_{j_1}$ and also $v_{i_1} \sim_p v_{j_1}$. Thus, since $\mathcal{M}$ is deterministic, it implies that the two rules must coincide, i.e., $u_{i_1} = u_{j_1}$

and $v_{i_1} = v_{j_1}$. Similarly, we can conclude that all the rewriting rules applied when recognizing $w$ and $v = ww'$ are exactly the same, till we start parsing $w'$. So, at some moment in the recognition process of $v$, after parsing $w$, both heads of the Watson-Crick automaton are at the beginning of $w'$. Since $q_0$ is the only state of $\mathcal{M}$ we conclude that $w'$ is also accepted by the automaton, i.e., $w' \in L(\mathcal{M})$.

To conclude, for any non-empty word $v \in L(\mathcal{M}) \setminus L$, there exists $w \in L$ such that $v = ww'$ and $w' \in L(\mathcal{M})$. By applying this process inductively, we obtain $v \in L^+$. □

Now, it seems natural to ask whether strongly deterministic stateless Watson-Crick automata can recognize the Kleene star of any prefix code.

**Theorem 11.** *Let $L \subseteq V^*$ be a finite prefix code. Then, there exists a strongly deterministic stateless Watson-Crick automaton recognizing the language $L^*$.*

*Proof*: Let $L = \{w_1, \ldots, w_n\} \subseteq V^*$, with $n \geq 1$, be a finite prefix code. We construct a stateless Watson-Crick automaton $\mathcal{M} = (V, \iota, \{q_0\}, q_0, \{q_0\}, P)$ where $P$ contains all the rewriting rules of the form $q_0 \begin{pmatrix} w_i \\ w_i \end{pmatrix} \rightarrow q_0$ with $1 \leq i \leq n$. It is easy to see that the automaton $\mathcal{M}$ accepts the language $L^*$. Moreover, since $L$ is a prefix code, the constructed Watson-Crick automaton is deterministic. □

Moreover, if we take for example $L = \{a^{2n}b^{2n} \mid n \geq 1\} \cup \{b^{2n}a^{2n} \mid n \geq 1\}$, then we can construct a strongly deterministic stateless Watson-Crick automaton recognizing $L^*$; due to page limitations we do not include this construction here. However, it is not known yet whether this is true for the Kleene closure of any infinite prefix code.

The next result gives a rather unexpected undecidability property.

**Theorem 12.** *Given a non-deterministic Watson-Crick automaton, it is undecidable whether any of its non-deterministic rules can be used in some computation.*

*Proof*: In order to prove this, we use the fact that it is undecidable whether a Turing machine accepts the empty word, see [3]. Given a deterministic Turing machine $T$, we construct a Watson-Crick automaton $\mathcal{M}$ which verifies whether the input is a valid sequence of consecutive configurations of the Turing machine starting from the empty tape. Since the Turing machine is deterministic, we can simulate each of its transitions with deterministic rewriting rules of the Watson-Crick automaton. The only non-deterministic rules of $\mathcal{M}$ occur at the moment when the Turing machine halts. The detailed description the Watson-Crick automaton is very technical and we do not include it here due to page limitations.

Intuitively, the Watson-Crick automaton $\mathcal{M}$ receives an input of the form $\begin{bmatrix} \sharp\ q_0\ \sharp\ u_1q_{i_1}v_1\ \sharp\ \ldots\ \sharp\ u_nq_{i_n}v_n\ \sharp \\ \sharp\ q_0\ \sharp\ u_1q_{i_1}v_1\ \sharp\ \ldots\ \sharp\ u_nq_{i_n}v_n\ \sharp \end{bmatrix}$, where $q_0$ is the initial state of the Turing machine, $q_{i_1}, \ldots, q_{i_n}$ are states of $T$ and $u_iv_i$ is the tape content at step $i$, where the reading head is on the first character of $v_i$. Then, the Watson-Crick automaton verifies in a deterministic way that for any $1 \leq j \leq n-1$, $u_{j+1}q_{i_{j+1}}v_{j+1}$ is a valid configuration obtained from $u_jq_{i_j}v_j$, by applying one of the deterministic rules of $T$. As soon as the Turing machine enters a final state, we let the Watson-Crick automaton finish reading

the rest of the input in a non-deterministic way. Thus, the Watson-Crick automaton uses any of its non-deterministic rules if and only if the Turing machine enters a final state and halts when started with the empty word as input. Since it is undecidable whether a given Turing machine accepts the empty word, it becomes also undecidable whether any of the non-deterministic rules of the Watson-Crick automaton is used. □

The previous result can be also reformulated as follows.

**Theorem 13.** *It is undecidable whether a given non-deterministic Watson-Crick automaton is weakly deterministic.*

As a consequence of the proof of Theorem 12, we get also an alternative proof for the known undecidability result of the emptiness problem for (non-)deterministic multihead automata. Indeed, in the proof of Theorem 12 we construct a Watson-Crick automaton which accepts an input if and only if it represents a valid sequence of consecutive configurations of a Turing machine starting from the empty tape. However, since it is undecidable whether a Turing machine accepts the empty word, we obtain in turn that it is undecidable whether the language recognized by the Watson-Crick automaton is empty or not. Moreover, the construction from the previous theorem can be easily made fully deterministic.

**Corollary 14.** *Given a (deterministic) Watson-Crick automaton $\mathcal{M}$, it is undecidable whether the recognized language $L(\mathcal{M})$ is empty.*

## 4 State complexity of Watson-Crick automata

It is well-known that Watson-Crick automata are more powerful than classical finite automata, see e.g., [10]. Moreover, it was shown in [9] that Watson-Crick automata recognize some regular languages in a more efficient manner. We devote this section to the study of state complexity of languages accepted by deterministic and non-deterministic Watson-Crick automata. For more details on state complexity, we refer the reader to [4] and [15]. Note that the transformation from [7] preserves the number of states so, when working with non-deterministic Watson-Crick automata we can suppose, without loss of generality, that the complementarity relation $\rho$ is actually the identity $\iota \subseteq V \times V$.

It is well-known that the state complexity of some families of finite languages is unbounded when we consider the finite automata recognizing them. However, as illustrated by our next result, this is not the case anymore when we consider the non-deterministic Watson-Crick automata recognizing them.

**Theorem 15.** *Any finite language can be recognized by a non-deterministic Watson-Crick automaton with two states.*

*Proof*: Let $L = \{w_1, \ldots, w_n\} \subset V^*$ be a finite language. We construct the Watson-Crick automaton $\mathcal{M} = (V, \iota, \{q_0, q_1\}, q_0, F, P)$, where $F = \{q_1\}$ and $P$ con-

tains rewriting rules of the form $q_0 \begin{pmatrix} w_i \\ w_i \end{pmatrix} \rightarrow q_1$, for all $1 \leq i \leq n$. Clearly, the language recognized by $\mathcal{M}$ is $L$. □

On the other hand, if we restrict to strongly deterministic Watson-Crick automata, then there exists a family of finite languages with unbounded state complexity.

**Theorem 16.** *For any $k \geq 2$ there exists a finite language $L_k \subseteq V^*$ such that any strongly deterministic Watson-Crick automaton recognizing $L_k$ needs at least $k$ states.*

*Proof*: Let $L_k = \{a^i \mid 1 \leq i \leq k-1\}$, with $k \geq 2$, and $\mathcal{M}$ be a strongly deterministic Watson-Crick automaton recognizing it. Since $L_k \subset \{a\}^*$, any rule of $\mathcal{M}$ is of the form $q_1 \begin{pmatrix} a^i \\ a^j \end{pmatrix} \rightarrow q_2$ for some $i, j \geq 0$. Moreover, since $\mathcal{M}$ is deterministic, for every state $q_1$ we can have at most one rewriting rule of the form mentioned above.

We claim now that in each final state $q_f$ we accept only one word. Otherwise, we have one of the following two cases: either $q_f$ occurs twice in some computation, or there exist two different rewriting sequences starting in the initial state $q_0$ and ending in $q_f$. The first case implies the existence of a cycle and thus the accepted language would not be finite. In the second case, we obtain that there exists a state $q$ from which we can continue in two different ways using two different rewriting rules, contradicting the determinism. Moreover, in the initial state we cannot accept any of the words from $L_k$ since otherwise we would again have a cycle.

Thus, a strongly deterministic Watson-Crick automaton accepting $L_k$ needs at least $k$ states: one initial and $k-1$ final ones. Moreover, clearly, such a deterministic Watson-Crick automaton can be easily constructed. □

On the other hand, if we look at Watson-Crick automata with non-injective complementarity relations, then the infinite hierarchy from the previous theorem collapses.

**Theorem 17.** *For any $k \geq 2$, the language $L_k = \{a^i \mid 1 \leq i \leq k-1\}$ can be recognized by a deterministic Watson-Crick automaton with two states and having a non-injective complementarity relation.*

*Proof*: Let $L_k = \{a^i \mid 1 \leq i \leq k-1\}$, for a given $k \geq 2$. We construct a deterministic Watson-Crick automaton $\mathcal{M} = (V, \rho, Q, q_0, F, P)$, where $V = \{a, b, c, d\}$, $\rho = \{(a,b), (b,a), (a,c), (c,a), (a,d), (d,a)\}$, $Q = \{q_0, q_1\}$, $F = \{q_1\}$, and the set $P$ of rewriting rules is

$$P = \left\{ q_0 \begin{pmatrix} a^{2i} \\ b^i c^i \end{pmatrix} \rightarrow q_1, \quad q_0 \begin{pmatrix} a^{2j-1} \\ b^{j-1} d c^{j-1} \end{pmatrix} \rightarrow q_1 \mid 1 \leq i \leq \left\lfloor \frac{k-1}{2} \right\rfloor, 1 \leq j \leq \left\lceil \frac{k-1}{2} \right\rceil \right\}.$$

Then, it is easy to see that $L(\mathcal{M}) = L_k$ and, moreover, $\mathcal{M}$ is deterministic. □

It is only natural now to ask whether also for the case of non-deterministic Watson-Crick automata there exists a family of languages with unbounded state complexity; this question was first stated in [9].

We show next that any unary regular language can be recognized by a (non-deterministic) Watson-Crick automaton with only three states. Actually, we prove this property for block non-deterministic finite automata, which can be considered a special case of Watson-Crick automata.

A *block non-deterministic finite automaton* (for short, block-NFA) is a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_F)$ where $\delta$ consists of a finite number of rules $(q_1, w, q_2)$, with $q_1, q_2 \in Q$ and $w \in \Sigma^*$. Clearly, a block-NFA is a special case of a Watson-Crick automaton where the two reading heads are required to always move together.

An arbitrary unary regular language is denoted by a regular expression of the form

$$a^{j_1} + \ldots + a^{j_{r-1}} + a^{j_r}(a^{i_1} + \ldots + a^{i_{s-1}})(a^m)^*, \tag{3}$$

$0 \leq j_1 < \ldots < j_{r-1} < j_r$, $0 \leq i_1 < \ldots < i_{s_1} < m$, $r, s \geq 0$. Here the words $a^{j_1}$, ..., $a^{j_{r-1}}$ are usually called the "tail" of the language and the remaining words belong to the cycle of length $m$.

**Theorem 18.** *Any unary regular language $L$ can be recognized by a block-NFA $\mathcal{A}$ having three states. Furthermore, $\mathcal{A}$ can be restricted to be unambiguous.*

*Proof*: Let $L$ be a unary regular language described by a regular expression of the form (3). We construct a block-NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_F)$ where $Q = \{q_0, q_1, q_2\}$, $Q_F = \{q_1, q_2\}$, and $\delta$ contains the rules

- $(q_0, a^x, q_1)$, where $x \in \{j_1, \ldots j_{r-1}\}$,
- $(q_0, a^x, q_2)$ where $x \in \{j_r + i_1, \ldots, j_r + i_{s-1}\}$,
- $(q_2, a^m, q_2)$.

It is easy to see that each word of $L$ is accepted by a unique computation of $\mathcal{A}$. □

The following result is an immediate consequence of Theorem 18.

**Corollary 19.** *Any regular unary language can be recognized by a non-deterministic Watson-Crick automaton with only three states.*

Note that the construction from the previous theorem can be slightly modified such that any unary regular language can be accepted by a deterministic Watson-Crick automaton with non-injective complementarity relation. However, it remains open whether we can still improve this state complexity, i.e., whether Watson-Crick automata with only two states can recognize all unary regular languages.

As illustrated by Theorem 15 and Corollary 19, some infinite hierarchies collapse when switching from the finite automata to the Watson-Crick automata recognizing them. However, in both cases, this is mainly due to the fact that Watson-Crick automata can read blocks of letters at each step. Thus, we investigate next what happens with infinite hierarchies of languages accepted by block-NFA's when we consider the Watson-Crick automata recognizing them.

**Theorem 20.** *For any $k \geq 1$, there exists a regular language $L_k$ such that any block-NFA recognizing $L_k$ needs more than $k$ states.*

*Proof*: Let $L_k = (10^*)^{k+1}1$ and assume that $L_k$ is recognized by a block-NFA $\mathcal{A}$ having $k$ states. Let $N$ be the length of the longest word appearing in rules of $\mathcal{A}$ and take $w_N = (10^N)^{k+1}1$. Since $w_N \in L_k$, $\mathcal{A}$ has an accepting computation $C$ on $w_N$. By the choice of $N$, the computation $C$ stops between the $i$th and $(i+1)$st occurrence of 1, for each $i = 1, \ldots, k+1$. Let $q_i$ be a state occurring in computation $C$ between the $i$th and $(i+1)$st occurrence of 1, $1 \leq i \leq k+1$. By the pigeon-hole-principle there exist $1 \leq j_1 < j_2 \leq k+1$ such that $q_{j_1} = q_{j_2}$. Thus $\mathcal{A}$ accepts also words having $k+2+l(j_2-j_1)$ occurrences of 1 for any $l \geq -1$; clearly, some of them are not in $L_k$. □

Considering the state complexity of languages, the previous result shows the existence of an infinite hierarchy. Although block-NFA's are a particular type of Watson-Crick automata, we show next that this hierarchy collapses when we look at the Watson-Crick automata accepting them.

**Theorem 21.** *For any $k \geq 1$, the language $L_k = (10^*)^{k+1}1$ can be recognized by a Watson-Crick automaton with three states.*

*Proof*: Let $\mathcal{M}_k = (V, \iota, \{q_0, q_1, q_2\}, q_0, F, P)$ be a Watson-Crick automaton, where $V = \{0, 1\}$, $F = \{q_2\}$, and $P$ contains the following productions:

- $q_0 \begin{pmatrix} \lambda \\ 1u \end{pmatrix} \to q_1$, for any word $u \in \{0,1\}^k$,
- $q_1 \begin{pmatrix} 0 \\ x \end{pmatrix} \to q_1$, where $x \in \{0, 1\}$,
- $q_1 \begin{pmatrix} 1 \\ \lambda \end{pmatrix} \to q_1$, and $q_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} \to q_2$.

With the first production, we advance the lower head $k+1$ characters, independently of what we have on the tape. Then, the upper head will catch up this advance only after reading $k+1$ times the character 1. Then, the automaton enters the final state after reading the last character 1. Since a word is accepted only when both reading heads have completely parsed the input word, the language recognized by $\mathcal{M}$ is $L_k$. □

The results of this section illustrate the fact that there are many complex languages which can be accepted by Watson-Crick automata with a bounded number of states. Although we do not include a formal proof in this paper, another such complex family of languages accepted by Watson-Crick automata with only a small number of states is $L_k = 10^+1^20^+ \ldots 0^+1^{2^k}$, for any $k \geq 2$. However, it remains open whether for all $k > 1$ there exists a language $L_k$ such that any non-deterministic Watson-Crick automaton accepting $L_k$ needs at least $k$ states [9].

## Acknowledgments.

This research was supported by Natural Sciences and Engineering Research Council of Canada Discovery Grant, UWO Faculty of Science Grant, and Canada Research Chair Award to L.K. and Natural Sciences and Engineering Research Council of Canada Discovery Grant to K.S.

## References


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

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

[3] J.E. Hopcroft, J. D. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley, Reading, MA, 1979.

[4] J. Hromkovic, *Descriptional complexity of finite automata: Concepts and open problems*, J. ALC 7, 519-531, 2002.

[5] O.H. Ibarra, B. Ravikumar, *On partially blind multihead finite automata*, Theoret. Comput. Sci. 356, 190–199, 2006.

[6] H. Jürgensen, S. Konstantinidis, *Codes*, In: Handbook of Formal Languages, Vol. I (G. Rozenberg, A. Salomaa, Eds.), Springer, 511-607, 1997.

[7] D. Kuske, P. Weigel, *The Role of the Complementarity Relation in Watson-Crick Automata and Sticker Systems*, DLT 2004, LNCS, **3340**, 272-283, (2004).

[8] C. Martín-Vide, Gh. Păun, *Normal Forms for Watson-Crick Finite Automata*, in F. Cavoto, ed., The Complete Linguist: A Collection of Papers in Honour of Alexis Manaster Ramer: 281-296. Lincom Europa, Munich, 2000.

[9] A. Păun, M. Păun, *State and transition complexity of Watson-Crick finite automata*, In: Fundamentals of Computation Theory, FCT'99, G. Ciobanu and G. Păun, Eds., LNCS 1684, 409-420, 1999.

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

[11] E. Petre, *Watson-Crick $\omega$-Automata*, J. Autom. Lang. Comb. **8(1)**, 59-70, 2003.

[12] D. Röthlisberger et. al., *Kemp elimination catalysts by computational enzyme design*, Nature **453**, 190-195, 2008.

[13] E. Shapiro, Y. Benenson, *Bringing DNA computers to life*, Scientific American, 294, 44-51, 2006.

[14] K. Wagner, G. Wechsung, Computational Complexity, D. Reidel Publishing Company, 1986.

[15] S. Yu, *State complexity of finite and infinite regular languages,* Bull. EATCS 76, 142-152, 2002.