# Parallel Communicating Watson-Crick Automata Systems

Elena Czeizler* and Eugen Czeizler*

**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 introduce the concept of parallel communicating Watson-Crick automata systems. It consists of several Watson-Crick finite automata parsing independently the same input and exchanging information on request, by communicating states to each other. We investigate the computational power of these systems and prove that they are more powerful than classical Watson-Crick finite automata, but still accepting at most context-sensitive languages. Moreover, if the complementarity relation is injective, then we obtain that this inclusion is strict. For the general case, we also give some closure properties, as well as a characterization of recursively enumerable languages based on these systems.

## 1 Introduction

Watson-Crick finite automata, introduced in [5], are a counterpart of finite automata working on double stranded sequences. As suggested by the name, these automata are mainly inspired from molecular computing and are intended as a formalization of DNA manipulation. The two strands of the input are separately scanned from left to right by read only heads controlled by a common state. The characters on the corresponding positions from the two strands are linked by a complementarity relation, inspired from the Watson-Crick complementarity of DNA nucleotides. Several variants of these automata were investigated in [11, 12, 13, 15], see also [14] for a comprehensive presentation.

Distributed computations play a major role in modern computer science; multiprocessor computers, distributed data bases, computer networks, etc., introduced notions such as distribution, parallelism, and communication. The theory of grammar and automata systems was developed as a mathematical model for distributed and parallel computations.

*Department of Mathematics, University of Turku and Turku Centre for Computer Science, Turku 20520, Finland, E-mail: `elena.czeizler@utu.fi, euczei@utu.fi`

An automata system is a set of automata working together on the same input, according to a well specified protocol, in order to accept one language. There are two basic classes of automata systems: *sequential* and *parallel*.

The sequential class is represented by *cooperating distributed* automata systems. Here, all components work on a common input tape and at each step only one automaton is active. An example of such systems is the *cooperating distributed push-down automata system*, introduced and studied in [3].

A *parallel communicating automata system* is a construct consisting of several automata working synchronously, each on its own input tape, and communicating on request. Special *query states* are provided, each of them pointing to exactly one component of the system. When a component $i$ of the system reaches a query state $K_j$, the current state of the component $j$ will be communicated to $i$ and the computation continues. There are two important classifications of parallel communicating systems concerning the *communication graph* and the *returning feature*. An automata system is called *centralized* if only one component, the *master*, may introduce query states, and *non-centralized* otherwise. An automata system is called *returning* if after communicating, a component resumes the computation from its initial state, and *non-returning* if it remains in its current state. There are several papers in the literature investigating this class of systems. For example, parallel communicating push-down automata systems communicating by stacks were introduced in [2] and parallel communicating automata systems communicating by states were introduced in [10], see also [9] for a survey.

Cooperating distributed Watson-Crick automata systems were investigated in [1], where it was proved that distribution does not bring any change in the acceptance power of Watson-Crick finite automata, except for the case of one state automata, i.e. *stateless Watson-Crick automata*.

In this paper we introduce the notion of *parallel communicating Watson-Crick automata system* as a set of Watson-Crick finite automata working independently on their own input tape and communicating states on request. We consider only non-centralized and non-returning systems. Although every component has its own double-stranded tape, the input is the same on all of them. At the beginning, all components are in their initial states and start parsing synchronously the input from left to right. The communication between components is done using query states as described before for general parallel communicating automata systems. An input is accepted by the system if all components are in final states when they completely parsed the tape. Moreover, if one of the components stops before the others, the system halts and rejects the input. Hence, in order to accept, the components either finish at the same time or wait for each other at the end of the computation.

Combining the notions of Watson-Crick automata and parallel communicating systems comes naturally due to the new developments in DNA manipulation techniques. While classical Watson-Crick finite automata use just one of the essential features of DNA, i.e. the *Watson-Crick complementarity*, the systems introduced here open new possibilities in exploiting also the *massive parallelism* of DNA computations.

The structure of the paper is as follows. In Section 2 we fix our terminology and introduce some basic notions and results. Section 3 is devoted to the computational power of these systems. We start by giving an example of a parallel communicating Watson-Crick automata system proving that the accepting power is enhanced. We also prove that the languages accepted by these systems are at most context-sensitive. Moreover, if the complementarity relation is injective, as in the case of DNA nucleotides, then one letter-languages accepted by these systems are regular. In Section 4 we investigate some closure properties. We also give a characterization of recursively enumerable languages based on these systems. In Section 5 we present some open problems.

## 2 Preliminaries

In this section we give basic definitions and some already known results we need later on. We start by considering the classical Watson-Crick finite automata introduced in [5] and then define the parallel communicating version. We assume that the reader is familiar with the fundamental concepts from formal languages and automata theory. For more details we refer to [7], [14], and [16].

For a finite set $Q$, let $card(Q)$ and $2^Q$ denote the cardinality and the power set of $Q$, respectively. Let $V$ be a finite alphabet. We denote by $V^*$ the set of all finite words over $V$, by $\lambda$ the empty word, and by $V^+$ the set of all nonempty finite words over $V$, $V^+ = V^* \backslash \{\lambda\}$. For $w \in V^*$ we denote by $|w|$ the length of $w$.

Given two alphabets $V$ and $U$, we define a *morphism* as a function $h : V \to U^*$, extended to $h : V^* \to U^*$ by $h(\lambda) = \lambda$ and $h(w_1 w_2) = h(w_1)h(w_2)$, for $w_1, w_2 \in V^*$. If $h(a) \neq \lambda$ for each $a \in V$, then we say that $h$ is a *$\lambda$-free* morphism. We define a *projection* associated to the alphabet $V$ as the morphism $pr_V : (V \cup U)^* \to V^*$ such that $pr_V(a) = a$ for all $a \in V$ and $pr_V(a) = \lambda$ otherwise. For two morphisms $h_1, h_2 : V^* \to U^*$, we define the *equality set* of $h_1, h_2$ as:

$$EQ(h_1, h_2) = \{w \in V^* \mid h_1(w) = h_2(w)\}.$$

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. We say that $\rho$ is injective if any $a \in V$ has a unique complementary symbol $b \in V$ with $(a, b) \in \rho$. In accordance with the representation of DNA molecules, viewed as two strings written one over the other, we write $\begin{pmatrix} V^* \\ V^* \end{pmatrix}$ instead of $V^* \times V^*$ and an element $(w_1, w_2) \in V^* \times V^*$ as $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$.

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

$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \ldots \begin{bmatrix} a_n \\ b_n \end{bmatrix} \in WK_\rho(V)$ can be also written in a more compact form as $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, where $w_1 = a_1 a_2 \ldots a_n$ and $w_2 = b_1 b_2 \ldots b_n$.

The essential difference between $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ and $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ is that $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ 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 *Watson-Crick finite automaton* is a 6-tuple $\mathcal{M} = (V, \rho, Q, q_0, F, \delta)$, 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,
- $\delta : Q \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to 2^Q$ is a mapping, called the *transition function*, such that $\delta(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) \neq \emptyset$ only for finitely many triples $(q, w_1, w_2) \in Q \times V^* \times V^*$.

We can replace the transition function with rewriting rules, by using

$$s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s' \text{ instead of } s' \in \delta(s, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}).$$

We define transitions in a Watson-Crick finite automaton as follows. For $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \in \begin{pmatrix} V^* \\ V^* \end{pmatrix}$ such that $\begin{bmatrix} v_1 u_1 w_1 \\ v_2 u_2 w_2 \end{bmatrix} \in WK_\rho(V)$ and $s, s' \in Q$ we write

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} s \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Rightarrow \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} s' \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

if and only if $s' \in \delta(s, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix})$. If we denote by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$, then the language accepted by a Watson-Crick automaton is:

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

$$\text{and } \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}.$$

Hence, a word $w_1$ is accepted by $\mathcal{M}$ if starting from the initial state, after parsing the whole input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ we are in a final state.

Let us continue now by defining *parallel communicating Watson-Crick automata systems.*

A parallel communicating Watson-Crick automata system of degree $n$, denoted by $PCWK(n)$, is an $(n+3)$-tuple

$$\mathcal{A} = (V, \rho, A_1, A_2, \ldots, A_n, K),$$

where

- $V$ is the input alphabet;
- $\rho$ is the complementarity relation;
- $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$, $1 \le i \le n$, are Watson-Crick finite automata, where the sets $Q_i$ are not necessarily disjoint;
- $K = \{K_1, K_2, \ldots, K_n\} \subseteq \cup_{i=1}^{n} Q_i$ is the set of query states.

The automata $A_1, A_2, \ldots, A_n$ are called the *components* of the system $\mathcal{A}$. Note that any Watson-Crick finite automaton can be viewed as a parallel communicating Watson-Crick automata system of degree 1.

A configuration of a parallel communicating Watson-Crick automata system is a $2n$-tuple $(s_1, \binom{u_1}{v_1}, s_2, \binom{u_2}{v_2}, \ldots, s_n, \binom{u_n}{v_n})$ where $s_i$ is the current state of the component $i$ and $\binom{u_i}{v_i}$ is the part of the input word which has not been read yet by the component $i$, for all $1 \le i \le n$. We define a binary relation $\vdash$ on the set of all configurations by setting

$$(s_1, \binom{u_1}{v_1}, s_2, \binom{u_2}{v_2}, \ldots, s_n, \binom{u_n}{v_n}) \vdash (r_1, \binom{u'_1}{v'_1}, r_2, \binom{u'_2}{v'_2}, \ldots, r_n, \binom{u'_n}{v'_n})$$

if and only if one of the following two conditions holds:

- $K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset$, $\binom{u_i}{v_i} = \binom{x_i}{y_i}\binom{u'_i}{v'_i}$, and $r_i \in \delta_i(s_i, \binom{x_i}{y_i})$, $1 \le i \le n$;
- for all $1 \le i \le n$ such that $s_i = K_{j_i}$ and $s_{j_i} \notin K$ we have $r_i = s_{j_i}$, whereas for all the other $1 \le l \le n$ we have $r_l = s_l$. In this case $\binom{u'_i}{v'_i} = \binom{u_i}{v_i}$, for all $1 \le i \le n$.

If we denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$, then the language recognized by a parallel communicating Watson-Crick automata system $\mathcal{A}$ is defined as:

$$L(\mathcal{A}) = \{w_1 \in V^* \mid (q_1, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, q_2, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \ldots, q_n, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}) \vdash^*$$
$$(s_1, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, s_2, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, \ldots, s_n, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}),\ s_i \in F_i,\ 1 \leq i \leq n\}.$$

Intuitively, the language accepted by such a system consists of all words $w_1$ such that in every component we reach a final state after reading all input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$.

In the next section we study the connection between the family of languages accepted by parallel communicating Watson-Crick automata systems and the class of *context-sensitive* languages. For doing this we use a special type of automata characterizing this class of languages.

*Linearly bounded automata* are a special class of Turing machines which have two extra symbols in their input alphabet, say # and \$, called the left and right end-markers, respectively. The automaton can neither overwrite these markers nor move left or right from them. Hence, this type of automata uses only a limited amount of tape. Similarly, *$k$-head linearly bounded automata* are a special class of $k$-tape Turing machines which use only limited amount of each tape. On each step the $k$ heads move independently, according to the state of the automaton and the symbol read on each individual tape.

The following two results are well known in the literature, see for example [6] and [7].

**Theorem 1.** *$L \subseteq V^*$ is a context-sensitive language if and only if it is accepted by a linearly bounded automaton.*

We say that an automaton $A$ is of *space complexity* $S(n)$ if, for every accepted input of length $n$ there is some accepting computation in which at most $S(n)$ tape-cells are parsed by any read-write head.

**Theorem 2.** *If a language $L$ is accepted by a $k$-tape Turing machine of space complexity $S(n)$, then $L$ is accepted by some one-tape Turing machine of the same space complexity.*

The following lemma comes as a direct consequence of the previous two results and will be used in our future considerations.

**Lemma 3.** *A language $L$ is context-sensitive if and only if it is accepted by a $k$-head linearly bounded automaton.*

## 3 Computational power

Let us start by giving an example of a language accepted by a parallel communicating Watson-Crick automata system of degree 3.

| | | |
|---|---|---|
| $\delta_1(q_1, \binom{x}{\lambda}) = q_1$ | $\delta_2(q_2, \binom{x}{x}) = q_2$ | $\delta_3(q_3, \binom{x}{x}) = q_3$ |
| $\delta_1(q_1, \binom{\#}{\lambda}) = r_1$ | $\delta_2(q_2, \binom{\#}{\#}) = p_1$ | $\delta_3(q_3, \binom{\#}{\#}) = s_1$ |
| $\delta_1(r_1, \binom{x}{\lambda}) = r_1$ | $\delta_2(p_1, \binom{x}{\lambda}) = p_1$ | $\delta_3(s_1, \binom{x}{x}) = s_1$ |
| $\delta_1(r_1, \binom{\#}{\lambda}) = r_2$ | $\delta_2(p_1, \binom{\#}{\lambda}) = p_2$ | $\delta_3(s_1, \binom{\#}{\#}) = s_2$ |
| $\delta_1(r_2, \binom{x}{\lambda}) = r_2$ | $\delta_2(p_2, \binom{x}{\lambda}) = p_2$ | $\delta_3(s_2, \binom{x}{\lambda}) = s_2$ |
| $\delta_1(r_2, \binom{\#}{\lambda}) = r_3$ | $\delta_2(p_2, \binom{\#}{\lambda}) = p_3$ | $\delta_3(s_2, \binom{\#}{\lambda}) = s_3$ |
| $\delta_1(r_3, \binom{x}{\lambda}) = r_3$ | $\delta_2(p_3, \binom{x}{\lambda}) = p_3$ | $\delta_3(s_3, \binom{x}{x}) = s_3$ |
| $\delta_1(r_3, \binom{\#}{\lambda}) = r_4$ | $\delta_2(p_3, \binom{\#}{\lambda}) = p_4$ | $\delta_3(s_3, \binom{\#}{\#}) = f_3$ |
| $\delta_1(r_4, \binom{x}{\lambda}) = r_4$ | $\delta_2(p_4, \binom{x}{x}) = p_4$ | $\delta_3(f_3, \binom{y}{z}) = f_3$ |
| $\delta_1(r_4, \binom{\#}{\lambda}) = r_5$ | $\delta_2(p_4, \binom{\#}{\#}) = f_2$ | |
| $\delta_1(r_5, \binom{x}{x}) = r_5$ | $\delta_2(f_2, \binom{y}{z}) = f_2$ | |
| $\delta_1(r_5, \binom{\lambda}{\#}) = f_1$ | | |
| $\delta_1(f_1, \binom{\lambda}{y}) = f_1$ | | |

with $x \in \{a, b\}$, $y, z \in \{a, b, \#, \lambda\}$

Table 1: The transition functions of Example 4

**Example 4.** Let $\mathcal{A} = (\{a, b, \#\}, \rho, A_1, A_2, A_3, \emptyset)$, where $\rho$ is the identity relation, i.e., $\rho = \{(a, a), (b, b), (\#, \#)\}$, $A_1 = (\{a, b, \#\}, \rho, \{q_1, r_1, r_2, r_3, r_4, r_5, f_1\}, q_1, \{f_1\}, \delta_1)$, $A_2 = (\{a, b, \#\}, \rho, \{q_2, p_1, p_2, p_3, p_4, f_2\}, q_2, \{f_2\}, \delta_2)$, and $A_3 = (\{a, b, \#\}, \rho, \{q_3, s_1, s_2, s_3, f_3\}, q_3, \{f_3\}, \delta_3)$. The transition functions of the three components are defined in Table 1.

The system works as follows. The first component verifies that the input is of the form $\begin{bmatrix} w_1\#w_2\#w_3\#w_4\#w_5\#w_6 \\ w_1\#w_2\#w_3\#w_4\#w_5\#w_6 \end{bmatrix}$ with $w_i \in \{a, b\}^+$ for all $1 \leq i \leq 6$, and moreover $w_1 = w_6$. Simultaneously the second and the third component impose the constraints $w_2 = w_5$ and $w_3 = w_4$, respectively. Thus, the language accepted by $\mathcal{A}$ is $L = \{w_1\#w_2\#w_3\#w_3\#w_2\#w_1 \mid w_1, w_2, w_3 \in \{a, b\}^*\}$.

On the other hand, it was proved in [18] that the language $L$ cannot be accepted by a 2-head finite automaton. Since Watson-Crick automata are equivalent with 2-head finite automata, see [14], we have the following result.

**Theorem 5.** *Parallel communicating Watson-Crick automata systems are more powerful than Watson-Crick finite automata.*

$$\begin{array}{ll}
\delta_1(q_1, \binom{xy}{z}) = q_1 \text{ for any } x, y, z \in V & \delta_2(q_2, \binom{xy}{\lambda}) = q_2 \text{ for any } x, y \in V \\
\delta_1(q_1, \binom{\#}{x}) = q_x \text{ for any } x \in V & \delta_2(q_2, \binom{\#}{\lambda}) = K_1 \\
\delta_1(q_x, \binom{\lambda}{\lambda}) = K_2 & \delta_2(q_x, \binom{\lambda}{x}) = q_3 \text{ for any } x \in V \\
\delta_1(q_3, \binom{\lambda}{x}) = q_x \text{ for any } x \in V & \delta_2(q_3, \binom{\lambda}{\lambda}) = K_1 \\
\delta_1(q_3, \binom{\lambda}{\#}) = q_{f_1} & \delta_2(q_{f_1}, \binom{\lambda}{x}) = q_{f_2} \text{ for any } x \in V \\
\delta_1(q_{f_1}, \binom{\lambda}{\lambda}) = q_{f_1} & \delta_2(q_{f_2}, \binom{\lambda}{x}) = q_{f_2} \text{ for any } x \in V \cup \{\#\}
\end{array}$$

Table 2: The transition functions of Example 6

Next, let us illustrate the communication between components by considering a parallel communicating Watson-Crick automata system accepting the non context-free language $L = \{ww\# \mid w \in V^+\}$, where $\# \notin V$ and $|V| \geq 2$.

**Example 6.** Let $\mathcal{A} = (V \cup \{\#\}, \rho, A_1, A_2, K)$ be a parallel communicating Watson-Crick automata system where $\rho = \{(a, a) \mid a \in V\} \cup \{(\#, \#)\}$, $K = \{K_1, K_2\}$, $A_1 = (V \cup \{\#\}, \rho, Q_1, q_1, \{q_{f_1}\}, \delta_1)$, and $A_2 = (V \cup \{\#\}, \rho, Q_2, q_2, \{q_{f_2}\}, \delta_2)$. The sets of states are $Q_1 = \{q_1, q_3, q_{f_1}, K_2\} \cup \{q_x \mid x \in V\}$ and $Q_2 = \{q_2, q_3, q_{f_1}, q_{f_2}, K_1\} \cup \{q_x \mid x \in V\}$, while the transition functions are defined in Table 2.

The first component finds the middle of the input word, by placing the two reading heads one at the end and the other in the middle of the input word. In parallel, to preserve the synchronization, the second component moves one reading head to the end of the input while the other one remains unmoved. At the same time we also check that the input has odd length. Then, by using communication between components we check letter by letter that the input is indeed of the form $\begin{bmatrix} ww\# \\ ww\# \end{bmatrix}$.

A natural question regarding these systems is the relation between the languages they accept and the family of context-sensitive languages.

We first need a generalization of a result already known for Watson-Crick finite automata, see [14].

**Lemma 7.** *Every parallel communicating Watson-Crick automata system is equivalent with a system where in every component we have only rules of the form* $s\binom{w_1}{w_2} \to \binom{w_1}{w_2} s'$*, with* $|w_1 w_2| \leq 1$.

*Proof.* Let $\mathcal{A} = (V, \rho, A_1, \ldots, A_n, K)$ be a parallel communicating Watson-Crick automata system with $n$ components, where $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$ for all $1 \leq i \leq n$. Let us first index by unique labels all transitions from all components and define the constant $m = max\{|w_1| + |w_2| \mid s\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$ is a production in one of the components of the system$\}$.

We construct a parallel communicating Watson-Crick automata system $\mathcal{A}' = (V, \rho, A'_1, \ldots, A'_n, K)$, where $A'_i = (V, \rho, Q'_i, q_i, F_i, \delta'_i)$ are obtained from $A_i$ as follows.

Let $s\begin{pmatrix} w_1 \ldots w_p \\ w'_1 \ldots w'_{p'} \end{pmatrix} \to \begin{pmatrix} w_1 \ldots w_p \\ w'_1 \ldots w'_{p'} \end{pmatrix} s'$, with $w_1, \ldots, w_p, w'_1, \ldots, w'_{p'} \in V$ be a transition rule from $A_i$, indexed with the unique label $j$. Then, in $A'_i$ we introduce $m$ new states $r^j_1, r^j_2, \ldots, r^j_m$ and the following transitions:

$$s\begin{pmatrix} w_1 \\ \lambda \end{pmatrix} \to \begin{pmatrix} w_1 \\ \lambda \end{pmatrix} r^j_1, \quad \ldots, \quad r^j_{p-1}\begin{pmatrix} w_p \\ \lambda \end{pmatrix} \to \begin{pmatrix} w_p \\ \lambda \end{pmatrix} r^j_p,$$

$$r^j_p\begin{pmatrix} \lambda \\ w'_1 \end{pmatrix} \to \begin{pmatrix} \lambda \\ w'_1 \end{pmatrix} r^j_{p+1}, \quad \ldots, \quad r^j_{p+p'-1}\begin{pmatrix} \lambda \\ w'_{p'} \end{pmatrix} \to \begin{pmatrix} \lambda \\ w'_{p'} \end{pmatrix} r^j_{p+p'},$$

$$r^j_{p+p'}\begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \to \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} r^j_{p+p'+1}, \quad \ldots, \quad r^j_m\begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \to \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} s'.$$

Thus, any transition from $A_i$ is replaced in $A'_i$ by $m + 1$ transitions of the form requested by the lemma. Also, since this construction preserves the synchronization between components, the system $\mathcal{A}'$ recognizes the same language as $\mathcal{A}$ but with linear time delay. □

**Theorem 8.** *The family of languages accepted by parallel communicating Watson-Crick automata systems is included in the family of context-sensitive languages.*

*Proof.* We can assume without loss of generality, that all components of the parallel communicating Watson-Crick automata system have only rules of the form described in Lemma 7. Then, for any such system $\mathcal{A}$ of degree $n$ we can construct a $2n$-tape linearly bounded automaton $\mathcal{M}$ which recognizes the same language. Each $2p + i$ tape, with $0 \le p \le n - 1$ and $1 \le i \le 2$, simulates the $i$-th tape of the $(p + 1)$-th component of $\mathcal{A}$. All the states in $\mathcal{M}$, except the final one, encode information about the states of all $n$ components of system $\mathcal{A}$. At each computational step, we read a character on each tape and either move the reading head one step to the right or remain on the same position, according to the evolution of system $\mathcal{A}$. For query steps, we just modify the information encoded in the state, i.e., we enter a new state in $\mathcal{M}$, whereas the input and positions of the reading heads remain unchanged. The final state of $\mathcal{M}$ is reached only from states encoding the information that all components of system $\mathcal{A}$ are in final states and all the $2n$ reading heads are positioned on the right end marker.

From this construction we obtain that automaton $\mathcal{M}$ accepts the same language as system $\mathcal{A}$. Hence, from Lemma 3, we obtain that $L(\mathcal{A})$ is a context-sensitive language. □

So far we considered only the general case where the complementarity relation $\rho$ has no restrictions, except for symmetry. However, in [17] the case of an injective complementarity relation inspired by the real Watson-Crick complementarity of DNA nucleotides was discussed. For the rest of this section we restrict ourselves to this particular case. In order to investigate the computational power of these systems, we relate them to *k-head automata*, as they were defined in [10].

A *k-head automaton* is a 6-tuple $\mathcal{M} = (k, Q, V, f, q_0, F)$ where $Q$ is the set of states, $V$ is the input alphabet, $f : Q \times (V \cup \{\lambda\})^k \to 2^Q$ is the transition function, $q_0$ is the initial state, and $F \subseteq Q$ is the set of final states. Any computation starts in the initial state and with all the reading heads on the leftmost character of the input. Then, for any transition $q \in f(s, a_1, a_2, \ldots, a_k)$ and all $1 \leq i \leq k$, the $i$-th head reads $a_i$ from the input tape and the automaton passes from state $s$ into state $q$. A word $w$ is accepted if after finitely many moves the automaton enters a final state, the input being completely read by all heads. In all the other cases the input word is rejected.

**Theorem 9.** *Any language recognized by a parallel communicating Watson-Crick automata system of degree n, with injective complementarity relation, can be also recognized by a 2n-head automaton.*

*Proof.* For the clarity of the proof, we consider only systems of degree 2, whereas the reasoning remains the same for the general case.

Let $\mathcal{A} = (V, \rho, A_1, A_2, K)$ be a parallel communicating Watson-Crick automata system of degree 2, accepting the language $L \subseteq V^*$, where $A_1 = (V, \rho, Q_1, q_1, F_1, \delta_1)$, $A_2 = (V, \rho, Q_2, q_2, F_2, \delta_2)$, and $K = \{K_1, K_2\}$. Since the relation $\rho$ is injective, we can take it to be the identity relation; thus all components have on both tapes the same word $w \in V^*$. Also, by Lemma 7 we can suppose that in every component we have only rules of the form $s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$, with $|w_1 w_2| \leq 1$.

Let us construct now a 4-head automaton $\mathcal{M} = (4, Q, V, f, q_0, F)$ where $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$, $F = F_1 \times F_2$, and the transition function $f$ is as follows:

- $f((p,q), w_1, w_2, w_3, w_4) = (p_1, q_1)$ whenever $p, q \notin K$, $\delta_1(p, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = p_1$, and $\delta_2(q, \begin{pmatrix} w_3 \\ w_4 \end{pmatrix}) = q_1$;
- $f((K_2, q), \lambda, \lambda, \lambda, \lambda) = (q, q)$;
- $f((p, K_1), \lambda, \lambda, \lambda, \lambda) = (p, p)$.

At any step the automaton $\mathcal{M}$ simulates the corresponding moves of the two components of $\mathcal{A}$. If the components are not in a query state and they read $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ and $\begin{pmatrix} w_3 \\ w_4 \end{pmatrix}$ respectively from the input tape, with $|w_1 w_2| \leq 1$ and $|w_3 w_4| \leq 1$, then in $\mathcal{M}$ each head reads $w_1, w_2, w_3$, and $w_4$, respectively, and it enters into the corresponding state. Otherwise, i.e., we are in a state $(K_2, s)$ or $(s, K_1)$, we just simulate the query by entering state $(s, s)$ and leaving the input unchanged. Since a word is accepted by $\mathcal{M}$ only if it is in a final state when all the reading heads have finished parsing the input, then $w \in L(\mathcal{A})$ implies $w \in L(\mathcal{M})$ and hence $L(\mathcal{A}) \subseteq L(\mathcal{M})$.

Let now $w$ be a word accepted by $\mathcal{M}$. From the construction of the transition function $f$, each computational step in $\mathcal{M}$ can be translated into a computational step in $\mathcal{A}$ when we consider the input $\begin{bmatrix} w \\ w \end{bmatrix}$. Moreover after the final computational step, all 4 heads of $\mathcal{M}$ have completely read the input and the automaton is in a final state. This implies that at the same step both components of system $\mathcal{A}$ are in final states, while their reading heads from the lower and from the upper strands have completely parsed the input. So, we have $w \in L(\mathcal{A})$ and hence $L(\mathcal{M}) \subseteq L(\mathcal{A})$. $\square$

**Observation.** The equivalence between Watson-Crick automata and 2-head automata is proved in [14] regardless of the structure of the complementarity relation using a similar construction as above. In their case, the second head of the 2-head automaton "guesses" the complement of the character read from the input tape, and simulates the corresponding move from the Watson-Crick automaton. However, in our proof, the injectivity of the complementarity relation plays an important role. If the complementarity relation would not be injective, then for all positions $i$ of the input word, several reading heads would have to guess exactly the same complement but at different time steps. However, by definition this constraint cannot be imposed. Hence, the injectivity of the complementarity relation is a necessary condition in Theorem 9.

It is known from [10] that $k$-head automata are equivalent with parallel finite automata systems with $k$ components and communicating by states. Moreover, it is proved in [8] that the languages accepted by multihead nondeterministic pushdown automata satisfy the semilinearity property. Hence, parallel finite automata systems communicating by states accept only semilinear languages. Since any semilinear language over an one-letter alphabet is regular, we have the following result.

**Corollary 10.** *Every one-letter language accepted by a parallel communicating Watson-Crick automata system with injective complementarity relation is regular.*

Recently, it was proved in [4] that on one-letter alphabets, parallel communicating Watson-Crick automata system with non-injective complementarity relation accept also some non-regular languages, e.g. $L = \{a^{n^2} \mid n \geq 2\}$.

## 4 Closure properties

In this section we consider some closure properties of the family of languages accepted by parallel communicating Watson-Crick automata systems. From now on, $V$ is the input alphabet and $\# \notin V$ is a special character not included in it; let $V' = V \cup \{\#\}$. We also extend the complementarity relation by adding $(\#, \#) \in \rho$.

**Theorem 11.** *Let $L_1, L_2 \subseteq V^*$ be two languages accepted by some parallel communicating Watson-Crick automata systems of degrees $n_1$ and $n_2$, respectively, using the same complementarity relation. Then the language $(L_1\#)\bigcap(L_2\#)$ is also accepted by a system of degree $n_1 + n_2$.*

*Proof.* Let $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$, where

$$\mathcal{A}_1 = (V, \rho, A_1, \dots, A_{n_1}, K), \text{ with } A_i = (V, \rho, Q_i, q_i, F_i, \delta_i) \text{ and}$$

$$\mathcal{A}_2 = (V, \rho, A'_1, \dots, A'_{n_2}, K'), \text{ with } A'_i = (V, \rho, Q'_i, q'_i, F'_i, \delta'_i).$$

We construct a new system of degree $n_1 + n_2$

$$\mathcal{A} = (V', \rho, \overline{A}_1, \dots, \overline{A}_{n_1}, \overline{A}'_1, \dots, \overline{A}'_{n_2}, K \cup K'), \text{ where}$$

- $\overline{A}_1 = (V', \rho, Q_1 \cup \{q_1^f, q_1^v\} \cup \{v_i^{i-1} \mid 2 \leq i \leq n_1\} \cup \{K_2, \dots, K_{n_1}\}, q_1, \{q_1^f\}, \overline{\delta}_1)$,
- $\overline{A}_i = (V', \rho, Q_i \cup \{q_i^f, q_i^v\} \cup \{v_i^j \mid 1 \leq j \leq n_1 - 1\}, q_i, \{q_i^f\}, \overline{\delta}_i),\ 2 \leq i \leq n_1$
- $\overline{A}'_1 = (V', \rho, Q'_1 \cup \{q_1'^f, q_1'^v\} \cup \{v_i'^{i-1} \mid 2 \leq i \leq n_2\} \cup \{K'_2, \dots, K'_{n_2}\}, q'_1, \{q_1'^f\}, \overline{\delta}'_1)$,
- $\overline{A}'_i = (V', \rho, Q'_i \cup \{q_i'^f, q_i'^v\} \cup \{v_i'^j \mid 1 \leq j \leq n_2 - 1\}, q'_i, \{q_i'^f\}, \overline{\delta}'_i),\ 2 \leq i \leq n_2$.

The components $\overline{A}_i$ and $\overline{A}'_i$ are obtained from $A_i$ and $A'_i$, respectively, by adding some states and some new transition rules, as follows:

(i) for all $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$: $\overline{\delta}_i(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = \delta_i(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix})$ and $\overline{\delta}'_j(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = \delta'_j(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix})$,

(ii) for all $1 \leq i \leq n_1$: $\overline{\delta}_i(s, \begin{pmatrix} \# \\ \# \end{pmatrix}) = q_i^v$, for any $s \in F_i$, $\overline{\delta}_i(q_i^f, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i^f$,

(iii) $\overline{\delta}_1(q_1^v, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_2$, $\overline{\delta}_1(v_i^{i-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_{i+1}$, for all $2 \leq i \leq n_1 - 1$, and $\overline{\delta}_1(v_{n_1}^{n_1-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_1^f$,

(iv) for all $2 \leq i \leq n_1$: $\overline{\delta}_i(q_i^v, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = v_i^1$, $\overline{\delta}_i(v_i^j, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = v_i^{j+1}$, for $1 \leq j \leq n_1 - 2$, and $\overline{\delta}_i(v_i^{n_1-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i^f$,

(v) for all $1 \leq i \leq n_2$: $\overline{\delta}'_i(s, \begin{pmatrix} \# \\ \# \end{pmatrix}) = q_i'^v$, for any $s \in F'_i$, $\overline{\delta}'_i(q_i'^f, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i'^f$,

(vi) $\overline{\delta}'_1(q'^v_1, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = K'_2$, $\overline{\delta}'_1(v'^{i-1}_i, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = K'_{i+1}$, for all $2 \leq i \leq n_2 - 1$, and $\overline{\delta}'_1(v'^{n_2-1}_{n_2}, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = q'^f_1$,

(vii) for all $2 \leq i \leq n_2$: $\overline{\delta}'_i(q'^v_i, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = v'^1_i$, $\overline{\delta}'_i(v'^j_i, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = v'^{j+1}_i$, for $1 \leq j \leq n_2 - 2$, and $\overline{\delta}'_i(v'^{n_2-1}_i, \begin{pmatrix}\lambda\\ \lambda\end{pmatrix}) = q'^f_i$.

The system works as follows. We first check in parallel if a word $w$ is in both languages. In order to have $w \in L_1\#$, the first $n_1$ components have to reach final states and read $\begin{pmatrix}\#\\ \#\end{pmatrix}$ at exactly the same time. We use transitions of type (i) until every component $A_i$ reaches $\begin{pmatrix}\#\\ \#\end{pmatrix}$ in a final state, at which moment it enters a special state $q^v_i$. All we have to do now is to verify that all first $n_1$ components entered the states $q^v_i$ at the same time. This is done by using a verification procedure composed of transitions of type (iii) and (iv). Similarly, we use transitions of type (vi) and (vii) to impose the same condition for the last $n_2$ components. Then, each component enters the new final states $q^f_i$ or respectively $q'^f_i$ and waits for all the others to finish parsing the input. Hence, the accepted language is $(L_1\#)\bigcap(L_2\#)$. □

Using a similar technique, we also obtain the following result.

**Theorem 12.** *Let $L_1, L_2 \subseteq V^*$ be two languages accepted by some parallel communicating Watson-Crick automata systems of degrees $n_1$ and $n_2$, respectively, using the same complementarity relation. Then the language $L_1\#L_2\#$ is also accepted by a system of degree $n_1 + n_2$.*

*Proof.* We construct a new system $\mathcal{A}$ of degree $n_1+n_2$ which works as follows. The first $n_1$ components recognize the language $L_1\#(V \cup \{\#\})^*$ by verifying that the first $\begin{pmatrix}\#\\ \#\end{pmatrix}$ is read by all of them at exactly the same moment and then they enter a new final state in which they finish reading the input string. Similarly, the last $n_2$ components recognize the language $V^*\#L_2\#$.

Since a word is accepted by $\mathcal{A}$ if and only if all components reach final states and read all the input, the language accepted by $\mathcal{A}$ is $L_1\#L_2\#$. □

**Theorem 13.** *Let $L \subseteq V^*$ be a language accepted by some parallel communicating Watson-Crick automata system. Then the language $(L\#)^*$ is also accepted by a system of equal degree.*

*Proof.* Let $L = L(\mathcal{A})$ where $\mathcal{A} = (V, \rho, A_1, \ldots, A_n, K)$ is a system of degree $n$ with each $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$. Starting from $\mathcal{A}$ we construct a new system $\mathcal{A}' = (V \cup \{\#\}, \rho, A'_1, \ldots, A'_n, K)$ with $A'_i = (V \cup \{\#\}, \rho, Q'_i, q^0_i, \{q^0_i, q^f_i\}, \delta'_i)$ by adding some new states and transitions as follows. In order to recognize also the empty word, we introduce in each component a new initial and final state $q_i^0$. We also introduce transitions $q_i^0 \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} q_i$, where $q_i$ is the initial state of component $i$ in system $\mathcal{A}$. Then, the system $\mathcal{A}'$ simulates $\mathcal{A}$ on each component until we reach $\binom{\#}{\#}$.

Next, we use the verification procedure described in Theorem 11 to check that all components read $\binom{\#}{\#}$ at exactly the same moment in which case they each enter a new final state $q_i^f$. Then, by introducing transitions of the form $q_i^f \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} q_i$ in each component, we assure that the system can loop, also preserving the synchronization of components.

Thus, the system recognizes the language $\{\lambda\} \cup (L\#)^+$. □

Next, we give a representation result for recursively enumerable languages using languages accepted by parallel communicating Watson-Crick automata systems. We start by recalling a known characterization of recursively enumerable languages, see [14].

**Lemma 14.** *For each recursively enumerable language $L \subseteq V^*$, there exist two $\lambda$-free morphisms $h_1, h_2$, a regular language $R$, and a projection $pr_V$ such that $L = pr_V(h_1(EQ(h_1, h_2)) \cap R)$.*

The next lemma was also proved in [14].

**Lemma 15.** *If $h_1, h_2 : V^* \to V^*$ are two morphisms, then $h_1(EQ(h_1, h_2))$ can be recognized by a Watson-Crick finite automaton.*

Using the previous two results as well as the closure of parallel communicating Watson-Crick automata systems under intersection, we can prove the following characterization.

**Theorem 16.** *For each recursively enumerable language $L \subseteq V^*$, there exists a projection $pr_V$ such that $L = pr_V(L(\mathcal{A}))$, where $\mathcal{A}$ is a parallel communicating Watson-Crick automata system of degree 2.*

*Proof.* Let $L$ be a recursively enumerable language. From Lemma 14 and Lemma 15 we have that there exists a projection $pr_V$ such that $L = pr_V(L' \cap R)$, where $L'$ is recognized by a Watson-Crick finite automaton and $R$ is a regular language. Moreover, for any given complementarity relation $\rho$ we can easily construct a Watson-Crick automaton $\mathcal{M}$ such that $L(\mathcal{M}) = R$.

From Theorem 11 we obtain that there exists a parallel communicating Watson-Crick automata system $\mathcal{A}$ of degree 2 such that $L(\mathcal{A}) = (L'\#) \cap (R\#)$. Since $\# \notin V$, we can extend the projection $pr_V$ by setting $pr_V(\#) = \lambda$ and obtain $L = pr_V(L(\mathcal{A}))$. □

## 5 Conclusions

In this paper we introduced and investigated parallel communicating Watson-Crick automata systems. We prove that their accepting power is increased compared to Watson-Crick finite automata. However, every language accepted by a Watson-Crick finite automata system is context-sensitive. Moreover, one-letter languages accepted by such systems but with an injective complementarity relation prove to be regular. We also investigate some closure properties for these systems and give a representation of recursively enumerable languages.

Many questions and problems have remained open. For example it would be interesting to investigate other closure properties, e.g. under union or complementation. Also, it remains open what is the accepting power of systems using non-injective complementarity relations, when we restrict only to one-letter alphabets.

## 6 Acknowledgement

The authors are grateful to Prof. Arto Salomaa for valuable discussions. Also, the authors are thankful to the anonymous reviewers for their useful comments.

## References

[1] S. Balan, *Distributed Processing in Automata*, Master Thesis, Department of Computer Science and Engineering, Indian Institute of Technology, (2000).

[2] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, G. Vaszil, *Parallel Communicating Pushdown Automata Systems*, Int. J. Found. Comput. Sci. 11(4), 633-650, (2000).

[3] E. Csuhaj-Varjú, V. Mitrana, G. Vaszil, *Distributed Pushdown Automata Systems: Computational Power*, Proc. DLT 2003, LNCS, 2710, 218-229, (2003).

[4] E. Czeizler, E. Czeizler, *On the Power of Parallel Communicating Watson-Crick Automata Systems*, Theoretical Computer Science, 358: 1, 142-147, (2006). Also as TUCS Techreport 722, (2005).

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

[6] M. A. Harrison, *Introduction to formal language theory*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1978.

[7] J. E. Hopcroft, J. D. Ullman, *Introduction to automata theory, languages, and computation.*, Addison-Wesley, (1979).

[8] O. H. Ibarra, *A note on semilinear sets and bounded-reversal multihead push-down automata*, Information Processing Letters, 3, 25-28, (1974).

[9] C. Martín-Vide, V. Mitrana, *Parallel communicating automata systems. A Survey*, Korean Journ. of Comp. and Appl. Math 7: 2, 237-257, (2000).

[10] C. Martín-Vide, A. Mateescu, V. Mitrana, *Parallel finite automata systems communicating by states*, Intern. Journ. of Found. of Comp. Sci. 13: 5, 733-749, (2002).

[11] 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).

[12] V. Mihalache, A. Salomaa, *Lindenmayer and DNA: Watson-Crick DOL systems*, Current Trends in Theoretical Computer Science, World Sci., 740-751, (2001).

[13] A Păun, M. Păun, *State and transition complexity of Watson-Crick finite automata*, Proc. Fundamentals of Computation Theory, FCT'99, LNCS **1684**, Springer-Verlag, 409-420, (1999).

[14] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, (1998).

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

[16] G. Rozenberg, A. Salomaa (eds.) *The Handbook of Formal Languages*, Springer-Verlag, (1997).

[17] J. M. Sempere, *A Representation Theorem for Languages accepted by Watson-Crick Finite Automata*, Bulletin of the EATCS 83, 187-191, (2004).

[18] A. C. Yao, R. L. Rivest, $k+1$ *heads are better than* $k$, Journal of the Associaton for Computing Machinery, 25:2, 337-340, (1978).