ARTO SALOMAA

Turku Centre for Computer Science and,
Department of Mathematics, University of Turku
FIN-20014 Turku, Finland
asalomaa@utu.fi

A Short Survey on Watson-Crick Automata

Elena Czeizler Eugen Czeizler
Department of Mathematics, University of Turku and
Turku Centre for Computer Science
Turku 20520, Finland
elpetr@utu.fi,euczei@utu.fi

Abstract

This paper surveys some known results in Watson-Crick automata theory. In particular, we concentrate on the computational power, complexity measures, decidability problems, and systems of Watson-Crick automata working together on the same input. This selection of topics is not exhaustive, reflecting the research interests of the authors. A series of open problems and questions is also included.

1 Introduction

The remarkable progress witnessed by molecular biology and biotechnology in the last couple of decades, particularly in sequencing, synthesizing, and manipulating DNA molecules, raised the possibility of using DNA as a support for computation. The computer science community was quick to react to this challenge and many computational models were build attempting to exploit the advantages of nano-level biomolecular computing. One of them is the Watson-Crick automata, the focus of this survey.

Watson-Crick automata, introduced in [8], represent one instance of mathematical model abstracting biological properties for computational purposes. They are finite automata with two reading heads, working on double stranded sequences. 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 DNA nucleotides. The two strands of the input are separately scanned from left to right by read only heads controlled by a common state. Over the time, several variants and restrictions of Watson-Crick automata were introduced and investigated. For example, initial stateless Watson-Crick finite automata, reverse Watson-Crick automata, Watson-Crick two-way finite automata, Watson-Crick automata with a Watson-Crick memory, and Watson-Crick transducers were considered in [14]. Also, simple and 1-limited Watson-Crick automata with bounded number of leaps between the two strands were investigated in [12], while Watson-Crick ω-automata were discussed in [15].

In this survey we concentrate only on the basic form of Watson-Crick automata and its subclasses: simple, 1-limited, stateless, and all-final. We present an overview of the results, mainly concentrating on the computational power, complexity measures, decidability problems, and systems of Watson-Crick automata working together on the same input.

Section 3 is devoted to the computational power of these automata and their position in the Chomsky hierarchy. We start by presenting a result from [10] showing that the structure of the complementarity relation plays no actual role for Watson-Crick automata, i.e., any language accepted by a Watson-Crick automaton is also accepted by one with a one-to-one complementarity relation. We also present here the representation theorem for languages accepted by Watson-Crick automata from [18], as well as a series of characterizations of recursively enumerable languages starting from languages accepted by Watson-Crick automata considered in [14].

In Section 4 we present some complexity measures and several related decidability problems investigated in [13]. We also present how these complexities give a measure of comparison between the efficiency of finite automata and Watson-Crick automata when recognizing regular languages.

When considering DNA molecules as a possible support for computations, we may use two key features: the Watson-Crick complementarity and the massive parallelism. However, Watson-Crick finite automata exploit only the first feature. One of the first questions of this field was how to create a mathematical model efficiently using both features of DNA computations. Parallel communicating Watson-Crick automata systems, introduced and investigated in [6] and [7], represent a possible solution for this problem. In Section 5 we present several results concerning the computation power of these systems, some closure properties of the family of accepted languages, as well as some interesting examples. Contrary to the result presented in Section 3, in this case the structure of the complementarity relation plays an active role. While only regular one-letter languages are accepted when using injective complementarity relations, with non-injective ones we can accept more; an original example illustrating this feature is given in this section. Distributed Watson-Crick automata systems, introduced and investigated in [9], are also shortly presented here.

In the following section we give some basic definitions and notations further used in the paper.

2 Preliminaries

We start by assuming the reader is familiar with the fundamental concepts from formal languages and automata theory; for more details we refer to [17].

Let V be a finite alphabet. We denote by V^* the set of all finite words over V, by λ 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. For a finite set Q, let us denote by $card(Q)$ and 2^Q the cardinality and the power set of Q, respectively.

Given two finite 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 $h(w_1w_2) = h(w_1)h(w_2)$, for $w_1, w_2 \in V^*$. If $h(a) \neq \lambda$ for each $a \in V$, then we say that h is a *λ-free morphism*. A *projection* associated to the alphabet V is 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. A morphism $h : V^* \to U^*$ is called a *coding* if $h(a) \in U$ for each $a \in V$ and a *weak coding* if $h(a) \in U \cup \{\lambda\}$ for each $a \in V$. The *equality set* of two morphisms $h_1, h_2 : V^* \to U^*$ is defined as:

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

Let us define now a symmetric relation $\rho \subseteq V \times V$, called *the Watson-Crick complementarity relation on V*, inspired by the Watson-Crick complementarity of nucleotides in the double stranded DNA molecule. We say that ρ is injective if for any $a \in V$ there exists 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 $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ instead of the element $(w_1, w_2) \in V^* \times V^*$.

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 ρ.

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, and $\delta : Q \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to 2^Q$ 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 $s\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$ 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:

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

Let us illustrate now the previous definition of a Watson-Crick automaton by considering the following simple example.

Example 1. *Let $\mathcal{M} = (\{a, b, c\}, \rho, \{q_0, q_1, q_2, q_f\}, q_0, \{q_f\}, \delta)$ be a Watson-Crick automaton where ρ is the identity relation and the transition function is given in Table 1. It is easy to see that the automaton $\mathcal{M}$ accepts the non context-free language $\{a^n b^n c^n \mid n \geq 1\}$.*

$$\begin{array}{ll}
\delta(q_0, \binom{a}{\lambda}) = q_0 & \delta(q_0, \binom{b}{a}) = q_1 \\
\delta(q_1, \binom{b}{a}) = q_1 & \delta(q_1, \binom{c}{b}) = q_2 \\
\delta(q_2, \binom{c}{b}) = q_2 & \delta(q_2, \binom{\lambda}{c}) = q_f \\
\delta(q_f, \binom{\lambda}{c}) = q_f &
\end{array}$$

Table 1: The transition function of Example 1

Another type of language accepted by a Watson-Crick automaton can be defined by considering the set of rewriting rules instead of the recognized sequences, see [8], [12], and [14]. Each rewriting rule is labelled and the accepted language is defined as the set of sequences of labels of rules used when accepting a word $w \in L(\mathcal{M})$. Although interesting results are obtained for this type of languages, here we are not considering them at all.

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 $\mathcal{M} = (V, \rho, Q, q_0, F, \delta)$ 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 at each step the automaton reads either from the upper or from the lower strand, i.e., for any rewriting rule $s\binom{w_1}{w_2} \to \binom{w_1}{w_2}s'$ we have either $w_1 = \lambda$ or $w_2 = \lambda$;
- *1-limited* if it is simple and at every step the automaton reads only one letter, i.e., for any rewriting rule $s\binom{w_1}{w_2} \to \binom{w_1}{w_2}s'$ we have $|w_1w_2| = 1$.

3 The Computational Power of Watson-Crick Finite Automata

In this section we discuss the accepting power of Watson-Crick automata. The essential difference between these and finite automata is in the data structure they process: while finite automata work on finite words, Watson-Crick automata work on double stranded words where characters on corresponding positions from the two strands are connected by the complementarity relation. Thus, it is only natural to ask in what way the structure of the complementarity relation influences the accepting power. This question was recently answered in [10].

Theorem 1. *Let $\mathcal{M} = (V, \rho, Q, q_0, F, \delta)$ be a Watson-Crick automaton with an arbitrary complementarity relation $\rho \subseteq V \times V$. Then, we can construct a Watson-Crick automaton $\mathcal{M}' = (V, \Delta_V, Q, q_0, F, \delta')$ with the identity complementarity relation $\Delta_V = \{(a, a) \mid a \in V\}$ such that $L(\mathcal{M}) = L(\mathcal{M}')$.*

The automaton $\mathcal{M}'$ is constructed such that it accepts an input $\begin{bmatrix} u \\ u \end{bmatrix} \in WK_{\Delta_V}(V)$ if there exists a word $v \in V^*$ for which $\begin{bmatrix} u \\ v \end{bmatrix} \in WK_\rho(V)$ is accepted by $\mathcal{M}$. More precisely, any rewriting rule $s\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$ from $\mathcal{M}$ is replaced in $\mathcal{M}'$ by $s\begin{pmatrix} w_1 \\ w_2' \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2' \end{pmatrix} s'$ where $\begin{bmatrix} w_2' \\ w_2 \end{bmatrix} \in WK_\rho(V)$. Actually, this result holds for any one-to-one complementarity relation $\rho' \subseteq V \times V$ instead of Δ_V. Moreover, this result is valid for all the subclasses of Watson-Crick automata defined in Section 2, i.e., stateless, all final, simple, and 1-limited, as well as for Watson-Crick ω-automata introduced in [15].

The relations between the families of languages accepted by the subclasses of Watson-Crick automata and their position in the Chomsky hierarchy were comprehensively investigated in [14]. A synthesis of these relations is given in the following result.

Theorem 2.

i) Simple and 1-limited Watson-Crick automata accept the same family of languages, i.e., the family of languages accepted by Watson-Crick automata with arbitrary rewriting rules.

ii) There is a strict inclusion between the families of languages accepted by stateless Watson-Crick automata, all final Watson-Crick automata, and Watson-Crick automata with arbitrary rewriting rules, respectively.

iii) The family of languages accepted by Watson-Crick automata with arbitrary rewriting rules is strictly included in the family of context-sensitive languages.

iv) Watson-Crick automata with arbitrary rewriting rules are more powerful than finite automata when considering arbitrary alphabets, but they are equivalent on one-letter languages.

v) Watson-Crick automata with arbitrary rewriting rules are equivalent with two-head finite automata and parallel communicating finite automata systems with two components, see [2].

Recently, a representation theorem for languages accepted by Watson-Crick automata with arbitrary rules was given in [18].

Theorem 3. *Let L be a language accepted by a Watson-Crick automaton with arbitrary rules. Then, $L = g(h^{-1}(L_1 \cap L_2) \cap R))$ where L_1 is a linear language, L_2 is an even linear language, R is a regular language, and g and h are two morphisms.*

One of the most investigated subjects in formal language theory concentrates on finding characterizations of recursively enumerable languages starting from languages in a proper subfamily and using certain operations. In [14], this was done starting from the family of languages accepted by Watson-Crick automata, using only some projections or weak codings.

Theorem 4. *Let L be a recursively enumerable language. Then, we have the following representations:*

i) L is the weak coding of a language accepted by a Watson-Crick automaton from any of the subclasses all-final, simple, 1-limited, or with arbitrary rules.

ii) $L = h(L_1 \cap R)$ where L_1 is a language accepted by a stateless Watson-Crick automaton, R is a regular language, and h is a projection.

iii) $L = h(L_1)$ where L_1 is a language accepted by a Watson-Crick automaton with arbitrary rules and h is a projection.

iv) $L = h(L_1 \cap R)$ where L_1 is a language accepted by an all-final, simple Watson-Crick automaton, R is a regular language, and h is a projection.

Since the family of recursively enumerable languages is closed under the applied operations, the previous theorem gives in fact a series of characterizations of recursively enumerable languages.

4 Complexity Measures and Decidability Problems

Defining and investigating complexity measures for Watson-Crick automata was proposed in [14] as another important class of problems for this field. In [13] three such measures are considered: the classical state and transition complexity measures are extended from automata theory while the maximal distance between the two reading heads is a specific dynamical one.

For a Watson-Crick automaton $\mathcal{M} = (V, \rho, Q, q_0, F, \delta)$, we define *State*$(\mathcal{M})$ and *Trans*$(\mathcal{M})$ as the number of states and rewriting rules of $\mathcal{M}$, respectively. These measures can be naturally extended to languages by defining *State*(L) and *Trans*(L), respectively, as the minimal number of states and transitions, respectively, needed by a Watson-Crick automaton in order to accept the language L. Although 1-limited Watson-Crick automata are equivalent with Watson-Crick automata with arbitrary rewriting rules, *State* and *Trans* complexities give different results for these subclasses; hence, we use $State_1$, $Trans_1$ and $State_a$, $Trans_a$, respectively depending on the subclass of Watson-Crick automata considered.

Let $\Delta : q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s_f$ with $s_f \in F$ be a *successful computation* in $\mathcal{M}$. We define

$$Dist(\Delta) = max\{||w_1'| - |w_2'|| \mid \Delta : q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{pmatrix} w_1' \\ w_2' \end{pmatrix} s \begin{pmatrix} w_1'' \\ w_2'' \end{pmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s_f\},$$

i.e., the maximal distance between the two reading heads during the computation Δ. For any word $w \in L(\mathcal{M})$ we define

$$Dist(w, \mathcal{M}) = min\{Dist(\Delta) \mid \Delta : q_0 \begin{bmatrix} w \\ w' \end{bmatrix} \Rightarrow^* \begin{bmatrix} w \\ w' \end{bmatrix} s_f,\ s_f \in F\}.$$

Let us define now the third complexity measure for Watson-Crick automata,

$$Dist(\mathcal{M}) = sup\{Dist(w, \mathcal{M}) \mid w \in L(\mathcal{M})\},$$

which can be naturally extended to languages as

$$Dist(L) = inf\{Dist(\mathcal{M}) \mid L = L(\mathcal{M})\}.$$

The next result from [13] proves that if the distance between the two reading heads is always bounded, then the behavior of the Watson-Crick automaton can be described using only the states and a "window" of bounded length. Hence the two reading heads can be eliminated and the Watson-Crick automaton can be simulated by a finite automaton.

Theorem 5. *$REG = \{L \mid \exists c > 0$ with $Dist(L) < c\}$, where REG is the set of regular languages.*

At the beginning of a computation both reading heads are placed on the first symbols of the two input strands. It is natural to ask whether or not the two heads find themselves again on the same position during a computation. This question was investigated in [12], relating it to the *Post Correspondence Problem*.

Theorem 6. *Given a simple Watson-Crick automaton $\mathcal{M}$, it is undecidable wether or not there are computations in $\mathcal{M}$ where the two heads of $\mathcal{M}$ are placed on the same position at least twice; the same is true if we look for having the two heads on the same position infinitely many times.*

The following result, from [13], proves that state and transition complexity measures are non-trivial for 1-limited Watson-Crick automata, i.e., there exist languages of arbitrary complexity.

Theorem 7. *For any $n \geq 0$ there exist languages L_n and L'_n accepted by 1-limited Watson-Crick automata such that $State_1(L_n) \geq n$ and $Trans_1(L'_n) \geq n$.*

This problem remains open in the case of Watson-Crick automata with arbitrary rules. Moreover, the connectedness problem remains open in both cases: it is still not known whether for each natural number n grater than a given threshold there is a language with complexity n.

Several decision and computational problems related to complexity measures are investigated in [13]. Although in the case of finite automata some problems are easily decidable, they prove to be undecidable in the case of Watson-Crick automata. The following result gives a synthesis of the problems discussed in [13].

Theorem 8.

i) *For any given positive integer n, it is not decidable whether $Comp_X(L(\mathcal{M}))= n$, where $Comp \in \{State, Trans\}$, $X \in \{1, a\}$, and $\mathcal{M}$ is either an 1-limited Watson-Crick automaton or a Watson-Crick automaton with arbitrary rules.*

ii) *None of the measures $State_1(L), State_a(L), Trans_1(L)$, and $Trans_a(L)$ can be algorithmically computed, where L is a language recognizable by an appropriate Watson-Crick automaton.*

iii) *For any Watson-Crick automaton $\mathcal{M}$ there is no algorithm able to construct $\mathcal{M}_0$ such that $L(\mathcal{M}_0) = L(\mathcal{M})$ and $Comp(\mathcal{M}_0) = Comp_X(L(\mathcal{M}))$, where $Comp \in \{State, Trans\}$ and $X \in \{1, a\}$, i.e., there is no algorithm computing the minimal Watson-Crick automaton.*

iv) *$Dist(\mathcal{M})$ is not algorithmically computable.*

The complexities *State* and *Trans* also give a measure of comparison between the efficiency of finite automata and Watson-Crick automata when recognizing regular languages. In [13] it is proved that Watson-Crick automata can recognize regular languages significantly more efficient than finite automata. For instance, there exists a family of languages L_n, with $n \geq 1$, such that $State(L_n) \geq n$ when

L_n is recognized by a finite automaton, while $State(L_n) \leq k$ for some constant k when L_n is recognized by a Watson-Crick automaton with arbitrary rewriting rules. Similar results are also given using the complexity *Trans*.

5 Watson-Crick Automata Systems

An automata system consists of a set of automata of the same type, working together on the same input according to a predefined protocol. Depending on the protocol used there are two classes of such systems: *sequential* and *parallel*. Both classes were investigated for different types of automata, see for example [4], [5], [9], and [11].

The sequential class is represented by *cooperating distributed* systems. In this case, all components work on the same input tape and at each moment only one component is active. There are several ways in which the control is transferred from one component to another: depending on the number of steps performed, i.e., $= k$-mode, $\leq k$-mode, and $\geq k$-mode, as long as a component can continue the computation, i.e., t-mode, or arbitrary, i.e., $*$-mode; for more details see [3]. 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 the stateless subclass.

On the other hand, *parallel communicating Watson-Crick automata systems*, PCWKS for short, introduced in [6], proved to be more powerful. A PCWKS consists of a set of Watson-Crick automata working independently on their own input tape and communicating states on request. Special *query states* are provided, each pointing to exactly one component of the system. When the i-th component enters the query state K_j, the current state of the j-th component is transmitted to component i and the computation continues. There are two important classifications of parallel communicating systems concerning the *communication graph* and the *returning feature*. A system is called *centralized* if only one component, the *master*, may introduce query states, and *non-centralized* otherwise. A 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. While the non-centralized and non-returning PCWKS are investigated in [6] and [7], all the other cases are still open.

Formally, a PCWKS of degree n is a $(n + 3)$-tuple

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

where

- V is the input alphabet;
- ρ is the complementarity relation;
- $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$, $1 \leq i \leq n$, 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 is a PCWKS of degree 1.

A configuration of a PCWKS 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 $1 \leq i \leq n$. 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 $r_i \in \delta_i(s_i, \binom{x_i}{y_i})$, $1 \leq i \leq n$;
- for all $1 \leq i \leq n$ 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 $1 \leq l \leq n$ we have $r_l = s_l$. In this case $\binom{u'_i}{v'_i} = \binom{u_i}{v_i}$, for all $1 \leq i \leq n$.

If we denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$, then the language recognized by a PCWKS 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 $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$. Moreover, if one of the components stops before the others, the system halts and rejects the input.

Let us continue by considering a simple example from [6] illustrating the communication between components.

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

Table 2: The transition functions of Example 2

Example 2. *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 even length. Then, by using communication between components we check letter by letter that the input is indeed of the form $\begin{bmatrix} ww\# \\ ww\# \end{bmatrix}$.

Although for Watson-Crick automata it is enough to consider only one-to-one complementarity relations, for PCWKS the structure of this relation is very important. In [6] it is proved that if the complementarity relation is injective, then PCWKS accept only regular one-letter languages. However, using non-injective complementarity relations PCWKS can accept also non-regular one-letter languages such as $L = \{a^{n^2} \mid n \geq 2\}$, see [7].

Using a similar technique, let us construct now a PCWKS recognizing the non-regular language $L = \{a^{2^n} \mid n \geq 1\}$.

Example 3. *Let $\mathcal{A} = (\{a, b, c\}, \rho, A_1, A_2, K)$ be a PCWKS of degree 2, where:*

- $\rho = \{(a, b), (a, c)\}$,
- $A_1 = (\{a, b, c\}, \rho, \{q_1, r_b, r_c, r_{bc}, r_{cb}\}, q_1, \{r_b, r_c, r_{bc}, r_{cb}\}, \delta_1)$,
- $A_2 = (\{a, b, c\}, \rho, \{q_2, s_1, r_b, r_c, r_{bc}, r_{cb}, f_2, K_1\}, q_2, \{f_2\}, \delta_2)$.

$Component A_1$	$Component A_2$
$q_1 \binom{a}{b} \to r_b$	$q_2 \binom{aa}{bc} \to s_1$
$r_b \binom{a}{c} \to r_{bc}$	$s_1 \binom{\lambda}{\lambda} \to K_1$
$r_{bc} \binom{a}{c} \to r_c$	$r_{bc} \binom{\lambda}{\lambda} \to f_2$
$r_{bc} \binom{a}{b} \to r_{cb}$	$r_{cb} \binom{\lambda}{\lambda} \to f_2$
$r_b \binom{a}{b} \to r_b$	$f_2 \binom{\lambda}{\lambda} \to f_2$
$r_c \binom{a}{c} \to r_c$	$r_{bc} \binom{aa}{bb} \to K_1$
$r_c \binom{a}{b} \to r_{cb}$	$r_c \binom{aa}{bb} \to K_1$
$r_{cb} \binom{a}{b} \to r_b$	$r_b \binom{aa}{cc} \to K_1$
	$r_{cb} \binom{aa}{cc} \to K_1$

Table 3: The rewriting rules of Example 3

The rewriting rules of the two components are given in Table 3.

The starting point of the previous construction is the identity

$$2^n = 1 + 2^0 + 2^1 + 2^2 + \ldots + 2^{n-1}.$$

Thus, our PCWKS accepts a word a^{2^n} if and only if its complement is of the form bcbbccccb..., i.e., consisting of consecutive blocks of b's and c's such that, starting from the third one, the length of a block is double the length of the previous one. During a computation the second component of the system is always one block ahead of the first component. By permanent communication between the two components the system verifies that any two consecutive blocks, except the first two, fulfill the length condition: each time the first component reads a character from a block the second component reads two characters from the following one. Moreover, the second component can enter its final state only at the end of a block. Since in the final state the second component does not continue to read the input, a word is accepted if and only if it is of the form a^{2^n}.

The following theorem synthesizes the results from [6] and [7] illustrating the accepting power of PCWKS.

Theorem 9.

i) Parallel communicating Watson-Crick automata systems are more powerful than Watson-Crick finite automata. E.g. the language $\{a^{2^n} \mid n \geq 1\}$ from Example 3 cannot be accepted by a Watson-Crick automaton.

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

iii) Every one-letter language accepted by a parallel communicating Watson-Crick automata system with injective complementarity relation is regular. However, using non-injective complementarity relations they accept also non-regular one-letter languages.

iv) 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.

We end this section by discussing the closure of the family of languages recognized by PCWKS under intersection, union, and Kleene $*$ given in [6]. In all these cases a special delimiter # is needed in order to preserve the overall synchronization between the components of the system. Thus, we extend the alphabet V to $V \cup \{\#\}$ and the complementarity relation to $\rho \cup \{(\#, \#)\}$.

Theorem 10. *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 following statements hold.*

i) The language $(L_1\#) \bigcap (L_2\#)$ is accepted by a system of degree $n_1 + n_2$.

ii) The language $L_1\#L_2\#$ is accepted by a system of degree $n_1 + n_2$.

iii) The language $(L_1\#)^*$ is accepted by a system of degree n_1.

6 Open Problems

Recently introduced, Watson-Crick automata are thoroughly investigated in many papers. Different aspects of these systems are considered, creating a vast and interesting subject. Moreover, many of the proofs distinguish themselves by the beauty of their constructions.

In this survey we put together only some of the known results concerning the computational power and complexity measures of Watson-Crick automata. We also discuss different problems concerning distributed and parallel communicating Watson-Crick automata systems. Moreover, several open problems and questions are mentioned throughout the paper.

In Section 4 we discuss three complexity measures for Watson-Crick automata: *State*, *Trans*, and *Dist*. Although it is known that state and transition complexity measures are non-trivial for 1-limited Watson-Crick automata, this problem remains open for Watson-Crick automata with arbitrary rules. That is, it is not known whether for any $n \geq 0$ there exist languages L_n and L'_n such that $State_a(L_n) \geq n$ and $Trans_a(L'_n) \geq n$. Also, the connectedness problem remains open for both 1-limited and Watson-Crick automata with arbitrary rules, i.e., it is not known whether for each natural number n greater than a given threshold there is a language with complexity n.

One of the early open problems proposed in [14] concerns the efficient use of the massive parallelism feature of DNA molecules. Although parallel communicating Watson-Crick automata systems were introduced as a possible answer to this question, it is still worth looking for other, maybe better approaches. Also, several interesting problems concerning PCWKS are still open. Depending on the communication graph and the returning feature there are two types of classifications for PCWKS: centralized/non-centralized and returning/non-returning. Until now, only non-centralized and non-returning PCWKS were considered while all the other cases remain open. Finally, it would be also interesting to investigate other closure properties for the family of languages accepted by PCWKS, with or without the special delimiter # used in Theorem 10.

References

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

[2] L. C , *Watson-Crick Automata and PCFAS with Two Components: A Computational Power Analogy*, Conf. Computing Frontiers, 150-161, (2004).

[3] E. C -V ´, J. D , J. K , G . P˘ , *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London (1994).

[4] E. C -V ´, C. M ´ -V , V. M , G. V , *Parallel Communicating Pushdown Automata Systems*, Int. J. Found. Comput. Sci. 11(4), 633-650, (2000).

[5] E. C -V ´, V. M , G. V , *Distributed Pushdown Automata Systems: Computational Power*, Proc. DLT 2003, LNCS, 2710, 218-229, (2003).

[6] E. C , E. C , *Parallel Communicating Watson-Crick Automata Systems*, Proc. 11th International Conference AFL, (2005).

[7] E. C , E. C , *On the Power of Parallel Communicating Watson-Crick Automata Systems*, submitted. Also as TUCS Techreport 722, (2005).

[8] R. F , G . P˘ , G. R , A. S , *Watson-Crick Finite Automata*, Proc 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 297-328, (1997).

[9] K. K , M. S B , P. H , *Distributed Processing in Automata*, Int. J. Found. Comput. Sci. 10 (4), 443–464, (1999).

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

[11] C. M ´ -V , V. M , *Parallel Communicating Automata Systems. A Survey*, Korean Journ. of Comp. and Appl. Math 7: 2, 237-257, (2000).

[12] C. M ´ -V , G . P˘ , *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).

[13] A. P˘ , M. P˘ , *State and Transition Complexity of Watson-Crick Finite Automata*, Fundamentals of Computation Theory, 409-420, (1999).

[14] G . P˘ , G. R , A. S , *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, (1998).

[15] E. P , *Watson-Crick ω-Automata*, J. Autom. Lang. Comb. **8(1)**, 59-70, (2003).

[16] G. R , A. S (.) *The Handbook of Formal Languages*, Springer-Verlag, (1997).

[17] A. S , *Formal Languages*, Academic Press, New York, (1973). Revised edition in the series "Computer Science Classics", Academic Press, (1987).

[18] J. M. S , *A Representation Theorem for Languages Accepted by Watson-Crick Finite Automata*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS **83**, 187-191, (2004).