Elena Czeizler | Eugen Czeizler

On the Power of Parallel Communicating Watson-Crick Automata Systems

Turku Centre *for* Computer Science

TUCS Technical Report
No 722, November 2005

On the Power of Parallel Communicating Watson-Crick Automata Systems

Elena Czeizler
Department of Mathematics, University of Turku and
Turku Centre for Computer Science
Turku 20520, Finland
elenac@it.utu.fi

Eugen Czeizler
Department of Mathematics, University of Turku and
Turku Centre for Computer Science
Turku 20520, Finland
eugenc@it.utu.fi

TUCS Technical Report
No 722, November 2005

Abstract

Parallel communicating Watson-Crick automata systems were introduced in [1] as possible models of DNA computations. This combination of Watson-Crick automata and parallel communicating systems comes as a natural extension due to the new developments in DNA manipulation techniques. It is already known, see [4], that for Watson-Crick finite automata, the complementarity relation plays no active role. However, this is not the case when considering parallel communicating Watson-Crick automata systems. In this paper we prove that non-injective complementarity relations increase the accepting power of these systems. We also prove that although Watson-Crick automata are equivalent to two-head finite automata, this equivalence is not preserved when comparing parallel communicating Watson-Crick automata systems and multi-head finite automata.

Keywords: Watson-Crick, parallel communicating automata systems, complementarity relation

TUCS Laboratory
Discrete Mathematics for Information Technology Laboratory

1 Introduction

Watson-Crick finite automata are a counterpart of finite automata working on double stranded sequences. They were introduced in [3] and, as suggested by the name, they are intended as a formalization of DNA manipulation. 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 similarly with the Watson-Crick complementarity of DNA nucleotides. Several variants of these automata were investigated in [6], [7], [8], and [10], see also [9] for a comprehensive presentation.

When considering DNA molecules as a possible support for computations, we may exploit two key features: the Watson-Crick complementarity and the *massive parallelism*. While Watson-Crick automata make use only of the first one, *parallel communicating Watson-Crick automata systems*, introduced in [1], come as a possible answer to the problem formulated in [9] of combining the two features into a model of DNA computations.

A parallel communicating Watson-Crick automata system, PCWKS for short, consists of a set of Watson-Crick finite automata working independently on their own input tape and communicating states on request. 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. As in the case of other parallel communicating automata systems, see for example [5], the communication between components is done using special *query states*, each of them pointing to exactly one component of the system. When component i reaches a query state K_j, the current state of the component j will be communicated to i and the computation continues. We refer to [2] for different paradigms of parallelism and communication in *grammar systems*.

Another question from [9] is about the role of the complementarity relation regarding the expressive power of Watson-Crick automata. A first answer is given in [4], where it is proved that 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. This result is also extended for the Watson-Crick ω-automata introduced in [10].

In this paper we prove that for PCWKS, the complementarity relation plays an active role. In [1] it was shown that systems with injective complementarity relation accept only regular one-letter languages. Here we prove that if the relation is not injective, then we can accept also some non-regular one-letter languages.

Since in [9] it is proved that Watson-Crick automata are equivalent with *two-head finite automata*, a natural question is whether this equivalence is preserved when considering PCWKS and multi-head finite automata. It is already known, see [5], that n-head finite automata are equivalent with *parallel communicating finite automata systems* (communicating by states) with n components and they recognize only *semilinear languages*. The main

result of this paper proves that PCWKS are more powerful: they recognize all languages accepted by parallel communicating finite automata systems and also some non semilinear languages such as $\{a^{n^2} \mid n \geq 2\}$.

2 Definitions

We assume that the reader is familiar with the fundamental concepts from formal languages and automata theory; for more details we refer to [11].

For a finite alphabet V we denote by V^* the set of all finite words over V and by λ the empty word.

Let now $\rho \subseteq V \times V$ be a symmetric relation, 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 any $a \in V$ has the 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 $\binom{V^*}{V^*}$ 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 ρ. 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 words 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, $\delta : Q \times \binom{V^*}{V^*} \to 2^Q$ is a mapping, called the *transition function*, such that $\delta(q, \binom{w_1}{w_2}) \neq \emptyset$ 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

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

For more details on Watson-Crick automata we refer to [9].

A parallel communicating Watson-Crick automata system of degree n, PCWKS(n) for short, 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 $\mathcal{A}$.

These systems were introduced in [1] where their accepting power and some closure properties were investigated. They are composed of several Watson-Crick automata working independently on their tapes and communicating on request by use of query states. Each of these states points to exactly one component of the system such that, when a component A_i reaches a query state K_j, the current state of the component A_j will be communicated to A_i and the computation continues.

A configuration of a PCWKS(n) is a $2n$-tuple

$$(s_1, \begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, s_2, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, \ldots, s_n, \begin{pmatrix} u_n \\ v_n \end{pmatrix})$$

where s_i is the current state of the component A_i and $\begin{pmatrix} u_i \\ v_i \end{pmatrix}$ is the part of the input which has not been read yet by the component A_i, for all $1 \leq i \leq n$. We define a binary relation $\vdash$ on the set of all configurations by setting

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

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

- $K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset$, $\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \begin{pmatrix} u'_i \\ v'_i \end{pmatrix}$, and $r_i \in \delta_i(s_i, \begin{pmatrix} x_i \\ y_i \end{pmatrix})$, for all $1 \leq i \leq n$;
- 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 $\begin{pmatrix} u'_i \\ v'_i \end{pmatrix} = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$, 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:

$$L(\mathcal{A}) = \{w_1 \in V^* \mid (q_1, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, q_2, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \ldots, q_n, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}) \vdash^*$$

$$(s_1, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, s_2, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \ldots, s_n, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}),\ s_i \in F_i,\ 1 \leq i \leq n\}.$$

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 the input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$.

3 Main result

In [1] it was proved that PCWKS with injective complementarity relation accept only regular one-letter languages. In this section we prove that by using non-injective complementarity relation we increase the power of these systems.

Theorem 1 *The language $\{a^{n^2} \mid n \geq 2\}$ can be accepted by a parallel communicating Watson-Crick automata system with three components and a non-injective complementarity relation.*

Proof: The proof is based on the following observation. A word $w \in \{a\}^*$ is of the form a^{n^2} for some $n \geq 2$ if and only if:

(1) we can divide w into blocks of equal length and

(2) the number of such blocks is equal to their length.

Hence, we build a system accepting a word $w = a^{n^2}$ only when its complement is of the form $b^n c^n b^n c^n \ldots$ with exactly $n-1$ alternations between blocks of b's and c's. The system is composed of three components; the first two verify that the complement has alternating blocks of b's and c's of equal length, while the first and the third components verify that the number of such blocks is equal to the length of the first block of b's.

Formally, we construct a PCWKS $\mathcal{A} = (\{a, b, c\}, \rho, A_1, A_2, A_3, K)$, where:

- $\rho = \{(a, b), (a, c)\}$,
- $A_1 = (\{a, b, c\}, \rho, \{q_1, r_1, s_b, s_c, s_{bc}, s_{cb}, f_1^c, f_1^b\}, q_1, \{f_1^c, f_1^b\}, \delta_1)$,
- $A_2 = (\{a, b, c\}, \rho, \{q_2, r_1, s_b, s_c, s_{bc}, s_{cb}, f_1^c, f_1^b, f_2, K_1\}, q_2, \{f_2\}, \delta_2)$,
- $A_3 = (\{a, b, c\}, \rho, \{q_3, r_1, s_b, s_c, s_{bc}, s_{cb}, f_1^c, f_1^b, f_3, K_1\}, q_3, \{f_3\}, \delta_3)$.

The transition functions of the components are given in Table 1.
For an input of the form

$$\begin{bmatrix} aaaaaaaaaaaaa \ldots \\ bb \ldots bcc \ldots cbb \ldots \end{bmatrix}$$

Component A_1	Component A_2	Component A_3
$q_1 \binom{a}{b} \to \binom{a}{b} r_1$	$q_2 \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} K_1$	$q_3 \binom{a}{b} \to \binom{a}{b} K_1$
$r_1 \binom{a}{b} \to \binom{a}{b} r_1$	$r_1 \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} K_1$	$r_1 \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} K_1$
$r_1 \binom{a}{c} \to \binom{a}{c} s_{bc}$	$s_{bc} \binom{a}{b} \to \binom{a}{b} K_1$	$s_{bc} \binom{a}{b} \to \binom{a}{b} K_1$
$s_{bc} \binom{a}{c} \to \binom{a}{c} s_c$	$s_c \binom{a}{b} \to \binom{a}{b} K_1$	$s_c \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} K_1$
$s_c \binom{a}{c} \to \binom{a}{c} s_c$	$s_{cb} \binom{a}{c} \to \binom{a}{c} K_1$	$s_{cb} \binom{a}{b} \to \binom{a}{b} K_1$
$s_c \binom{a}{b} \to \binom{a}{b} s_{cb}$	$s_b \binom{a}{c} \to \binom{a}{c} K_1$	$s_b \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} K_1$
$s_{cb} \binom{a}{b} \to \binom{a}{b} s_b$	$f_1^c \binom{a}{c} \to \binom{a}{c} f_2$	$f_1^b \binom{a}{c} \to \binom{a}{c} f_3$
$s_b \binom{a}{b} \to \binom{a}{b} s_b$	$f_1^b \binom{a}{b} \to \binom{a}{b} f_2$	$f_1^c \binom{a}{c} \to \binom{a}{c} f_3$
$s_b \binom{a}{c} \to \binom{a}{c} s_{bc}$	$f_2 \binom{a}{c} \to \binom{a}{c} f_2$	$f_3 \binom{a}{c} \to \binom{a}{c} f_3$
$s_b \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} f_1^b$	$f_2 \binom{a}{b} \to \binom{a}{b} f_2$	$f_3 \binom{a}{b} \to \binom{a}{b} f_3$
$s_c \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} f_1^c$	$f_2 \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} f_2$	
$f_1^b \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} f_1^b$		
$f_1^c \binom{\lambda}{\lambda} \to \binom{\lambda}{\lambda} f_1^c$		

Table 1: The transition rules of system $\mathcal{A}$

the system evolves as follows. The first component reads the first block of b's transmitting at each step its state to the other two components. Meanwhile, the third component counts the first block by reading $\binom{a}{b}$ and then waits for the signal of the next alternation of letters. The second component also waits for the same signal but without reading any input.

When component A_1 reads the first letter of the next block, it enters a special state s_{bc} signaling both the change between blocks and its type, in this case from b's to c's. From now on, the second component starts reading the input and by communicating with the first component checks that any two consecutive one-letter blocks have equal length. Also, when receives the signal of the alternation, the third component counts this next block by reading the next $\binom{a}{b}$. In general, component A_3 reads one $\binom{a}{b}$ each time it receives from A_1 a signal of alternation, i.e., state s_{bc} or s_{cb}.

The computation evolves similarly, until the first component reads all the input. Then, it enters one of the final states f_1^b or f_1^c depending on the type of the last block. Both A_2 and A_3 acknowledge this signal and react as follows. The second component enters its final state f_2 only if it reads on the input tape exactly the letter indicated by the signal. In this final state it finishes reading the input. The third component also enters its final state only if it reads $\binom{a}{c}$ on the input tape, verifying that the number of blocks equals the length of the first block; then it finishes reading the input.

If there exist two consecutive blocks of b's and c's of different lengths, then the second component enters a deadlock. On the other hand, if the number of blocks of b's and c's is not equal to the size of the first block, then the third component enters a deadlock. On both cases the input is not accepted by the system. □

As an immediate consequence we have the following result.

Corollary 2 *There exist PCWKS with non-injective complementarity relation accepting non-regular one-letter languages.*

Moreover, any language accepted by a PCWKS with injective complementarity relation can be accepted also by a system with non-injective complementarity relation. Thus, we have the following result.

Corollary 3 *Parallel communicating Watson-Crick automata systems with non-injective complementarity relation are more powerful than systems with injective complementarity relation.*

Although Watson-Crick automata are equivalent to two-head finite automata, see [9], this is not true anymore when considering PCWKS and multi-head automata. In order to prove this, let us first recall the following result from [5].

Theorem 4 *A language is accepted by an n-head finite automaton if and only if it is accepted by a parallel communicating finite automata system with n components.*

In the same paper, it is also proved that parallel communicating finite automata systems accept only semilinear languages.

Moreover, it is easy to prove that for any parallel communicating finite automata system with n components we can construct an equivalent Watson-Crick system with n components and the identity complementarity relation. Since the language $\{a^{n^2} \mid n \geq 2\}$ is not semilinear we obtain the following result.

Corollary 5 *Parallel communicating Watson-Crick automata systems are more powerful than multi-head automata and parallel communicating finite automata systems respectively.*

References

[1] E. Czeizler, E. Czeizler, *Parallel Communicating Watson-Crick Automata Systems*, in Z. Ésik, Z. Fülöp eds., Proceedings of Automata and Formal Languages, AFL'05, 83-96, 2005.

[2] J. Dassow, Gh. Păun, G. Rozenberg, *Grammar Systems* in G. Rozenberg, A. Salomaa (eds.), The Handbook of Formal Languages, Springer-Verlag, Vol **2**, 155-213, (1997).

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

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

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

[6] 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).

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

[8] 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).

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

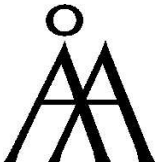
[10] E. Petre, *Watson-Crick ω-Automata*, J. Autom. Lang. Comb. 8(1), 59-70, (2003).

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

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1640-9
ISSN 1239-1891