Fundamenta Informaticae ? (2011) 1–13
IOS Press

Hierarchies of Stateless Multicounter $5' \to 3'$ Watson-Crick Automata Languages

Ömer Eğecioğlu
Department of Computer Science, University of California,
Santa Barbara, CA 93106, USA
omer@cs.ucsb.edu

László Hegedüs, Benedek Nagy
Department of Computer Science, Faculty of Informatics,
University of Debrecen, Debrecen, 4032 Hungary
hegedus.laszlo@inbox.com, nbenedek@inf.unideb.hu

Abstract. We consider stateless counter machines which mix the features of one-head counter machines and a special type of two-head Watson-Crick automata (WK-automata). Our Watson-Crick counter machines are biologically motivated. They have two heads that read the input starting from the two extremes. The reading process is finished when there are no more symbols between the heads, i.e., every letter of the input is processed by either head. Depending on whether the heads are required to advance at each move, we distinguish between realtime and non-realtime machines. If every counter makes at most k alternations between nondecreasing and decreasing modes in every computation, then the machine is k-reversal. It is reversal bounded if it is k-reversal for some k. In this paper we concentrate on the properties of both deterministic and nondeterministic stateless WK-automata with reversal bounded counters.

1. Introduction

A well-investigated branch of DNA computing is the theory of Watson-Crick automata ([3, 18]). These are finite state machines equipped with two read-only heads. They operate on strings modeling DNA molecules, i.e., double stranded sequences of bases. The strands of a DNA molecule have directions as

Address for correspondence: Benedek Nagy, Department of Computer Science, Faculty of Informatics, University of Debrecen, Debrecen, 4032, Hungary.

a result of the underlying chemical bonds, determining the $5'$ and $3'$ ends of a strand. The two strands have opposite biochemical directions. Between the two strands, there is a one-to-one correspondence of the bases given by the so-called Watson-Crick complementarity relation. In this way a strand of the molecule uniquely defines the other, and therefore the DNA molecules can be described by ordinary strings (as, for instance, in [9, 10]). Our machines will operate on strings representing DNA molecules. In biology several enzymes are known to act on a DNA strand in direction from $5'$ to $3'$. Consequently, in the case of $5' \to 3'$ Watson-Crick automata, at the beginning of a computation, the reading heads start from opposite ends of the input and they move in the opposite direction from a computational point of view (but the same direction biochemically). These automata have been used to characterize linear context-free languages in [13]. In this paper we consider only $5' \to 3'$ Watson-Crick automata, and consequently use the terminology *WK-automata* and omit the symbols $5' \to 3'$.

Stateless machines (i.e. machines with only one state) have recently been connected to certain aspects of membrane computing and P systems, a subarea of molecular computing that was introduced by Gheorghe Păun [16, 17]. A membrane in a P system consists of a multiset of objects drawn from a given finite type set $\{a_1, \ldots, a_m\}$. The system has no global state and works on the evolution of objects in a massively parallel way. Thus, the membrane can be modeled as having counters $c_1, \ldots, c_m$ to represent the multiplicities of objects of types $a_1, \ldots, a_m$, respectively. A P system can then be thought of as a counter machine in a nontraditional form: without states, and with parallel counter increments/decrements. It is therefore natural to consider the model of computation which has no states but is equipped with counters. These are the two features that motivate the study of stateless multicounter WK-automata. As the name indicates, a stateless machine (without additional storage) cannot store any information by states. Thus other methods are used, e.g., the automaton is equipped with some number of counters which store zero at the beginning and again zero at the end of a computation. Since there are no final states, acceptance of an input string has to be defined in a different way. It is well known that nondeterministic pushdown automata with states are equivalent to stateless nondeterministic pushdown automata (where acceptance is by "null" stack) although this is not true for the deterministic case [4, 7]. In [6, 19] the computing power of stateless multihead automata with respect to decision problems and head hierarchies were investigated. The machine can be deterministic, nondeterministic, one-way, two-way, etc. In [8], various types of stateless restarting automata and two-pushdown automata were compared to the corresponding machines with states.

In two-way automata a head can move in both directions. Two-head automata is a special case of multihead automata, but in our machine the heads move to opposite directions, which is the main feature of $5' \to 3'$ WK-automata. Even though there is a rich literature on various types of automata, it seems that no models equivalent to multicounter $5' \to 3'$ WK-automata have been defined, i.e., automata where the two heads run in opposite directions and, instead of states, the automata is equipped by counters.

If the machine is not allowed to make transitions without moving a head, then the model is called *realtime*. Otherwise it is *non-realtime*. The models also take into account the behavior of the counters. Consider the numerical sequence of distinct values of a single counter during a computation of a counter machine M. If this sequence has k local maxima, then we say that the counter makes k reversals. For example a counter with distinct contents $0, 1, 2, 3, 2, 3, 4, 3, 2, 1, 0$ makes 2 reversals. If each counter of M makes at most k reversals on any computation (accepting or not), then the machine is called *k-reversal*. M is *reversal bounded* if it is k-reversal for some k.

Deterministic stateless (one head, one-way) m-counter machines were investigated in [1], where hierarchies with respect to the number of counters and number of reversals were studied. Similar hierarchy results and characterizations are reported in [5] for the non-realtime versions. Hierarchies of the accepted language families by WK-automata are presented in [12], including stateless versions without counters. Some results about the deterministic case can also be found in [2]. In this paper we concentrate on nondeterministic stateless realtime WK-automata with counters. Here special variations, namely the strong realtime WK-automata are also introduced. In strong realtime WK-automata both heads move in every transition when they are not close to meet. We give some examples and establish hierarchies of WK-automata with respect to counters and reversals.

2. Stateless multicounter WK-automata

We first give a semi-informal explanation of the basic notions of stateless multicounter WK-automata to help the readability. The input is of the form $¢w\$$ with $w \in \Sigma^*$ and $¢$ and $\$$ are endmarkers that are not in Σ. The machine has two read-only heads H_1, H_2. Head H_1 moves from left to right and H_2 moves from right to left. Originally, H_1 is on $¢$ and H_2 is on \$. The machine is equipped with m counters, that are initially all zero. A move of the machine depends on the symbols under the heads and the signs of the counters (the automata can distinguish two cases: zero or positive). It consists of moving the heads and at the same time incrementing, decrementing, or leaving unchanged each counter. The input w is accepted by M if the counters are again zero when the heads *meet*, as explained below.

The essence of when the heads H_1 and H_2 meet is captured best by making use of a function φ which indicates whether the heads are close or far apart in processing the input. This locality requirement can be justified in part by biological properties that give rise to WK-automata. For the model it suffices to know if there are zero, one, two, or more than two letters between the heads. The letters that are not processed yet are considered to be between the heads. The heads meet when there are zero letters between them. (The heads are close to meet if one or two letters are between them.) We define

$$\varphi(M) = \begin{cases} p & \text{if there are } p \in \{0,1,2\} \text{ letters between the two heads of } M, \\ \infty & \text{if there are more than two letters between the heads of } M. \end{cases}$$

We use the notation $\varphi(M)$ although φ is actually a function of the current positions of the heads of M, i.e., function of the actual configuration.

For a deterministic stateless multicounter WK-automaton M, a transition (a move)

$$((x, y; s_1, s_2, \ldots, s_m), p) \to (d_1, d_2; e_1, e_2, \ldots, e_m) \tag{1}$$

has the following parameters: $x, y \in \Sigma \cup \{¢, \$\}$ are the symbols read by the heads H_1 and H_2, respectively; s_i is the sign of counter C_i: $s_i = 0$ if the i-th counter is zero, $s_i = 1$ if it is positive. $s_1 s_2 \cdots s_m$ is referred to as a *sign vector*; $p \in \{0, 1, 2, \infty\}$ is the parameter described above; $d_1, d_2 \in \{0, 1\}$ indicate the direction of move of the heads with $d_1 + d_2 \le p$. A value 0 signifies that the head stays where it is, $d_1 = 1$ means that H_1 moves one cell to the right, and $d_2 = 1$ means that H_2 moves one cell to the left; $e_i = +, -$, or 0, corresponding to the operations of increment, decrement, or leave unchanged the contents of the i-th counter. Here $e_i = -$ applicable only if $s_i = 1$. A move (1) is possible if and only if $\varphi(M) = p$. It should be noted that $\varphi(M)$ is not part of the system, nor it is a counter, just a technical parameter. M is *nondeterministic* if multiple choices are allowed for the right hand side of (1).

The machine is *realtime* if at least one of d_1 and d_2 is not zero for any move of the machine. Otherwise it is *non-realtime*. Thus in a realtime machine at least one of the heads must move at every step of the computation.

The machine is k-*reversal* if for a specified k, no counter makes more than k alternations between increasing mode and decreasing mode (i.e. k pairs of increase followed by decrease stages) in any computation, accepting or not. The machine is *reversal bounded* if it is k-reversal for some k.

We denote the set of all nondeterministic k-reversal m counter non-realtime WK-automata by WKC_m^k, and the realtime versions by RWKC_m^k. The reversal bounded versions are denoted by

$$\text{WKC}_m^* = \bigcup_{k=0}^{\infty} \text{WKC}_m^k, \;\; \text{RWKC}_m^* = \bigcup_{k=0}^{\infty} \text{RWKC}_m^k;$$

while WKC_m^∞ and RWKC_m^∞ are notations for the unbounded reversal versions. We use a "d" prefix to refer to the deterministic versions of these machines. So dRWKC_m^k denotes all deterministic realtime k-reversal m-counter machines. This notation is also used for the corresponding language classes.

The formal definition of a nondeterministic stateless multicounter WK-automaton is as follows.

Definition 2.1. A nondeterministic stateless multicounter WK-automaton is a quintuple $M = (\Sigma, m, \delta, ¢, \$)$ where Σ is a nonempty alphabet, m is the number of counters, δ is a mapping from $(\Sigma \cup \{¢, \$\})^2 \times \{1, 0\}^m \times \{0, 1, 2, \infty\}$ to $2^{(\{0,1\}^2 \times \{0,+,-\}^m)}$ and $¢, \$ \notin \Sigma$ are two special symbols called endmarkers.

The signs $\lceil$ and $\rfloor$ will be used to indicate the read heads H_1 and H_2 respectively. Thus, while an automaton is reading some word $a_1 a_2 \ldots a_n$ over Σ^*, the string

$$a_1 a_2 \ldots a_{k-1} \lceil a_k a_{k+1} \ldots a_{l-1} a_l \rfloor a_{l+1} \ldots a_n$$

with $w = a_1 a_2 \cdots a_n$ signifies that the left head is reading the symbol a_k and the right head is reading the symbol a_l.

If there is only one symbol between the two heads, in any move of the form $(x, y; s_1, s_2, \ldots, s_m; 1) \to (d_1, d_2; e_1, e_2, \ldots, e_m)$, x must be equal to y. This is because both heads would read the same symbol, so $x \neq y$ is not possible. In these cases only one of the heads is allowed to move and finish the input. Note that if one of the heads never moves, then the machine is of the type already considered in [1, 5].

An instantaneous description (ID) of M with input $¢w\$$ is a tuple

$$(c_1, c_2, \ldots, c_m, x\lceil y \rfloor z)$$

where c_i is the value of the i-th counter and $¢w\$ = xyz$ with the left head reading the first letter of y and the right head reading the last letter of y. Both x and z may be empty. The initial ID of M is $(0, 0, \ldots, 0, \lceil ¢w\$ \rfloor)$. We use $ID_1 \vdash ID_2$ to indicate the change in the ID after a single move of M. As usual, $\vdash^*$ denotes the reflexive, transitive closure of $\vdash$. The language accepted by M is

$$\{w \in \Sigma^* \mid (0, 0, \ldots, 0, \lceil ¢w\$ \rfloor) \vdash^* (0, 0, \ldots, 0, ¢u \rfloor \lceil v\$) \text{ with } w = uv\} .$$

In particular, if there is no transition corresponding to the reading of the two heads during a computation, then the input word is not accepted by the machine in that run.

Example 2.1. We show that a well-known non-context-free language is accepted by a 1-reversal WK-automaton with a single counter. Let $\Sigma = \{a, b, c\}$. Consider the stateless WK-automaton M with one counter whose moves are

$$\begin{array}{rcl}
(¢, \$; 0; \infty) & \to & (1, 1; 0) \\
(a, c; 0; \infty) & \to & (1, 1; +) \\
(a, c; 1; \infty) & \to & (1, 1; +) \\
(b, b; 1; \infty) & \to & (1, 0; -) \\
(b, b; 1; 2) & \to & (1, 0; -) \\
(b, b; 1; 1) & \to & (1, 0; -)
\end{array}$$

Then M is 1-reversal and accepts the language $\{a^n b^n c^n \mid n \geq 1\}$. About the accepted language: firstly, since both the heads move in opposite directions, at each step, if the two heads read simultaneously either a and a, c and c, a and b or b and c, no move is defined (indeed, in all these cases such a word does not belong to $\{a^n b^n c^n \mid n \geq 1\}$). The two heads read the blocks of a's and c's simultaneously, starting from the extremes, and moving in opposite directions. For each move in these blocks, when the heads are sufficiently far one from each other, the counter is increased by 1. Further, the machine allows the heads both to read b. The second head never moves again. Until the two heads are sufficiently far (fourth move), the second head does not move anymore, but the acceptance will be determined by the advancement of the first head, until it meets the second head. For each advancement, the counter is decreased by 1. Then if the number of the b's were the same of the a's (and of the c's), which is represented by the counter, the two heads will meet and the value of the counter will be 0 (acceptance). Otherwise, either the counter will assume a negative value (more b's than a's; and blocked computation without allowed movement with counter value 0 when b's are read) or counter 1 will assume a positive number, greater than of zero (less b's than a's). So the machine is clearly 1-reversal.

Let us see how it works on inputs abc and $aabbcc$:

$$(0, \lceil ¢abc\$ \rfloor) \vdash (0, ¢\lceil abc \rfloor \$) \vdash (1, ¢a\lceil b \rfloor c\$) \vdash (0, ¢ab\lceil \rfloor c\$).$$

$$(0, \lceil ¢aabbcc\$ \rfloor) \vdash (0, ¢\lceil aabbcc \rfloor \$) \vdash (1, ¢a\lceil abbc \rfloor c\$) \vdash (2, ¢aa\lceil bb \rfloor cc\$) \vdash$$
$$(1, ¢aab\lceil b \rfloor cc\$) \vdash (0, ¢aabb\lceil \rfloor cc\$).$$

Now we define a subclass of realtime WK-automata.

Definition 2.2. A realtime WK-automata is *strong realtime* iff each of the two heads moves at every move of the machine with $\varphi(M) \in \{\infty, 2\}$ and only the first head moves otherwise.

By definition the strong realtime machines are special realtime machines. One can consider WKC_0^0 machines also, these are without any counters. Their computing power is exactly the same as the stateless (type **N**) $5' \to 3'$ WK-automata of [12]. This language class is incomparable with the regular languages with respect to set theoretical inclusion: the language $a^*b^*c^*$ cannot be accepted, but the language of palindromes $\{w \mid w \in \{a, b\}^*,\ w = w^R\}$ is in WKC_0^0 as shown in [12]. In addition this language is accepted by a strong realtime automata. Now we give another example to demonstrate the power of strong realtime machines.

Example 2.2. Let $\Sigma = \{a, b, c, d\}$. Consider the stateless WK-automaton M with five counters whose moves are

$$\begin{array}{rcl}
(¢, \$; 0, 0, 0, 0, 0; \infty) & \to & (1, 1; 0, +, 0, 0, 0) \\
(a, d; 0, 1, 0, 0, 0; \infty) & \to & (1, 1; +, 0, 0, 0, 0) \\
(a, d; 1, 1, 0, 0, 0; \infty) & \to & (1, 1; +, 0, 0, 0, 0) \\
(b, c; 1, 1, 0, 0, 0; \infty) & \to & (1, 1; -, -, 0, +, 0) \\
(b, c; 1, 1, 0, 0, 0; 2) & \to & (1, 1; -, -, 0, 0, 0) \\
(a, c; 1, 1, 0, 0, 0; \infty) & \to & (1, 1; 0, -, +, 0, 0) \\
(a, c; 1, 0, 1, 0, 0; \infty) & \to & (1, 1; 0, 0, 0, 0, 0) \\
(b, c; 1, 0, 1, 0, 0; \infty) & \to & (1, 1; -, 0, -, +, 0) \\
(b, c; 1, 0, 1, 0, 0; 2) & \to & (1, 1; -, 0, -, 0, 0) \\
(b, c; 1, 0, 0, 1, 0; \infty) & \to & (1, 1; -, 0, 0, 0, 0) \\
(b, c; 1, 0, 0, 1, 0; 2) & \to & (1, 1; -, 0, 0, -, 0) \\
(b, d; 1, 1, 0, 0, 0; \infty) & \to & (1, 1; 0, -, 0, 0, +) \\
(b, d; 1, 0, 0, 0, 1; \infty) & \to & (1, 1; 0, 0, 0, 0, 0) \\
(b, c; 1, 0, 0, 0, 1; \infty) & \to & (1, 1; -, 0, 0, +, -) \\
(b, c; 1, 0, 0, 0, 1; 2) & \to & (1, 1; -, 0, 0, 0, -)
\end{array}$$

Then M is 1-reversal and accepts the language $\{a^n b^m c^n d^m \mid n, m \geq 1\}$. The first counter is counting up to $\min\{m, n\}$, while the other counters show the 'state' of the machine, i.e., the phase of the computation: counter 2 is positive in the initial phase (the heads read the blocks of a's and d's simultaneously). If $m = n$, then counter 4 represents the case: the same number of b's and c's should be read while counter 1 should be emptied. If $n > m$, then counter 3 is used, a's and c's are read simultaneously without changing the value of counter 1, then by finishing b's and c's counter 4 is used to finish the process. Similarly counter 5 indicates the case when $m > n$ and b's and d's are read by the heads, respectively. We give some examples that show how the machine accepts some short inputs.

$$(0, 0, 0, 0, 0, \lceil ¢abcd\$ \rfloor) \vdash (0, 1, 0, 0, 0, ¢\lceil abcd \rfloor \$) \vdash (1, 1, 0, 0, 0, ¢a\lceil bc \rfloor d\$) \vdash (0, 0, 0, 0, 0, ¢ab\lceil \rfloor cd\$).$$

$$(0, 0, 0, 0, 0, \lceil ¢abbcdd\$ \rfloor) \vdash (0, 1, 0, 0, 0, ¢\lceil abbcdd \rfloor \$) \vdash (1, 1, 0, 0, 0, ¢a\lceil bbcd \rfloor d\$) \vdash$$
$$(1, 0, 0, 0, 1, ¢ab\lceil bc \rfloor dd\$) \vdash (0, 0, 0, 0, 0, ¢abb\lceil \rfloor cdd\$).$$

$$(0, 0, 0, 0, 0, \lceil ¢aabccd\$ \rfloor) \vdash (0, 1, 0, 0, 0, ¢\lceil aabccd \rfloor \$) \vdash (1, 1, 0, 0, 0, ¢a\lceil abcc \rfloor d\$) \vdash$$
$$(1, 0, 1, 0, 0, ¢aa\lceil bc \rfloor cd\$) \vdash (0, 0, 0, 0, 0, ¢aab\lceil \rfloor ccd\$).$$

Note that with these more restricted machines we still can accept important non-context-free languages.

The families of nondeterministic/deterministic k-reversal m counter strong realtime WK-automata (and the language classes defined by them) are denoted by SRWKC_m^k and dSRWKC_m^k, respectively. The symbols ∞ and $*$ can similarly be used as for realtime/non-realtime machines.

3. Closure properties of stateless multicounter WK-automata languages

A number of closure properties of languages accepted by deterministic and nondeterministic stateless multicounter WK-automata follow.

Proposition 3.1. The language families accepted by deterministic and nondeterministic stateless strong realtime multicounter WK-automata are closed under intersection.

Proof:
Let M_1 and M_2 be two strong realtime multicounter WK-automata. Strong realtime machines read the input in a similar manner, therefore their heads are located exactly the same place after the same number of steps on any input (that could be processed by both machines). Therefore, if M_1 is a k_1-reversal m_1-counter machine and M_2 is a k_2-reversal m_2-counter machine, then we can construct a $\max\{k_1, k_2\}$-reversal, $(m_1 + m_2)$-counter machine which can simultaneously simulate M_1 and M_2: the first m_1 counters work exactly in the same way as at M_1, while the last m_2 counters work exactly in the same way as the counters of M_2. In this way, only those input will be accepted for which both M_1 and M_2 has an accepting run. Clearly the new machine is $\max\{k_1, k_2\}$ reversal; and when both M_1 and M_2 are deterministic, then so is the constructed machine. $\square$

Proposition 3.2. The language family accepted by nondeterministic stateless realtime multicounter WK-automata is closed under the union operation.

Proof:
If M_1 is a k_1-reversal m_1-counter machine and M_2 is a k_2-reversal m_2-counter machine, then we can construct a $\max\{k_1, k_2\}$-reversal, $(\max\{m_1, m_2\} + 2)$-counter machine which can simulate either M_1 or M_2 determined by a nondeterministic choice of the first step when exactly one of the two additional counters is set to 1 indicating which of the machines M_1 and M_2 is being simulated in this run. At the first step the machine may move as M_1 and set the first additional counter to 1. If M_2 is chosen to be simulated, then additionally to one of its initial move the second additional counter is set to 1. Then these last two counters do not change during the computation, only in accepting steps (with parameter $p \in \{1, 2\}$) are decreased. In this way, during the whole computation they keep track of which machine is being simulated. Thus, the moves simulating M_1 (except the initial ones) have the condition that the last counters are positive and zero, respectively. And similarly for M_2 they must be zero and positive, respectively. It is clear that the machine accept the union, i.e., it accepts the words that are accepted by at least one of the machines M_1 and M_2. Furthermore, the new machine is $\max\{k_1, k_2\}$-reversal since the additional counters make at most one reversal. Note that in case of $m_1 \neq m_2$ some of the counters never changed in some computations. $\square$

Proposition 3.3. The language families WKC$_m^x$, RWKC$_m^x$, SRWKC$_m^x$, dWKC$_m^x$, dRWKC$_m^x$, dSRWKC$_m^x$ ($m \in \mathbb{N}$, $x \in \mathbb{N} \cup \{*, \infty\}$) are closed under reversal (taking mirror image) of words.

Proof:
Suppose a machine accepts w with k-counters and m-reversals. Then w^R can also be accepted with the same parameters, just the behavior of the heads have to be interchanged. $\square$

Proposition 3.4. For $|\Sigma| \geq 2$, language families accepted by deterministic or nondeterministic stateless realtime or strong realtime multicounter WK-automata are not closed under the concatenation operation.

Proof:
Consider the language of even palindromes $L = \{ww^R \mid w \in \{a,b\}^*\}$. It can be accepted by a strong realtime deterministic WK-automaton without counters as shown in [12]. Its concatenation with itself is $L \cdot L = \{ww^R uu^R \mid w, u \in \{a,b\}^*\}$. The first head can read w and/or the second head can read u^R. The read part must be stored. Since an RWKC$_m^\infty$ (and so an SRWKC$_m^\infty$) machine has only memory by counters and it is realtime, it is not possible to store a word of arbitrary length as w or u^R could be. Without storing any (or both) of these words our stateless machine is unable to check their reverse. □

Corollary 3.1. The language family of deterministic/nondeterministic realtime stateless multicounter WK-automata is not closed under Kleene-closure.

4. Hierarchies for nondeterministic stateless multicounter WK-automata

In this section we start by proving that for stateless realtime multicounter WK-automata, nondeterministic and deterministic machines are not equivalent; the nondeterministic variations are more powerful, i.e.

Theorem 4.1. Nondeterministic stateless realtime multicounter WK-automata with k reversals and m counters accept more languages than deterministic stateless realtime multicounter WK-automata with the same parameters.

Proof:
Clearly all deterministic machines with m counters and k reversals are a special case of nondeterministic machines with the same number of counters and reversals. For $n \geq 0$ we define the language

$$L_n = \{a^i b^{j_1} a b^{j_2} \cdots a b^{j_n} \mid i, j_k \geq 0\,,\ k = 1, 2, \ldots, n, \text{ and } i = j_\ell \text{ where } \ell = 1, \text{ or } \ell = 2, \text{ or } \ldots, \ell = n\}.$$

Then $L' = \bigcup_{n=1}^{\infty} L_n$ cannot be accepted by a deterministic stateless realtime WK-automata with any number of reversals and any number of counters. The machine needs to test the values $\ell \in \{1, 2, \ldots, n\}$ to determine the value for which the equality holds. It is clear that for any numbers $k, m \in \mathbb{N}$: $w \in L_{2(k+1)(m+1)+1}$ cannot be accepted by a machine in WKC$_m^k$. To check which of the $2(k+1)(m+1)+1$ values matches to i cannot be done with m k-reversal counters in realtime.

Furthermore, in the nondeterministic case, only one counter is enough to accept the language. If $i = j_k$ for a $k \in \{1, 2, \ldots, n\}$, then the right head can read the end of the input till the end of the subword b^{j_k} without changing the counter, then at the beginning of that block (i.e., at the previous symbol a or \$ in case of $k = n$, i.e., at the last block) the counter is changed to 1. If the counter is positive, both heads move while reading symbols. The left head reads the prefix a^i while the right head reads b^{j_k}. Thus they are reading the same number of symbols. At the end of the blocks the counter set to be zero again only if both blocks ended at the same move. Then the remainder of the input can be read by the first head to finish the input. □

The following fact is a known result regarding to deterministic WK-automata (the details can be found in [2]).

Proposition 4.1. For fixed $i \geq 3$, languages in the form $L = \{a^{in} \mid n \geq 0\}$, cannot be accepted by any stateless deterministic realtime reversal bounded WK-automaton.

These languages are clearly accepted by nondeterministic stateless WK-automata with k-reversals and m-counters, where k, m depend on the value of i. If the machine is 1-reversal, $m = \lceil \frac{i}{2} \rceil - 1$ counters are needed to accept L (for $i = 1, 2, 5, 6, 7 \ldots$), and 2 counters for $i = 3$ and $i = 4$. It is clear that in the $i = 1, 2$ cases, $m = 0$ counters are needed. In the former (latter) case, the machine just reads the input with just one head (with both heads simultaneously), checking if the input is of the form a^* (or $(aa)^*$). Moreover this can be done by a deterministic machine.

If $i > 2$, then nondeterministic machines are used: they guess the value of n in the following way. If $i = 3$ or $i = 4$, then two counters are needed for a non-deterministic machine: First the machine uses the first counter: both head read a's and the counter is incremented. There is a nondeterministic choice to continue the work in this way, or jump to the other phase: set the second counter to 1 and decrease the first one. In case of $i = 3$ only the first head read the remained input, while in case of $i = 4$ both heads continue their work. In this phase of computation (the second counter is positive) the first counter is decremented in each movement. When the value of $\varphi(M)$ is 1 (at $i = 3$) or 2 (at $i = 4$), then this is the end the computation and both counters are decreased. The second counter becomes zero, the first one becomes zero only if the value n was correctly guessed and so the input word is in the language. (It could happen that the first counter is emptied before the heads met and therefore this run was not successful.)

If $i > 4$, then $m = \lceil \frac{i}{2} \rceil - 1 \geq 2$. We could distinct the first and the last counters (they will have special roles: starting and accepting). The machine works in the following manner. At the first phase, the first counter is used and incremented in each step, while both heads read a's. Then, by a nondeterministic choice, the machine may continue its work by decrementing the first counter and at the same time incrementing the second one. This phase goes till the first counter is emptied (or the input is processed and the machine gets stuck). When the first counter is empty, the machine starts to decrement the second counter in each step and incrementing the third one (if any). And so on, the new phase of the computation will start after the same number of steps that was guessed by the machine at the first phase (i.e. the maximal value of the first counter in this run). Finally, when the last counter is started to decrement only the first head reads the input if i is odd and both heads if i is even. The input gets accepted only if it was the desired form with the guessed number.

In this way, if i is even, then the machine reads with both heads simultaneously for the whole accepting process, using m counters. Then, for $i + 1$, one additional counter is needed. For every odd value of i, there is an $i+1$ even value, which needs the same number of counters. Thus, the equality $m = \lceil \frac{i}{2} \rceil - 1$ can be proven by induction.

Linear grammars and languages are closely connected to $5' \to 3'$ WK-automata [12, 13]. A linear grammar $G = (N, T, P, S)$ is in normal form if every rule has one of the following forms $A \to aBb, A \to aB, A \to Ba, A \to a$ with $A, B \in N$ and $a, b \in T$. This can be achieved by basic transformations of the rules.

Theorem 4.2. Let $G = (N, T, P, S)$ be a linear grammar in normal form. Then $L(G)$ can be accepted by a nondeterministic stateless realtime WK-automaton with unbounded number of reversals and $m = |N|$ counters so that $\sum_{i=1}^{m} c_i \leq 1$ at any time of the computation (where c_i is the value of the i-th counter).

Proof:
We give the construction of the machine which accepts $L(G)$. Each counter represents a nonterminal. Initially both heads make a step from the boundary markers and set the counter of S to 1 ($c_S = 1$). For every rule $X \to aYb \in P$ the move

$$((a,b;0,0,\ldots,0,s_X=1,0,\ldots,0),\infty) \to (1,1;0,0,\ldots,0,e_X=-,0,e_Y=+,0,\ldots,0)$$

is added. The order of c_X and c_Y in the sequence may be different depending on how the counters were assigned to the nonterminals. If $X = Y$, then the counters are not changed. For a rule $X \to aY \in P$ similar moves are added (for $\varphi(M) > 1$). This time the second head does not move: b can be any terminal symbol, so the moves should be constructed for all $b \in T$, and only the left head moves:

$$((a,b;0,0,\ldots,0,s_X=1,0,\ldots,0),\infty) \to (1,0;0,0,\ldots,0,e_X=-,0,e_Y=+,0,\ldots,0).$$

The case of $X \to Ya \in P$ is similar and $X \to a \in P$ can be handled with $\varphi(M) = 1$ by reading an a with the left input head and decreasing the value of counter c_X setting all counters to zero.

It is obvious that every successful derivation in G has an accepting run of the constructed WK-automaton, and vice-versa, therefore the machine accepts $L(G)$. □

Actually, fewer number of counters are sufficient; with a binary coding of non-terminal symbols a WK-automaton with $\lceil \log_2 |N| \rceil$ counters works.

A language is called *even-linear* if it can be generated by a grammar having rules only of the forms $A \to aBb, A \to a, A \to \lambda$, where A, B are nonterminals and a, b are terminals. (We refer to [13, 14] to a characterization of this language class by WK-automata.) Strong realtime machines are related to even-linear languages in a similar way as realtime machines are related to linear languages. The construction of the proof of the previous theorem leads to a strong realtime machine in case of even-linear grammar.

Remark 4.1. It is known that counter machines with a finite control and two counters accept all recursively enumerable languages [11]. Therefore we can add two additional counters to the automaton in Theorem 4.2 and obtain a machine equivalent to the Turing-machine. Consequently, by restricting their power, reversal boundedness plays a key role in the hierarchy of stateless multicounter WK-automata.

Let us define the following languages over $\Sigma = \{a,b,c\}$ for $j \geq 1$:

$$L_j = \{a^{n_1}ca^{n_2}c\cdots a^{n_j}cb^{n_1}c^{n_j}b^{n_2}c^{n_{j-1}}\cdots b^{n_{j-1}}c^{n_2}b^{n_j}c^{n_1} \mid n_i \geq 0 \text{ for all } i = 1,\ldots,j\}.$$

Theorem 4.3. For any $j \geq 1$, L_j is in dRWKC$_j^1$, but not in RWKC$_{j-1}^k$ for any $k \geq 1$.

Proof:
First we remark that without the delimiters c between the subwords a^{n_i}, only one counter is enough to accept L_j. The value $n_1 + n_2 + ... + n_j$ can be stored in the counter by reading all the a's by the left head. Then the lengths of the blocks of b's and c's can be checked by decrementing the counter (either head reads a b, the other must read a c at the same move).

It is easy to see that L_j can be accepted by a 1-reversal machine with j counters. The machine first counts the number of a's in the prefix a^{n_1} subword with counter 1, then, if a c is read, the second counter is increased by 1 to indicate that the a^{n_2} subword is going to be read. After reading a^{n_2} and increasing counter 2 in each step, another c comes. This time, while reading it, the second counter is decreased by 1 (it contained $n_2 + 1$, because of the first c) and the third counter is increased to indicate, that now a^{n_3} is to be read and counted. In this way, the machine counts all these a^{n_i} subwords and after the jth one, while reading a c with the left head, it also reads the right end marker, \$ with the right head, while decreasing the jth counter by 1 (this is also because of the previously read c). At this point, the value of counter i $(i = 1, \ldots, j)$ is n_i. After these steps, the subword $b^{n_1}c^{n_j}b^{n_2}c^{n_{j-1}} \cdots b^{n_{j-1}}c^{n_2}b^{n_j}c^{n_1}$ is processed while decrementing the corresponding counters.

We use induction on j to show that L_j cannot be accepted by $j - 1$ counters with any number of reversals. For $j = 1$, it is obvious that the language $L_1 = a^n cb^n c^n$ cannot be accepted without counters. Note, that the expression of L_{k+1} can be constructed from the expression of L_k in the following way:

1. let w and x be the sequences $b^{n_1}c^{n_k} \cdots b^{n_k}c^{n_1}$ and $a^{n_1}ca^{n_2}c \cdots a^{n_k}c$ respectively
2. $L_{k+1} = a^\ell cxb^\ell h(w)c^\ell$, where h is a morphism such that $h(b) = c$ and $h(c) = b$.

Suppose, that for some $k \geq 1$, L_k cannot be accepted with $k - 1$ counters with any number of reversals. For L_{k+1}, we have an additional counter. In this counter, we should store the number ℓ to test whether the first block of b's have the same number of symbols. So, the additional counter is needed to store ℓ. Then, after reading a^ℓ, or c^ℓ (or both) we have $k - 1$ counters and the remaining part of the input to be processed (without the b^ℓ subword) is in L_k, except that the corresponding b's and the c's in the suffix are exchanged. But this is not sufficient to accept L_k. Furthermore we know that L_k cannot be accepted with $k - 1$ counters. Thus it is impossible to get to b^ℓ without loss of information, so L_{k+1} cannot be accepted with k counters.

□

Corollary 4.1. The language classes dRWKC^1_m and RWKC$^k_{m-1}$ are incomparable (for $k \geq 1, m \geq 2$) under set theoretical inclusion.

Proof:
As we have shown in the proof of the previous theorem, L_m can be accepted by a 1-reversal m-counter machine, but cannot be accepted by a k-reversal $m - 1$ counter machine for any $k \geq 1$. On the other hand, the language L' that was in the proof of Theorem 4.1 is in RWKC$^k_{m-1}$ for $k \geq 1, m \geq 2$, since, as we have shown 1-reversal one counter is enough to accept it by a nondeterministic machine. But L' is not in dRWKC^1_m by the proof of Theorem 4.1. □

It is easy to prove that the following result holds.

Proposition 4.2. Stateless strong realtime multicounter WK-automata are less powerful than stateless realtime multicounter WK-automata with the same parameters.

To see this we consider that the language that is the concatenation of the language of palindromes over $\{a, b\}$ and the regular language c^*. This language needs only 1 counter to represent the phase when c^* is being read by the second head, so it is deterministic realtime 1-reversal 1-counter WK-automata language.

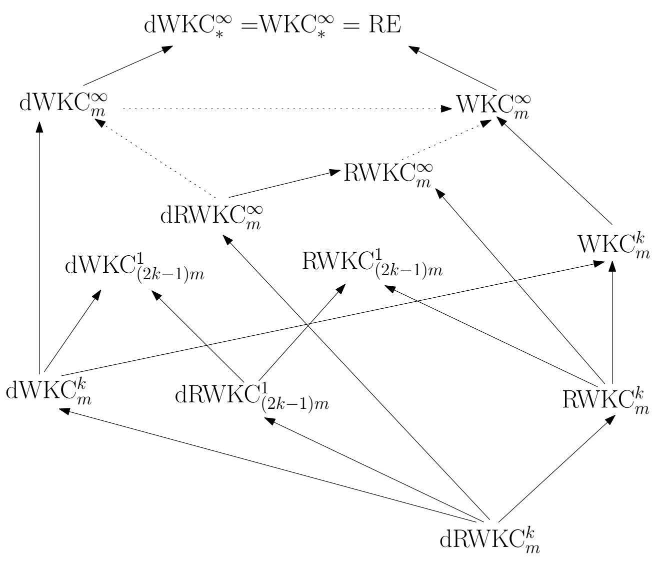
$\mathrm{dWKC}_*^\infty = \mathrm{WKC}_*^\infty = \mathrm{RE}$
dWKC_m^∞
WKC_m^∞
RWKC_m^∞
dRWKC_m^∞
$\mathrm{dWKC}^1_{(2k-1)m}$
$\mathrm{RWKC}^1_{(2k-1)m}$
WKC_m^k
dWKC_m^k
$\mathrm{dRWKC}^1_{(2k-1)m}$
RWKC_m^k
dRWKC_m^k

Figure 1. Hierarchy of stateless multicounter $5' \to 3'$ WK automaton languages.

5. Conclusions

We have considered multicounter $5' \to 3'$ WK-automata are considered. With arbitrary number of counters and without limiting the number of reversals the non-realtime versions are universal. We proved that nondeterministic variations are more powerful than deterministic variants with the same parameters. As expected, the restricted variants are usually less powerful than their not restricted variants. Reversal bounded machines accept a proper subset of the languages accepted by than unbounded reversal machines. Also, realtime machines are less powerful than non-realtime ones. k-reversal m counter machines can be simulated by 1-reversal automata with $(2k-1)m$ counters ([2]).

Figure 1 contains some of the hierarchy results about WK-automata. Arrows stand for proper inclusion. RE stands for the class of recursively enumerable languages.

Some open problems are represented by dotted arrows. Also minimal universal counter-machines can be considered analogously to minimal universal Turing-machines. Another further direction of research is to investigate such multicounter WK-automata where the heads do not sense when they meet, and therefore both heads read the full input in an accepting computation. Other extension can also be considered: multicounter WK-automata which may read strings and not just letters in a single transition, for example.

Acknowledgements

Comments of the anonymous reviewers helped improve the quality of this paper significantly and are therefore gratefully acknowledged. A portion of the results given here was presented in [15]. The research is supported by the TÁMOP 4.2.1/B-09/1/KONV-2010-0007 project, which is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European

Regional Development Fund.

References

[1] Eğecioğlu, Ö., Ibarra, O.H.: On stateless multicounter machines. Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS, vol. 5635, pp. 178–187. Springer, Heidelberg (2009)

[2] Eğecioğlu, Ö., Hegedüs, L., Nagy, B.: Stateless multicounter $5' \to 3'$ Watson-Crick Automata. In: Fifth International Conference on Bio-Inspired Computing: Theories and Applications, pp. 1599–1606. IEEE Press (2010)

[3] Freund, R., Păun, Gh., Rozenberg, G., Salomaa, A.: Watson-Crick Finite Automata. In: Third Annual DIMACS Symposium on DNA Based Computers, Philadelphia, pp. 535–546 (1994)

[4] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages and Computation. Series in Computer Science. Addison-Wesley, Reading, MA, 1979.

[5] Ibarra O.H., Eğecioğlu, Ö.: Hierarchies and characterizations of stateless multicounter machines. In: Ngo, H.G. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 408–417. Springer, Heidelberg (2009)

[6] Ibarra, O.H., Karhumäki, J., Okhotin, A.: On stateless multihead automata: hierarchies and the emptiness problem. In: Laber, E.S. et al. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 94–105. Springer, Heidelberg (2008)

[7] A. J. Korenjak and J. E. Hopcroft, Simple deterministic languages. Proc. of IEEE 7th Ann. Symp. on Switching and Automata Theory, 1966, pp. 36–46.

[8] M. Kutrib, H. Messerschmidt and Friedrich Otto, On stateless two-pushdown automata and restarting automata. *Proceedings of the 8th Automata and Formal Languages*, May, 2008, pp. 257–268.

[9] Leupold, P., Nagy, B.: $5' \to 3'$ Watson-Crick automata with several runs. In: Workshop on Non-Classical Models of Automata and Applications (NCMA), Wroclaw, Poland, pp. 167–180. book@ocg.at, Österreichischen Computer Gesellschaft (2009)

[10] Leupold, P., Nagy, B.: $5' \to 3'$ Watson-Crick automata with several runs. Fundamenta Informaticae 104, pp. 71–91 (2010) (extended version of [9]).

[11] Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Upper Saddle River, NJ, USA (1967)

[12] Nagy, B.: On a hierarchy of $5' \to 3'$ sensing WK finite automata languages. In: CiE 2009: Mathematical Theory and Computational Practice, Abstract Booklet, pp. 266–275. University of Heidelberg (2009)

[13] Nagy, B.: On $5' \to 3'$ sensing Watson-Crick finite automata. In: Garzon, M.H., Yan, H. (eds.) DNA 13. LNCS vol. 4848, pp. 256–262. Springer, Heidelberg (2008)

[14] Nagy, B.: $5' \to 3'$ sensing Watson-Crick finite automata. In: G. Fung (ed.) Sequence and Genome Analysis: Methods and Application II, iConcept Press Ltd. (2011), in print

[15] Nagy, B., Hegedüs, L., Eğecioğlu, Ö.: Hierarchy Results On Stateless Multicounter $5' \to 3'$ Watson-Crick Automata, IWANN 2011, LNCS vol. 6691, pp. 465-472. Springer, Heidelberg (2011)

[16] Păun, Gh.: Computing with Membranes. Journal of Computer and System Sciences 61/1, 108–143 (2000)

[17] Păun, Gh.: Computing with Membranes: An Introduction. Springer, Berlin (2002)

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

[19] Yang, L., Dang, Z., Ibarra, O.H.: On stateless automata and P systems. In: Pre-Proceedings of Workshop on Automata for Cellular and Molecular Computing, pp. 144–157 (2007)