Stateless multicounter 5# → 3# Watson–Crick automata: the deterministic case

László Hegedüs, Benedek Nagy & Ömer Eğecioğlu

Natural Computing
An International Journal

ISSN 1567-7818
Volume 11
Number 3

Nat Comput (2012) 11:361-368
DOI 10.1007/s11047-011-9290-9

Springer

Nat Comput (2012) 11:361–368
DOI 10.1007/s11047-011-9290-9

Stateless multicounter $5' \to 3'$ Watson–Crick automata: the deterministic case

László Hegedüs · Benedek Nagy · Ömer Eğecioğlu

Published online: 24 August 2011

Abstract We consider stateless counter machines which mix the features of one-head counter machines and special two-head Watson–Crick automata (WK-automata). These biologically motivated machines have heads that read the input starting from the two extremes. The reading process is finished when the heads meet. The machine is *realtime* or *non-realtime* depending on whether the heads are required to advance at each move. A counter machine is *k-reversal* if each counter makes at most k alternations between increasing mode and decreasing mode on any computation, and *reversal bounded* if it is k-reversal for some k. In this paper we concentrate on the properties of deterministic stateless realtime WK-automata with counters that are reversal bounded. We give examples and establish hierarchies with respect to counters and reversals.

Keywords Watson–Crick automata · Counter machine · DNA computing · Membrane computing · Reversal bounded machine · Stateless machine · Universality

L. Hegedüs · B. Nagy (✉)
Department of Computer Science, Faculty of Informatics, University of Debrecen, Debrecen 4032, Hungary
e-mail: nbenedek@inf.unideb.hu

L. Hegedüs
e-mail: hegedus.laszlo@inbox.com

Ö. Eğecioğlu
Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
e-mail: omer@cs.ucsb.edu

1 Introduction

Watson–Crick automata (Freund et al. 1994; Păun et al. 1998) are finite state machines motivated by nature. They work on DNA molecules, i.e., on double stranded sequences of bases. The strands of a DNA molecule have directions as a result of the underlying chemical bonds, determining the $5'$ and $3'$ ends of a strand. The two strands of a molecule are paired by hydrogen bonds if they have opposite directions and the sequence of bases match each-other, i.e., there is a one-to-one correspondence 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 (Leupold and Nagy 2009, 2010) without any restrictions. In $5' \to 3'$ Watson–Crick automata the reading heads start from the $5'$ ends of their strands, i.e., from opposite ends of the molecule regarding its physical body or mathematical description. These automata have been used to characterize linear context-free languages in (Nagy 2008). In this paper we consider only $5' \to 3'$ Watson–Crick automata, and consequently use the terminology *WK-automata* and omit the qualification $5' \to 3'$.

Stateless machines (i.e., machines with only one state) have been the subject of recent investigation because of their connection to certain aspects of membrane computing and P systems, a subarea of molecular computing that was introduced by Păun (1998, 2002). 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. The 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 our study of stateless multicounter WK-automata.

Stateless machines have no states to store information, and the move of such a machine depends only on the symbol(s) scanned by the input head(s) and the local portion of the memory unit(s). 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 PDA with states are equivalent to stateless nondeterministic PDA (where acceptance is by "null" stack) although this is not true for the deterministic case (Hopcroft and Ullman 1979; Korenjak and Hopcroft 1966) In (Ibarra et al. 2008; Yang et al. 2007) 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 (Kutrib et al. 2008), various types of stateless restarting automata and two-pushdown automata were compared to the corresponding machines with states.

If the machine is not allowed to make moves without moving the read head(s), then the model is called *realtime*. Otherwise the machine is *non-realtime* and can make moves that depend only on the contents of the counters without moving the heads. A counter machine is *k-reversal* if each counter makes at most k "full" alternations between increasing mode and decreasing mode and vice-versa on any computation (accepting or not), and is *reversal bounded* if it is k-reversal for some k.

Deterministic stateless (one-way) m-counter machines were investigated in (Eğecioğlu and Ibarra 2009), where hierarchies with respect to the number of counters and number of reversals were studied. Similar hierarchy results and characterizations are reported in (Ibarra and Eğecioğlu 2009) for the non-realtime versions. Hierarchies of the accepted language families by WK-automata are presented in (Nagy 2009) including stateless versions without counters. In this paper we concentrate on deterministic stateless realtime WK-automata with counters. We give examples and establish hierarchies of WK-automata with respect to counters and reversals.

2 Stateless multicounter WK-automata

The version of stateless WK-automata with counters has the following components. 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. 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 (zero or positive), and consists of moving the heads and at the same time incrementing or decrementing each of the counters. Depending on the types of head movements allowed, we obtain different classes of machines. Since there are no states, the acceptance of the input cannot be defined by final state. Instead, the input w is accepted by M if the counters are again zero when the heads *meet*.

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. Define

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

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

For a deterministic stateless multicounter WK-automaton M, a move

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

has the following parameters: $x, y \in \Sigma \cup \{¢, \$\}$ are the symbols under the heads H_1 and H_2, respectively; s_i is the sign of counter C_i:$s_i = 0$ if the ith counter is zero, $s_i = 1$ if it is positive. $s_1, s_2, \ldots, s_m$ is referred to as a *sign vector*; $d_1, d_2 \in \{0, 1\}$ indicate the direction of move of the heads with $d_1 + d_2 \leq p$. A 0 value 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 ith counter. Here $e_i = -$ is applicable only if $s_i = 1$. A move (1) 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 not both d_1 and d_2 are 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. In this paper we focus on deterministic realtime WK-automata with counters.

Every counter starts with value 0. Its value can only be increasing. It is in increasing mode till the first move that decreases its value. Then it is in decreasing mode until a move increases its value again (starting a new increasing

mode) or until the computation halts. An increasing mode followed by a decreasing mode is a "full" alternation. 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. Roughly speaking, k-reversal means that the number of change of a counter from increasing mode to decreasing mode is at most k for any counter of the automaton during any computation.

We denote the set of all (deterministic) 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 given by

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

while the unbounded reversal versions are denoted by WKC_m^∞ and RWKC_m^∞. This notation is also used for the corresponding language classes.

Remark 1 Consider a finite automaton with i states. A non-realtime WK automaton with reversal-unbounded counters can simulate this automaton with i counters. Assigning each counter to a state. The machine needs to use only the left reading head. Minsky showed that any Turing-Machine can be simulated by a finite automaton with 2 counters (Minsky 1967). Thus non-realtime WK automata with some number of unbounded reversal counters (some for the states of the finite automaton and at least two additional ones) can simulate any Turing-Machine.

Thus we may only consider automata with bounded reversals, and/or realtime machines and/or machine with bounded number of counters.

The formal definition of deterministic stateless multicounter WK automata is as follows.

Definition 1 A deterministic stateless multicounter WK-automaton is a quadruple $M = (\Sigma, \delta, ¢, \$)$ where Σ is a non-empty input alphabet, δ is a mapping from $(\Sigma \cup \{¢, \$\})^2 \times \{1,0\}^m \times \{0,1,2,\infty\}$ to $\{0,\ 1\}^2 \times \{0,\ +,\ -\}^m$ and $¢, \$ \notin \Sigma$ are two special symbols called endmarkers.

In the nondeterministic case, δ maps from $(\Sigma \cup \{¢, \$\})^2 \times \{1,0\}^m \times \{0,1,2,\infty\}$ to $P(\{0,\ 1\}^2 \times \{0,\ +,\ -\}^m)$ i.e., to the subsets of $\{0,\ 1\}^2 \times \{0,\ +,\ -\}^m$.

Remark 2 In the examples, the symbols $\lceil$ and $\rfloor$ are 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 $w = a_1 a_2 \ldots a_n$ signifies that the left head is reading the symbol a_k and the right head is reading the symbol a_l.

Note that if one of the heads never moves, then the machine is of the type already considered in (Eğecioğlu and Ibarra 2009; Ibarra and Eğecioğlu 2009). If both heads of M move, then $\lceil, H_1$ and $\rfloor, H_2$ can be used synonymously.

Remark 3 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.

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 ith counter and $w = xyz$ with the left head reading the first letter of y and the right head reading the last letter of y. 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. $\vdash^*$ denotes the reflexive, transitive closure of $\vdash$. The language accepted by M is

$$\{w \in \Sigma^* \mid (0, 0, \ldots, 0, \lceil ¢w\$ \rfloor) \vdash^+ (0, 0, \ldots, 0, ¢x \rfloor \lceil z\$) \text{ with } w = xz\}.$$

In this way $w \in L(M)$ iff there is a computation on input w such that all counters are zero and every letter of w was read by exactly one of the heads.

3 Examples and the 1-reversal case

We start with the following examples.

Example 1 The language of palindromes $L = \{w \mid w = w^R\}$ over an alphabet Σ is accepted by a stateless deterministic realtime WK-automaton without counters.

The machine only makes a move when the symbols under both heads are equal. If the length of the input word is odd, the last symbol can be read by either head. This way the machine accepts exactly the palindromes over Σ.

The language family det-2Lin (see (Nagy 2009), for more details on this family) is accepted by deterministic 2-head automata in such a way that the heads start from the opposite ends of the input and proceed until they meet.

By simulating deterministic finite automata, or deterministic 2-head (i.e., deterministic $5' \to 3'$ WK) automata, it is easy to show that all regular and all det-2Lin languages can be accepted by deterministic realtime multicounter WK-automata with unbounded reversals. The simulation

goes by binary counters, i.e., every counter refers to a unique state of the simulated automaton, noting that the initial configuration refers also for the starting state. When the machine senses that the input is fully read (i.e., in the last step), the counters used to simulate accepting states can be emptied.

Stateless deterministic realtime WK-automata with counters that are 1-reversal are already quite powerful as the following examples show:

Example 2 Let $\Sigma = \{a, b, c, d\}$. Consider the stateless WK-automaton M with two counters whose moves are

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

Then M is 1-reversal and accepts the language $\{ a^n b^n c^n d^n \mid n \geq 1 \}$. Let us see how the automaton works on inputs *abcd* and *aabbccdd*:

$$\begin{aligned} &(0, 0, \lceil \mathfrak{c} abcd\$ \rfloor) \vdash (0, 0, \mathfrak{c} \lceil abcd \rfloor \$) \vdash (1, 0, \mathfrak{c} a \lceil bc \rfloor d\$) \\ &\quad \vdash (0, 0, \mathfrak{c} ab \lceil \rfloor cd\$). \\ &(0, 0, \lceil \mathfrak{c} aabbccdd\$ \rfloor) \vdash (0, 0, \mathfrak{c} \lceil aabbccdd \rfloor \$) \vdash (1, 0, \mathfrak{c} a \lceil abbccd \rfloor d\$) \vdash \\ &(2, 0, \mathfrak{c} aa \lceil bbcc \rfloor dd\$) \vdash (1, 1, \mathfrak{c} aab \lceil bc \rfloor cdd\$ \rfloor) \vdash (0, 0, \mathfrak{c} aabb \lceil \rfloor ccdd\$) \ . \end{aligned}$$

The well known mildly context-sensitive, non-context-free language

$$\{a^n b^n c^n \mid n \geq 1\}$$

can also be accepted in a similar way.

Therefore non context-free languages can be accepted by a small number of counters and with only one reversal.

4 Restriction of reversal boundedness

For $w \in \Sigma^*$ and $a \in \Sigma$, define $|w|_a$ as the number of occurrences of a in w and consider the language $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$. It is known that L can be accepted by a stateless non-realtime 1-reversal 2-counter machine M but not by a stateless realtime k-reversal m-counter machine for any $k, m \geq 1$. We show that this result also holds for WK-automata.

Theorem 1 *The language $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$ cannot be accepted by a stateless realtime k-reversal m-counter WK-automaton for any $k, m \geq 1$.*

Proof Suppose L is accepted by some realtime k-reversal m-counter WK-automaton M. Then every word of the regular language $(ab)^*$ is accepted. If a word $(ab)^n$ is accepted by the machine in a way that only one of the heads moves while the other stays put, then the value of every counter is among 0, 1, 2, 3 at any time. This is because arbitrarily long words in $(ab)^*$ are accepted and when the machine senses that there are at most 2 letters to finish the input, it must be possible to decrease all counters to zero in two steps. Moreover reading any letter of the word, at least one of the counters must be increased/decreased for otherwise a word with an additional a or b would also be accepted. It follows that after at most $3m$ increasing steps there must be a one in which at least one counter decreases. Following this, after at most $3m$ decreasing steps there must be a step when a counter increases again. Then after at most $3m$ steps a counter must decrease again, and so on. Therefore for enough long word ($(ab)^{7m^2k}$ works) we can find a counter which makes more than k reversals. This contradiction gives the proof of the theorem for this case. If only the second head moves in M, the proof is similar.

Now we consider the case in which both heads move in the accepting computation on $(ab)^n$. Then there are two possibilities: if the machine M senses that their heads are close to each other when only one letter is being read to finish the input, then at this point every counter must be 0 or 1. Before this configuration both of the heads moved. There were three letters between them (configurations $(c_1, c_2, \ldots, c_m, \mathfrak{c} u \lceil aba \rfloor v\$)$ or $(c_1, c_2, \ldots, c_m, \mathfrak{c} u \lceil bab \rfloor v\$)$) and every counter value c_i was 0, 1 or 2. Since M has no knowledge that the input will be processed in two steps, the last but one configuration is reached in a deterministic manner, even if the input is longer. Thus, working on longer input words $u\ a(ba)^r\ v$ (or $u\ b(ab)^r\ v$) (with $r > 1$) the configuration $(c'_1, c'_2, \ldots, c'_m, \mathfrak{c} ua \lceil b(ab)^{r-1} \rfloor av\$)$ (or $(c'_1, c'_2, \ldots, c'_m, \mathfrak{c} ub \lceil a(ba)^{r-1} \rfloor bv\$)$ is obtained with counters having possible values only 0's and 1's. Then if $r = 2$, then the input must be accepted in at most 3 steps, therefore not any counters can have value more than 3 in this computation. Moreover at least one counter is changed in every step of the computation. When there are two letters between the heads when they sense their distance is small, then at this configuration the value of every counter is 0, 1 or 2. For longer input words the same holds if the heads read the same prefix and suffix of the input word. Therefore in this computation the possible values of counters cannot exceed 3.

In both cases for long enough words the accepting computation has more than k reversals, by a similar argument as we showed for the case when only one head moves during the computation. □

Remark 4 The language $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ of Theorem 1 can be accepted with a stateless deterministic realtime WK-automaton with unbounded reversals.

5 The unary case

A number of interesting combinatorial issues arise in the case of the unary alphabet $\Sigma = \{a\}$. It is known that the language $L = \{a^{2n} \mid n \geq 1\}$ cannot be accepted by any stateless deterministic non-realtime reversal bounded multicounter machine. However this language can be accepted by a stateless realtime WK-automaton *without* counters, with the following three deterministic rules:

$$(\text{¢}, \$; ; \infty) \to (1, 1;)$$
$$(a, a; ; \infty) \to (1, 1;)$$
$$(a, a; ; 2) \to (1, 1;)$$

On the other hand if the length is required to be a multiple of an integer $i \geq 3$, then the WK-automata can no longer accept this language as the following theorem shows.

Theorem 2 *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.*

Proof Let i be fixed. The language L is infinite. Since the machine is deterministic realtime and it senses that the input is nearly processed only when the heads are close enough (their distance is not greater than 2), in any accepting run the value of a counter cannot be more than i. This can be shown by the same method as used in the proof of Theorem 1. At every step of the machine some (at least one) counters change their values in order not to accept words with one or two additional a's. Then by similar argument as in the proof of Theorem 1 one can show that for long enough words of L the machine makes more than k reversals for any given k. □

Note that the languages in Theorem 2 are regular, therefore they can be accepted by stateless deterministic realtime machines with unbounded reversals.

We briefly consider the following combinatorial question: For 1-reversal stateless WK-automata with m counters over $\Sigma = \{a\}$ that accept singletons, what is the length of the longest string a^n that can be accepted as a function of m? A similar consideration was used to establish the counter hierarchy in the case of stateless deterministic realtime reversal bounded multicounter machines in (Eğecioğlu and Ibarra 2009). We omit the proof of the following result that holds for the WK-automata case.

Theorem 3 *The maximal length of the word a^n accepted by stateless deterministic 1-reversal m-counter WK-automaton that accepts a singleton is*

$$n = (m-1)2^{m+1} + 2m.$$

In fact the case of k reversals can be treated similarly following the proof sketch in (Eğecioğlu and Ibarra 2009). The maximum value of n in the case of stateless deterministic k-reversal m-counter WK-automata is found to be

$$n = ((2k-1)m - 1)2^{(2k-1)m+1} + 2(2k-1)m.$$

This dependence on m and k can be exploited to prove the hierarchy results with respect to realtime stateless multicounter WK-automata as given in Theorem 9.

6 Non-realtime machines

We give a number of results for the non-realtime case mostly without proofs.

The language $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ given in Theorem 1 can trivially be accepted by a 1-reversal non-realtime WK-automaton by accumulating the number of a's and b's in two counters, then simultaneously decreasing the counters after the heads meet.

Theorem 4 *The language $L = \{a^{n^2} \mid n \geq 0\}$ is accepted by a non-realtime deterministic stateless WK-automaton with four counters and unbounded reversals.*

Proof We use the fact that n^2 is the sum of the first n odd numbers. The main task is to read consecutive odd number of a's with a certain sequence of moves called an iteration. We will use a parity counter denoted by C_p. We need 3 more counters C_1, C_2, C_m respectively. The counter values $C_m = 0$ and $C_m = 1$ indicate whether C_1, or C_2 is to be used in the iteration.

Each odd number is of the form $2k + 1$ where k is $0, 1, 2, \ldots$. In the first iteration, only the counter C_p is incremented and we have the following sign vector: (1, 0, 0, 0). If there are no more a's to be read, the machine can now behave *non-realtime* and decrement C_p. Since 1 is a square, the word of only one a is accepted by the machine. If there are more a's to be read, first both heads move simultaneously each reading an a, decrementing C_p, while incrementing C_1. Then we get to the (0, 1, 0, 0) sign vector. Since C_p is zero, which indicates, that the machine has read an even number of a's in the iteration, one additional a must be read by one of the heads, while incrementing C_p and incrementing C_m. Now we have read the word *aaaa* and got the sign vector (1, 1, 0, 1). Then, like before, if there are no a's left, the counters are decremented in *non-realtime* mode, else the operation is continued.

A general move can be described as follows. The parity counter is always 1 at the beginning of the iteration except the first one, but that case is specified above. At the first step, C_p is decremented, while reading two a's and incrementing C_1 if C_m is 0 or C_2 if C_m is 1. Then we get two similar cases:

1. $C_m = 0$: in each step both heads move simultaneously, reading a's, while C_2 is decremented and C_1 is incremented. Thus the content of C_2 is moved to C_1. When C_2 is zero, C_1 contains the former contents of C_2 plus 1.
2. $C_m = 1$: Exactly the same as the $C_m = 0$ case, with the roles of C_1 and C_2 changed.

To finish the iteration, one additional a must be read, while C_p is incremented and C_m is decremented if it's value is 1, incremented if it's value is 0.

During the computation, at the end of any iteration, if no a's can be read, the machine can behave in *non-realtime* mode and decrements its counters.

The following rules are constructed this way:

$$(\not\vdash, \$; 0, 0, 0, 0; 2) \to (1, 1; 0, 0, 0, 0)$$
$$(\not\vdash, \$; 0, 0, 0, 0; \infty) \to (1, 1; 0, 0, 0, 0)$$
$$(a, a; 0, 0, 0, 0; 1) \to (1, 0; 0, 0, 0, 0)$$
$$(a, a; 0, 0, 0, 0; \infty) \to (1, 0; +, 0, 0, 0)$$
$$(a, a; 1, 0, 0, 0; \infty) \to (1, 1; -, +, 0, 0)$$
$$(a, a; 0, 1, 0, 0; \infty) \to (1, 0; +, 0, 0, +)$$
$$(a, a; 0, 1, 0, 0; 1) \to (1, 0; +, 0, 0, +)$$
$$(a, a; 1, 1, 0, 1; \infty) \to (1, 0; -, 0, +, 0)$$
$$(a, a; 1, 1, 0, 1; 1) \to (1, 0; -, 0, +, 0)$$
$$(a, a; 0, 0, 1, 1; 1) \to (1, 0; +, 0, 0, 0)$$
$$(a, a; 1, 0, 1, 0; \infty) \to (1, 1; -, +, 0, 0)$$
$$(a, a; 0, 1, 1, 0; \infty) \to (1, 1; 0, +, -, 0)$$
$$(a, a; 0, 1, 1, 1; \infty) \to (1, 1; 0, -, +, 0)$$
$$(a, a; 0, 0, 1, 1; \infty) \to (1, 0; +, 0, 0, -)$$
$$(a, a; 1, 0, 0, 0; 0) \to (0, 0; -, 0, 0, 0)$$
$$(a, a; 1, 0, 1, 0; 0) \to (0, 0; 0, 0, -, 0)$$
$$(a, a; 1, 1, 0, 1; 0) \to (0, 0; 0, -, 0, -)$$
$$(a, a; 1, 1, 0, 0; 0) \to (0, 0; 0, -, 0, 0)$$

Example 3 We give an example of an execution of the machine described in the proof of Theorem 4 Suppose the input is a^{16}. Then the machine makes the following sequence of moves:

$$(0, 0, 0, 0, \lceil \not\vdash aaaaaaaaaaaaaaaa\$\rfloor) \vdash (0, 0, 0, 0, \not\vdash\lceil aaaaaaaaaaaaaaaa\rfloor\$) \vdash$$
$$(1, 0, 0, 0, \not\vdash a\lceil aaaaaaaaaaaaaaa\rfloor\$) \vdash (0, 1, 0, 0, \not\vdash aa\lceil aaaaaaaaaaaaa\rfloor a\$) \vdash$$
$$(1, 1, 0, 1, \not\vdash aaa\lceil aaaaaaaaaaa\rfloor a\$) \vdash (0, 1, 1, 1, \not\vdash aaaa\lceil aaaaaaaaa\rfloor aa\$) \vdash$$
$$(0, 0, 2, 1, \not\vdash aaaaa\lceil aaaaaaaa\rfloor aaa\$) \vdash (1, 0, 2, 0, \not\vdash aaaaaa\lceil aaaaaaa\rfloor aaa\$) \vdash$$
$$(0, 1, 2, 0, \not\vdash aaaaaaa\lceil aaaaa\rfloor aaaa\$) \vdash (0, 2, 1, 0, \not\vdash aaaaaaaaa\lceil aaa\rfloor aaaaa\$) \vdash$$
$$(0, 3, 0, 0, \not\vdash aaaaaaaaa\lceil a\rfloor aaaaaa\$) \vdash (1, 3, 0, 1, \not\vdash aaaaaaaaaa\rfloor\lceil aaaaaa\$) \vdash^*$$
$$(0, 0, 0, 0, \not\vdash aaaaaaaaaa\rfloor\lceil aaaaaa\$)$$

Note that each time, if C_p is 1, a square number of a's have been read. Since the machine can only decrease it's counters when C_p is 1, only words consisting of square number of a's are accepted by it.

Thus the machine works in the expected way. It would be interesting to see if this language can be accepted with fewer than four counters.

Similar to the above result, we can prove:

Theorem 5 *$L = \{a^{n^2} \mid n \geq 0\}$ can be accepted by a non-realtime deterministic stateless WK-automaton with four counters and unbounded reversals.*

Proof The machine has four counters, C_1, C_2, C_3, C_m, respectively. As a starting move, one step is made by both heads, reading an a, while the counters C_1 and C_m increased. Now, the sign vector is (1, 0, 0, 1). The other steps are made as follows:

If C_1 and C_m are both not zero, then decrease C_1, while increasing C_2 and C_3 simultaneously and reading a's with both heads. If C_1 is zero and C_m is one, that means the machine has read the next power of two number of a's, which may be called j and can start setting C_1 to $2j$. This is done, when C_1 is zero, C_m is one and C_2, C_3 both contain some number. So the sign vector is (0, 1, 1, 1). Then in *non-realtime* mode, C_2 is decreased, while C_1 is increased. Note, that C_m has to be decreased to zero in the first step too. After a few steps, the sign vector is (1, 0, 1, 0). Now, the contents of C_3 must be moved to C_1 in a similar way, giving the sign vector (1, 0, 0, 0) and C_1 now contains the sum of the values of C_2 and C_3. Then C_m is increased by one, so that the following iteration can be started.

We specgified, that in the first iteration, two a's are read and C_1's value is set to one. By induction, if C_1's value is j and C_2, C_3 are both zero, a^{2j} is read, while C_2,C_3 are set to j. In the next iteration C_1 will contain $j + j$ and a^{4j} is read and so on.... Thus in each iteration the machine reads twice the a's as in the previous iteration.

The computation can be ended using *non-realtime* mode, when C_1 is zero and no more a's can be read.

The rules for accepting this language are:

$$(\not\vdash, \$; 0, 0, 0, 0; 2) \to (1, 1; 0, 0, 0, 0)$$
$$(\not\vdash, \$; 0, 0, 0, 0; \infty) \to (1, 1; 0, 0, 0, 0)$$
$$(a, a; 0, 0, 0, 0; 2) \to (1, 1; 0, 0, 0, 0)$$
$$(a, a; 0, 0, 0, 0; \infty) \to (1, 1; +, 0, 0, +)$$
$$(a, a; 1, 0, 0, 1; \infty) \to (1, 1; -, +, +, 0)$$
$$(a, a; 1, 0, 0, 1; 2) \to (1, 1; -, +, +, 0)$$
$$(a, a; 1, 1, 1, 1; \infty) \to (1, 1; -, +, +, 0)$$
$$(a, a; 1, 1, 1, 1; 2) \to (1, 1; -, +, +, 0)$$
$$(a, a; 0, 1, 1, 1; \infty) \to (0, 0; +, -, 0, -)$$
$$(a, a; 1, 1, 1, 0; \infty) \to (0, 0; +, -, 0, 0)$$
$$(a, a; 1, 0, 1, 0; \infty) \to (0, 0; +, 0, -, 0)$$
$$(a, a; 1, 0, 0, 0; \infty) \to (0, 0; 0, 0, 0, +)$$
$$(a, a; 0, 1, 1, 1; \infty) \to (0, 0; 0, -, -, -)$$
$$(a, a; 0, 1, 1, 0; \infty) \to (0, 0; 0, -, -, 0)$$

Example 4 The machine does the following steps in the case of a^{16}:

$(0,0,0,0,\lceil ¢aaaaaaaaaaaaaaaa\$\rfloor) \vdash (0,0,0,0,¢\lceil aaaaaaaaaaaaaaaa\rfloor\$) \vdash$
$(1,0,0,1,¢a\lceil aaaaaaaaaaaaaaa\rfloor a\$) \vdash (0,1,1,1,¢aa\lceil aaaaaaaaaaaaa\rfloor aa\$) \vdash$
$(1,0,1,0,¢aa\lceil aaaaaaaaaaaaa\rfloor aa\$) \vdash (2,0,0,0,¢aa\lceil aaaaaaaaaaaaa\rfloor aa\$) \vdash$
$(2,0,0,1,¢aa\lceil aaaaaaaaaaaaa\rfloor aa\$) \vdash (1,1,1,1,¢aaa\lceil aaaaaaaaaa\rfloor aaa\$) \vdash$
$(0,2,2,1,¢aaaa\lceil aaaaaaaa\rfloor aaaa\$) \vdash (1,1,2,0,¢aaaa\lceil aaaaaaaa\rfloor aaaa\$) \vdash$
$(2,0,2,0,¢aaaa\lceil aaaaaaaa\rfloor aaaa\$) \vdash^* (4,0,0,1,¢aaaa\lceil aaaaaaaa\rfloor aaaa\$) \vdash$
$(3,1,1,1,¢aaaaa\lceil aaaaaa\rfloor aaaaa\$) \vdash^* (0,4,4,1,¢aaaaaaaa\rfloor\lceil aaaaaaaa\$) \vdash^*$
$(0,0,0,0,¢aaaaaaaa\rfloor\lceil aaaaaaaa\$)$

As our previous examples show, there are non-semilinear languages that are accepted by non-realtime WK-automata. The following theorem holds both for realtime and non-realtime machines and analogous with the results of (Eğecioğlu and Ibarra 2009; Ibarra and Eğecioğlu 2009).

Theorem 6 *For every stateless deterministic (realtime/non-realtime) WK-automaton with k reversals and m counters, there exists a 1-reversal stateless (realtime/non-realtime) WK-automaton with $(2k-1)m$ counters that accepts the same language.*

The proof is given in (Eğecioğlu and Ibarra 2009) in the proof of Theorem 4 of that paper for stateless multicounter automata, which can be applied to WK-automata as well. The main idea is that if a counter does one reversal and would increase again, some other counter can take its place thus avoiding multiple reversals. This way one counter with k reversals can be substituted by k counters with one reversal. Further $(k-1)$ counters are needed to indicate which additional counter is being used. Thus to simulate m counters, $k+(k-1)=2k-1$ times m counters are needed.

Theorem 7 *There exist infinitely many regular languages that are not accepted by any* realtime, *or* non-realtime *k-reversal m-counter stateless WK-automaton for any fixed m, $k \geq 0$.*

Proof By Theorem 6 we can assume that $k=1$. Consider a finite language L over some alphabet V, which can be accepted by a stateless WK-automaton with m counters and 1 reversal. Select some word $s \in L$, which is accepted by using the most counters. Then s can be written in the form xyz, where y is some subword, which is processed by using l counters, x, z are some (possibly empty) words over V. Then the language $L' = \{xy^iz \mid i \geq 0\}$ is regular and any word in the form xy^iz can only be accepted by $m + l(i-1)$ counters. Since there are infinitely many non negative integers, there is always a word $w \in L'$ that can not be accepted by m counters for any fixed $m \geq 0$. □

Example 5 The language $(ab)^*$ is a simple regular language, but it cannot be accepted by any (realtime or non-realtime) stateless multicounter WK-automaton which is reversal bounded. (See the proof of Theorem 1 also.)

Theorem 8 *The language $L = \{ww \mid w \in \{a,b\}^*\}$ cannot be accepted by any stateless deterministic non-realtime multicounter k-reversal WK-automaton with m counters for any fixed k, $m \geq 0$.*

Proof Suppose that some stateless WK-automaton M accepts L. Then there exists some $x \in \{0,1\}^*$ such that $xx \in L$ and is accepted by M doing k reversals and using all m counters.

We can assume, that there is a sequence 01 (or 10) in x. As seen in Theorem 7, these sequences can only be processed while using some (at least one) counters because we don't have any states to indicate that there is a 1 to be read in the next step. Then x can be written in the form $x'01x''$.

If the machine accepts the language L, it should also accept $x'(01)^ix''x'(01)^ix''$ for all $i \geq 1$, thus making more than k reversals, which is a contradiction. □

We have the following hierarchy results:

Theorem 9 *The following hierarchy results hold (where RE denotes the class of Recursively Enumerable languages):*

1. $\mathrm{RWKC}^{\infty}_{*} \subset \mathrm{WKC}^{\infty}_{*} = \mathrm{RE}$,
2. $\mathrm{RWKC}^{*}_{*} \subset \mathrm{WKC}^{*}_{*}$,
3. $\mathrm{WKC}^{k}_{m} \subseteq \mathrm{WKC}^{1}_{(2k-1)m}$,
4. $\mathrm{WKC}^{k}_{m} \subset \mathrm{WKC}^{k}_{m+1}$.
5. $\mathrm{WKC}^{k}_{m} \subset \mathrm{WKC}^{k+1}_{m}$ (for $k < 2^{m-1}/m$).

7 Future work

The power of WK-automata lies somewhere between stateless one-head counter machines and stateless two-head counter machines. There remain a number of interesting aspects of the hierarchy of languages accepted by stateless multicounter WK-automata: non-realtime, nondeterministic versions and the case of unbounded reversals are among these. It is also of interest to identify the nature of separating languages for each of these classes. Work is in progress on various extensions of the properties considered in the present paper, as well as closure properties of the corresponding language classes. A detailed study of these will appear (Nagy et al. 2011).

Acknowledgements This paper is an extended version of (Eğecioğlu et al. 2010). The work is supported by the TÁMOP 4.2.1./B-09/1/KONV-2010-0007 project. The project is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund.

References

Eğecioğlu Ö, Ibarra OH (2009) On stateless multicounter machines. In: CiE 2009, LNCS, vol 5635. Springer, Heidelberg, pp 178–187

Eğecioğlu Ö, Hegedüs L, Nagy B (2010) Stateless multicounter $5' \to 3'$ Watson–Crick automata. In: 5th IEEE BIC-TA, pp 1599–1606

Freund R, Păun G, Rozenberg G, Salomaa A (1994) Watson–Crick finite automata. In: Proceedings of third annual DIMACS symposium on DNA based computers, pp 535–546

Hopcroft JE, Ullman JD (1979) Introduction to automata theory, languages and computation. Addison-Wesley, Reading

Ibarra OH, Eğecioğlu Ö (2009) Hierarchies and characterizations of stateless multicounter machines. In: COCOON 2009, LNCS 5609. Springer, Heidelberg, pp 408–417

Ibarra OH, Karhumäki J, Okhotin A (2008) On stateless multihead automata: hierarchies and the emptiness problem. In: LATIN'08, LNCS 4957. Springer, Heidelberg, pp 94–105

Korenjak AJ, Hopcroft JE (1966) Simple deterministic languages. In: FOCS, IEEE computer society, Los Alamitos, CA, pp 36–46

Kutrib M, Messerschmidt H, Otto F (2008) On stateless two-pushdown automata and restarting automata. In: 8th AFL, pp 257–268

Leupold P, Nagy B (2009) $5' \to 3'$ Watson–Crick automata with several runs. In: NCMA, pp 167–180

Leupold P, Nagy B (2010) $5' \to 3'$ Watson–Crick automata with several runs. Fundamenta Informaticae 104:71–91

Minsky ML (1967) Computation: finite and infinite machines. Prentice-Hall Inc., Upper Saddle River, NJ

Nagy B (2008) On $5' \to 3'$ sensing Watson–Crick finite automata. In: DNA 13, LNCS 4848. Springer-Verlag, Berlin, Heidelberg, pp 256–262

Nagy B (2009) On a hierarchy of $5' \to 3'$ sensing WK finite automata languages. In: CiE 2009, pp 266–275

Nagy B, Hegedüs L, Eğecioğlu Ö (2011) Hierarchy results on stateless multicounter $5' \to 3'$ Watson–Crick automata. In: IWANN 2011, LNCS 6691. Springer, Heidelberg, pp 465–472

Păun G (1998) Computing with membranes. Technical report, TUCS

Păun G (2002) Membrane computing: an introduction. Springer-Verlag, Heidelberg

Păun G, Rozenberg G, Salomaa A (1998) DNA computing: new computing paradigms. Springer-Verlag, Heidelberg

Yang L, Dang Z, Ibarra OH (2007) On stateless automata and P systems. In: Workshop on automata for cellular and molecular computing, pp 144–157