On Stateless Multicounter Machines[⋆]

Ömer Eğecioğlu and Oscar H. Ibarra[⋆⋆]

Department of Computer Science
University of California, Santa Barbara, CA 93106, USA
Email: {omer, ibarra}@cs.ucsb.edu

Abstract. We investigate the computing power of stateless multicounter machines with reversal-bounded counters. Such a machine has m-counters operating on a one-way input delimited by left and right end markers. The move of the machine depends only on the symbol under the input head and the signs of the counters (zero or positive). At each step, every counter can be incremented by 1, decremented by 1 (if it is positive), or left unchanged. An input string is accepted if, when the input head is started on the left end marker with all counters zero, the machine eventually reaches the configuration where the input head is on the right end marker with all the counters again zero. Moreover, for a specified k, no counter makes more than k-reversals (i.e., alternations between increasing mode and decreasing mode and vice-versa) on any computation, accepting or not. We mainly focus our attention on deterministic realtime (the input head moves right at every step) machines. We show hierarchies of computing power with respect to the number of counters and reversals. It turns out that the analysis of these machines gives rise to rather interesting combinatorial questions.

Keywords: Stateless multicounter machines, reversal-bounded, realtime computation, hierarchies.

1 Introduction

There has been recent interest in studying stateless machines (which is to say machines with only one state) because of their connection to certain aspects of membrane computing [8], a subarea of molecular computing that was introduced in a seminal paper by Gheorge Păun [6] (see also [7]).

Since stateless machines have no states, 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). Acceptance of an input string has to be defined in a different way. For example, in the case of a pushdown automaton (PDA), acceptance is by "null" stack. It is well known that nondeterministic PDAs with states are equivalent to stateless nondeterministic PDAs [1]. However, this is not true for the deterministic case [4]. For Turing Machines, where acceptance is when the

[⋆] This research was supported in part by NSF Grants CCF-0430945 and CCF-0524136.
[⋆⋆] Corresponding author

machine enters a halting configuration, it can be shown that the stateless version is less powerful than those with states. In [3,8] the computing power of stateless multihead automata were investigated with respect to decision problems and head hierarchies. For these devices, the input is provided with left and right end markers. The move depends only on the symbols scanned by the input heads. The machine can be deterministic, nondeterministic, one-way, two-way. An input is accepted if, when all heads are started on the left end marker, the machine eventually reaches the configuration where all heads are on the right end marker. In [5], various types of stateless restarting automata and two-pushdown automata were compared to the corresponding machines with states.

In this paper, we investigate the computing power of stateless multicounter machines with reversal-bounded counters. Such a machine has m-counters operating on a one-way input delimited by left and right end markers. The move of the machine depends only on the symbol under the input head and the signs of the counters, which indicate if the counter is zero or not. An input string is accepted if, when the input head is started on the left end marker with all counters zero, the machine eventually reaches the configuration where the input head is on the right end marker with all the counters again zero. Moreover, the machine is *k-reversal bounded*: i.e. for a specified k, no counter makes more than k alternations between increasing mode and decreasing mode and vice-versa in any computation, accepting or not. In this paper, we mainly study deterministic realtime (the input head moves right at every step) machines. We show hierarchies of computing power with respect to the number of counters and reversals.

2 Stateless Multicounter Machines

A deterministic stateless one-way m-counter machine operates on an input of the form ¢w\$, where ¢ and \$ are the left and right end markers. At the start of the computation, the input head is on the left end marker ¢ and all m counters are zero. The moves of the machine are described by a set of rules of the form:

$$(x, s_1, .., s_m) \to (d, e_1, ..., e_m)$$

where $x \in \Sigma \cup \{¢, \$\}$, Σ is the input alphabet, s_i = sign of counter C_i (0 for zero and 1 for positive), d = 0 or 1 (direction of the move of the input head: d = 0 means don't move, d =1 means move head one cell to the right), and $e_i = +, -$, or 0 (increment counter i by 1, decrement counter i by 1, or do not change counter i), with the restriction that $e_i = -$ is applicable only if $s_i = 1$. Note that since the machine is deterministic, no two rules can have the same left hand sides.

The input w is accepted if the machine reaches the configuration where the input head is on the right end marker \$ and all counters are zero. The machine is k-reversal if it has the property that each counter makes at most k "full" alternations between increasing mode and decreasing mode and vice-versa on any computation, accepting or not. Thus, e.g., $k = 2$ means a counter can only go from increasing to decreasing to increasing to decreasing.

There are two types of stateless machines: in *realtime machines* $d = 1$ in every move, i.e., the input head moves right at each step. In this case all the counters are zero when the input head reaches \$ for acceptance. In *non-realtime machines* d can be 0 or 1. In particular, when the input head reaches \$, the machine can continue computing until all counters become zero and then accept. In this paper, we are mainly concerned with deterministic realtime machines.

3 Stateless Realtime Multicounter Machines

We will show that these devices are quite powerful, even in the unary input alphabet case of $\Sigma = \{a\}$.

Since the machine is realtime, in each rule, $(x, s_1, .., s_m) \rightarrow (d, e_1, ..., e_m)$, $d = 1$. So there is no need to specify d and we can just write the rule as $(x, s_1, ..., s_m) \rightarrow (e_1, ..., e_m)$.

We refer to a vector of signs $v = (s_1, .., s_m)$ that may arise during the computation of an m-counter machine as a *sign-vector*. Thus a sign-vector is a binary string $s_1 \cdots s_m$ of length m, or equivalently a subset S of the set $\{1, 2, \ldots, m\}$. The correspondence between a subset S and a sign vector v is given by putting $s_i = 1$ iff $i \in S$. The string 0^m as a sign-vector signifies that all counters are zero. We will use both binary strings and subsets of $\{1, 2, \ldots, m\}$ interchangeably in describing sign-vectors. For $w \in \Sigma^*$ and $a \in \Sigma$, we define $|w|_a$ as the number of occurrences of a in w.

Theorem 1. *The language L over $\Sigma = \{a\}$ accepted by a stateless realtime multicounter machine M is either infinite or a singleton.*

Proof. We show that

1. if $(¢, 0^m) \rightarrow 0^m$ is a move of M then M accepts ε or Σ^*,
2. if $(¢, 0^m) \rightarrow S$ with $S \neq 0^m$ for some S is a move of M, then M accepts a singleton or an infinite language.

In the first case, it is possible that M has no move of the form $(a, 0^m) \rightarrow (e_1, ..., e_m)$. Then an accepting computation must proceed from ¢ to \$ directly; i.e. the accepted input is ¢\$. However if M does have the move $(a, 0^m) \rightarrow 0^m$, then M accepts Σ^*.

Suppose now $(¢, 0^m) \rightarrow S_1$ with $S_1 \neq 0^m$ is a rule of M and that M halts. Let t be the smallest integer such that M accepts a^t. Then $t > 0$ and when the machine is reading the a immediately before \$, the sign-vector is some nonempty S_t, and the counter values are 1 for counters corresponding to 1's in S_t, and 0 otherwise.

We consider the two subcases depending on whether or not $(a, 0^m) \rightarrow 0^m$ is a rule of M. If $(a, 0^m) \rightarrow 0^m$ is a rule of M then M accepts $a^t\Sigma^*$. If $(a, 0^m) \rightarrow S$ with $S \neq 0^m$ is a rule of M then let $r > 0$ be the smallest integer such that M accepts a^{t+r}. If there is no such integer, then M accepts the singleton a^t. Otherwise $a^{t+kr} \in L$ for $k \geq 0$. □

3.1 1-Reversal Machines

We are interested in machines accepting only a singleton language of the form $L = \{a^n\}$. By a rather intricate analysis, we are able to derive the precise value of the maximum such n. Moreover, we are able to show that the program of the machine achieving this n is unique (up to relabelling of the counter indices). Note that we can interpret a^n as the "maximum" number that a 1-reversal m-counter machine can count.

By Theorem 1, if $(¢, 0^m) \to 0^m$, then a realtime stateless machine accepts either ε or Σ^*. Therefore an accepting computation of an m-counter machine that accepts a non-null singleton can be represented in the form

$$0^m \to S_1 \to S_2 \to \cdots \to S_t \to 0^m \tag{1}$$

where $S_i \neq 0^m$ for $1 \leq i \leq t$ and the arrows indicate the sequence of sign-vectors after each move of the machine. Such a machine accepts the singleton language $\{a^t\}$, or equivalently can count up to t.

Borrowing terminology from the theory of Markov chains, we classify the nonempty sign-vectors in (1) as *transient* or *recurrent*, as follows:

1. S is transient if it appears exactly once in (1),
2. S is recurrent if it appears more than once in (1).

Thus a transient S contributes exactly one move, whereas a recurrent S is "re-entered" more than once during the course of the computation of M.

We prove a few lemmas characterizing possible computations of a 1-reversal machine accepting a singleton.

Lemma 1. *If S is transient and S' appears between two occurrences of S in (1), then $S \subseteq S'$.*

Proof. Suppose $j \in S \setminus S'$. Then counter C_j zero at the start of the computation, it is nonzero at the first appearance of S, and is zero at or before the appearance of S' in the computation. Since M is 1-reversal, it is not possible for counter C_j to be nonzero again at the second occurrence of S. □

Lemma 2. *If S is transient and S' appears between two occurrences of S in (1), then $S' = S$.*

Proof. Suppose that the first occurrence of S is followed by S''; i. e.

$$S \to S'' \to \cdots \to S' \to \cdots \to S$$

We first show that $S'' \subseteq S$. By way of contradiction, assume $j \in S'' \setminus S$. Then at S the counter C_j must be incremented from 0 to 1. But this cannot happen more than once at the two S's since M is 1-reversal. Therefore $S'' \subseteq S$. By Lemma 1, $S \subseteq S''$, and therefore $S = S''$. It follows that all sign-vectors S' between two occurrences of S are equal to S. □

The next lemma gives an upper bound on the number of distinct sign-vectors that can appear in (1).

Lemma 3. *Suppose $S_1, S_2, \ldots, S_d$ are the distinct non-null sign-vectors that appear in the computation of a 1-reversal machine M with m counters that accepts a singleton. Then $d \leq 2m-1$.*

Proof. Put $S_0 = S_{d+1} = 0^m$ and consider the $m \times (d+2)$ binary matrix B where the j-th column is S_{j-1} for $1 \leq j \leq d+2$. Since M is 1-reversal, each row of B has at most one interval consisting of all 1's. Therefore there are a total of at most m horizontal intervals of 1's in B. These together have at most $2m$ endpoints: i.e. a pair 01 where a such an interval of 1's starts, and a pair 10 where an interval ends. Since the S_j are distinct, going from S_j to S_{j+1} for $0 \leq j \leq d$ must involve at least one bit change, i.e. at least one of the $2m$ pairs of 01 and 10's. It follows that $d+1 \leq 2m$. □

Lemma 4. *Suppose at the first occurrence of the sign-vector S in (1), the values of the counters are $v_1(S) \geq v_2(S) \geq \cdots \geq v_m(S) \geq 0$. Then*

1. *$v_1(S) - 1$ is an upper bound to the number of times M makes a move from S back to S.*
2. *$v_1(S) + v_2(S)$ is an upper bound to the largest counter value when M makes a move from S to some S'.*

Proof. The lemma is a consequence of the fact that since M halts, some counter for any non-null sign-vector S must be decremented. □

Lemmas 3 and 4 immediately give an upper bound on how high a 1-reversal, m-counter machine can count. We necessarily have $v_1(S_1) = 1$, and this value can at most double at each S_i. Therefore we obtain the bound

$$1 + 2 + 2^2 + \cdots + 2^{2m-2} = 2^{2m-1} - 1 \ . \tag{2}$$

As an example, for $m = 2$, the bound given in (2) is 7. This is almost achieved by the machine M defined by the following program with four rules:

$$(¢, 00) \rightarrow +0,\ (a, 01) \rightarrow 0-,\ (a, 10) \rightarrow ++,\ (a, 11) \rightarrow -+ \ , \tag{3}$$

where we have used the notation $(a, 11) \rightarrow -+$ for the move $(a, 1, 1) \rightarrow (-, +)$, and similarly for others. The computation of M proceeds as follows:

Sign-vector	Entering counter values	Move number
0 0	0 0	0
1 0	1 0	1
1 1	2 1	2
1 1	1 2	3
0 1	0 3	4
0 1	0 2	5
0 1	0 1	6
0 0	0 0	

Therefore M can count up to $n = 6$. However, we can do better than (2). The reason is that in order to be able to double $v_1(S)$, other than the trivial case of $S = S_1$, S has to be recurrent, and it is not possible to have all S_i recurrent if M is 1-reversal. Consider the machine M_m^* whose moves are defined as:

$$\begin{array}{lcl} (¢, 0^m) & \to & +\,0^{m-1} \;, \\ (a, 1^i 0^{m-i}) & \to & +^{i+1}\,0^{m-i-1} \text{ for } 0 < i < m \;, \\ (a, 0^j 1^{m-j}) & \to & 0^j - +^{m-j-1} \text{ for } 0 \le j < m \;. \end{array} \tag{4}$$

The first set of sign-vectors above defined for input a are transient and the second ones are recurrent. The transient sign-vectors accumulate as much as possible in the counters, and the recurrent ones spend as much time as possible while about doubling the maximum counter value. The machine M described in (3) is M_2^*. Another example machine, M_3^*, is given in the Appendix.

Theorem 2. *The machine M_m^* described in (4) is 1-reversal and accepts the singleton $\{a^n\}$ with*

$$n = (m-1)2^m + m \;. \tag{5}$$

Furthermore, this quantity is tight, and the program of any machine that achieves this bound is unique up to relabelling of the counters.

Proof. First we establish the bound in (5). The first m moves of the machine result in the sign-vector 1^m and the counter contents $m, m-1, \ldots, 2, 1$. After m more moves, we arrive at the sign-vector 01^{m-1} with contents of the counters

$$0, 2m-1, 2m-2, \ldots, m+1.$$

At this point the first counter has made a reversal, but all the other counters are still increasing. When first j counters are zeroed, each of them has completed a single reversal and the sign-vector becomes $0^j 1^{m-j}$ for $1 \le j < m$. The largest counter value when we enter this sign-vector is $2^{j-1}m - 2^{j-1} + 1$. When the machine starts to decrement the last counter, its content is $2^{m-1}m - 2^{m-1} + 1$. Therefore the number of moves from 10^{m-1} to the last appearance of $0^{m-1}1$ is

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

We sketch the proof of uniqueness. Since M halts, any recurrent sign-vector must decrease one or more counters. Therefore a recurrent S is followed by some subset of S. Since each such recurrent sign-vector can be used to double the maximum content of the counters whereas a transient one only contributes 1 move, the chain of subsets must be as long as possible. By relabeling if necessary, we can assume that these subsets are $0^j 1^{m-j}$ for $0 \le j < m$. This leaves $m-1$ distinct sign vectors, and the pattern of intervals of 1's that is necessary because M is 1-reversal, forces these to be transient and in the form $1^i 0^{m-i} \to +^{i+1} 0^{m-i-1}$ for $0 \le i < m$. Finally the largest values of the counters when the machine enters the recurrent sign-vector 1^m is when the moves are defined as in M_m^*. □

Next we prove that for 1-reversal machines $m+1$ counters is better than m counters. Here we no longer assume that the language accepted is a singleton (or finite), nor a unary alphabet.

Theorem 3. *Suppose L is accepted by a realtime 1-reversal machine with m counters. Then L is accepted by a realtime 1-reversal machine with $m+1$ counters. Furthermore the containment is strict.*

Proof. Given a realtime 1-reversal machine M with m counters that accepts M, we can view M as an $m+1$ counter machine which behaves exactly like M on the first m counters, and never touches the $(m+1)$-st counter. Since the acceptance of an input string is defined by entering \$ when all counters are zero, this machine is also 1-reversal and accepts L. By theorem 2, the singleton $\{a^n \mid n = m2^{m+1} + m + 1\}$ is accepted by the 1-reversal machine M^*_{m+1}. Since $(m-1)2^m + m < m2^{m+1} + m + 1$ for $m > 0$, this language is not accepted by any 1-reversal realtime machine with m-counters. □

3.2 *k*-Reversal Machines

Now we consider k-reversal machines, $k \geq 1$. The next result gives an upper-bound on the maximal n that is countable by a k-reversal m-counter machine.

Theorem 4. *If the upper bound on n for 1-reversal m-counter machine is $f(m)$, then $f((2k-1)m)$ is an the upper bound on n for a k-reversal m-counter machine.*

Proof. We sketch the proof. Let $L = \{a^n\}$ be a singleton language accepted by a k-reversal m-counter machine M. We will show how we can construct from M a 1-reversal $(2k-1)m$-counter machine M' that makes at least as many steps as M and accepts a language $L' = \{a^{n'}\}$ for some $n' \geq n$. The result then follows. The construction of M' from M is based on the following ideas:

1. Consider first the case $k = 2$. Assume for now that the counters reverse from decreasing to increasing at different times.
2. Let C be a counter in M that makes 2 reversals. We associate with C three counters C, T, C' in M'. Initially, $T = C' = 0$.
3. C in M' simulates C in M as long as C does not decrement. When C decrements, T is set to 1 (i.e., it is incremented). Then as long as C does not increment the simulation continues.
4. When C in M increments, C in M' is decremented while simultaneously incrementing C' until C becomes zero. During the decrementing process all other counters remain unchanged. But to make M' operate in realtime, its input head always reads an a during this process.
5. When the counter C of M' becomes zero, T is set to zero (i,e., it is decremented), and C' is incremented by 1.
6. Then the simulation continues with C' taking the place of C. Counters C and T remain at zero and no longer used.

So if C in M makes 2 reversals, we will need three 1-reversal counters C, T, C' in M'. If C makes 3 reversals, we will need five 1-reversal counters C, T, C', T', C'' in M'. In general, if C makes k reversals, we will need $(2k-1)$ 1-reversal counters in M'. It follows that if there are m counters where each counter makes k reversals, we will need $(2k-1)m$ 1-reversal counters. If some of the counters "reverse" (to increasing) at the same time, we handle them systematically one at a time, by indexing the counters. $\square$

Corollary 1. *If $L = \{a^n\}$ is accepted by a k-reversal m-counter machine, then*

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

We can also prove that the number of counters matters for k-reversal machines.

Theorem 5. *For any fixed k, there is a language accepted by a k-reversal $(m+1)$-counter machine which is not accepted by any k-reversal m-counter machine.*

Proof. Consider $L = \{a^n\}$ where n is the largest number that a k-reversal m-counter machine can count. A bound for n is given in Corollary 1. Suppose M is such a machine and the sequence of sign-vectors in its calculation is

$$0^m \to S_1 \to S_2 \to \cdots \to S_t \to 0^m$$

where $S_i \ne 0^m$ for $1 \le i \le t$. M must have the rule $(a, S_t) \to (e_1, e_2, \dots, e_m)$ where $e_i = -$ for $i \in S_t$ and $e_i = 0$ otherwise. We construct a k-reversal $(m+1)$-counter machine M' that accepts a longer singleton. Define M' by

1. If $(¢, 0^m) \to (e_1, ..., e_m)$ is in M, then $(¢, 0^{m+1}) \to (e_1, ..., e_m, +)$ is in M',
2. If $(a, s_1, ..., s_m) \to (e_1, ..., e_m)$ is in M, then $(a, s_1, ..., s_m, 1) \to (e_1, ..., e_m, +)$ is in M',
3. $(a, 0, ..., 0, 1) \to (0, ..., 0, -)$ is in M'.

Thus with every move of M the counter C_{m+1} is incremented until the last S_t clears its contents, at which point C_{m+1} starts clearing its contents, i.e. makes one reversal. It is clear that M' is k-reversal like M, and it accepts the longer singleton $L = \{a^{2n-1}\}$. $\square$

We can also show that for a fixed m, which may depend on k, $k+1$ reversals are better than k.

Theorem 6. *For any fixed m and $k < 2^{m-1}/m$, there is a language accepted by a $(k+1)$-reversal m-counter machine which is not accepted by any k-reversal m-counter machine.*

Proof. We need the following generalization of Lemma 3 to k-reversal machines. Suppose $S_1, S_2, \dots, S_d$ are the distinct non-null sign-vectors that appear in the computation of a k-reversal machine M with m counters that accepts a singleton. Then $d \le 2km - 1$. To prove this inequality, put $S_0 = S_{d+1} = 0^m$ and as in Lemma 3, consider the $m \times (d+2)$ binary matrix B where the j-th column is S_{j-1} for $1 \le j \le d+2$. Since M is k-reversal, each row of B has at most k intervals consisting of all 1's. Therefore there are a total of at most km horizontal intervals of 1's in B, which together have at most $2km$ endpoints. Since the S_j are distinct, going from S_j to S_{j+1} for $0 \leq j \leq d$ must involve at least one bit change, i.e. at least one of the $2m$ pairs of 01 and 10's. It follows that $d + 1 \leq 2km$.

Since $k < 2^{m-1}/m$, $d < 2^m - 1$. Therefore we can find a non-null set S with $S \neq S_i$ for $1 \leq i \leq d$. Now the longest singleton accepted by a k-reversal m-counter machine M is $\{a^n\}$ where n is as given in Theorem 2. We use S to construct a $(k+1)$-reversal m-counter machine M' which accepts a string longer than n. M' is constructed as follows.

1. $(¢, 0^m) \rightarrow (e_1, ..., e_m)$ is in M', where $e_i = +$ for $i \in S$ and $e_i = 0$ otherwise.
2. If $(¢, 0^m) \rightarrow (f_1, ..., f_m)$ is in M, then $(a, S) \rightarrow (S_1)$ is in M', where S_1 is the sign-vector defined by $i \in S_1$ if $f_i = +$ and $i \notin S_1$ if $f_i = 0$,
3. If $(a, s_1, ..., s_m) \rightarrow (g_1, ..., g_m)$ is in M, with $s_1 \cdots s_m \notin \{0^m, S\}$, then $(a, s_1, ..., s_m) \rightarrow (g_1, ..., g_m)$ is in M'.

Thus M' makes an extra initial move, and then continues as M does. The appearance of S at the beginning of the computation can only introduce one more reversal. Therefore M' is $(k + 1)$-reversal, and accepts $\{a^{n+1}\}$. □

4 Stateless Non-Realtime Multicounter Machines

We briefly consider the case when in each rule, d can be 1 or 0 (i.e., the input head need not move right at each step).

4.1 1-Reversal Machines

Clearly, any language accepted by a realtime machine can be accepted by a non-realtime machine. We can show that the latter is strictly more powerful.

Theorem 7. *The language $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$ can be accepted by a non-realtime 1-reversal 2-counter machine M but not by any realtime k-reversal m-counter machine for any $k, m \geq 1$.*

Proof. M has counters C_1 and C_2. On input $¢w\$$, M reads the input and stores the number of a's (resp., b's) it sees in C_1 (resp., C_2). When the input head reaches \$, the counters are decremented simultaneously while the head remains on \$. M accepts if and only if the counters become zero at the same time.

Suppose L is accepted by some realtime k-reversal m-counter machine M'. Let x be a string with $|x|_a = |x|_b > 0$. Then x is accepted by M', i.e., M' on input $¢x\$$, starts with the input head on ¢ with all counters zero, computes, and accepts after reading the last symbol of x with all counters again at zero.

Consider now giving input xab to M. Then after processing x, all counters are zero. Clearly, after processing symbol a, at least one counter of M' must increment; otherwise (i.e., if all counters remain at zero), M will accept all strings of the form xa^i for all i, a contradiction. Then after processing b, all counters must again be zero, since xab is in L. It follows that on input xab, at least one counter made an additional reversal than on input x. Repeating the argument, we see that for some i, $x(ab)^i$ will require at least one counter to make $k + 1$ reversals. This is a contradiction. □

The result above can be made stronger. A non-realtime reversal-bounded multicounter machine is *restricted* if it can only accept an input when the input head first reaches the right end marker \$ and all counters are zero. Hence, there is no applicable rule when the input head is on \$. However, the machine can be non-realtime (i.e., need not move at each step) when the head is not on \$. The machine can also be *nondeterministic:* Two or more rules can have the same left hand side. Then by the same argument above, we have:

Corollary 2. $L = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b\}$ *cannot be accepted by any restricted nondeterministic non-realtime reversal-bounded multicounter machine.*

Theorem 7 and Corollary 2 show that allowing non-realtime computation (i.e., allowing the machine to remain) on \$ and continue computing makes the machine more powerful.

We note that we can easily construct a realtime unbounded-reversal 2-counter machine M to accept the language above. M uses two counters C, T (both initially zero) and operates as follows when given input $\cent w\$$:

M reads the first symbol of w and increments C. It also sets T to 1 if the symbol is b. We consider two cases: When $T = 0$, M continues reading the input, incrementing C when it sees an a and decrementing C when it sees a b, provided C is not zero. If C is zero and M sees a b, it increments C and sets T to 1. When $T = 1$, M continues reading the input, incrementing C when it sees a b and decrementing C when it sees an a, provided C is not zero. If C is zero and M sees an a, it increments C and sets T to 0.
When \$ is reached and C is zero, M accepts.

References

1. J.E. Hopcroft a nd J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Series in Computer Science. Addison -Wesley, Reading, MA, 1979.
2. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. Assoc. for Computing Machinery*, 25, 116–133, 1978.
3. O. H. Ibarra, J. Karhumaki, and A. Okhotin. On stateless multihead automata: hierarchies and the emptiness problem. *Proceedings of 8th Latin American Symposium*, LNCS 4957, 94–105, 2008.
4. A. J. Korenjak and J. E. Hopcroft. Simple deterministic languages. *Proc. of the 7th Annual Symp. on Foundations of Computer Science*, 36-46, IEEE Computer Society, 1966.
5. M. Kutrib, H. Messerschmidt, and Friedrich Otto. On stateless two-pushdown automata and restarting automata. *Pre-Proceedings of the 8th Automata and Formal Languages*, May, 2008.
6. Gh. Păun; Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1, 2000, 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `www.tucs.fi`).
7. Gh. Păun. *Computing with Membranes: An Introduction*, Springer, Berlin, 2002.
8. L. Yang, Z. Dang, and O.H. Ibarra. On stateless automata and P systems. *Pre-Proceedings of Workshop on Automata for Cellular and Molecular Computing*, August 2007.

Appendix

The computation of machine M_3^* is as follows. It counts up to $n = 19$.

Sign-vector	Entering counter values	Move number
0 0 0	0 0 0	0
1 0 0	1 0 0	1
1 1 0	2 1 0	2
1 1 1	3 2 1	3
1 1 1	2 3 2	4
1 1 1	1 4 3	5
0 1 1	0 5 4	6
0 1 1	0 4 5	7
0 1 1	0 3 6	8
0 1 1	0 2 7	9
0 1 1	0 1 8	10
0 0 1	0 0 9	11
0 0 1	0 0 8	12
0 0 1	0 0 7	13
0 0 1	0 0 6	14
0 0 1	0 0 5	15
0 0 1	0 0 4	16
0 0 1	0 0 3	17
0 0 1	0 0 2	18
0 0 1	0 0 1	19
0 0 0	0 0 0	