A note on realtime one-way synchronized alternating one-counter automata*

Juraj Hromkovič

Department of Computer Science, Comenius University, 842 15 Bratislava, Czechoslovakia, and
Fachbereich 17 Mathematik-Informatik, Universität-GH Paderborn, 4790 Paderborn, Germany

Katsushi Inoue

Department of Computer Science and Systems Engineering, Faculty of Engineering, Yamaguchi
University, Ube, 755, Japan

Communicated by A. Salomaa
Received November 1990

Abstract

Hromkovič, J. and K. Inoue. A note on realtime one-way synchronized alternating one-counter

This paper investigates the power of realtime one-way synchronized alternating one counter
automata (lsaca (1, real)), and shows that (1) lsaca(1, real) is more powerful than real-time
one-way nondeterministic multicounter automata, and (2) there exists a language accepted by
a lsaca(1, real), but not accepted by any realtime one-way alternating multi-stack counter automa-
ta. As a corollary of (2), we have: for each $k \geq 1$, realtime one-way synchronized alternating
$k$-counter ($k$-stack-counter) automata are more powerful than realtime one-way alternating $k$-
counter ($k$-stack-counter) automata. We, finally, show that realtime synchronized alternating finite
automata recognize exactly regular sets, i.e., that one counter is more powerful than no counter for
realtime synchronized alternating automata.

Correspondence to: J. Hromkovič, Department of Computer Science, Comenius University, 842 15
Bratislava, Czechoslovakia, and Fachbereich 17 Mathematik-Informatik, Universität-GH Paderborn, 4790
Paderborn, Germany.

* This work was done during the stay of the first author at Yamaguchi University.
1. Introduction

Synchronized alternation was introduced in [5] as a generalization of the alternation concept from [2] enabling a simple, natural form of communication among parallel processes of alternating devices. Although synchronized alternation is a very new concept, there are already several papers [3, 6–9, 12–15] showing the fruitfulness of this concept.

This paper continues to investigate synchronized alternating devices, especially realtime one-way synchronized alternating one-counter automata. The reason for investigating these devices is that we have already known that very simple types of synchronized alternating devices have a big computational power. For example, it is shown in [3, 7] that two-way synchronized alternating finite automata recognize exactly context-sensitive languages, and this result is still improved by showing that one-way synchronized alternating finite automata do the same [6]. Further, Slobodova [12] has shown that one-way synchronized alternating one blind-counter automata can simulate Turing machines. Thus, one may be interested in investigating the power of one-way synchronized alternating counter automata for some bounds on the time complexity. We are interested in achieving some separation results between synchronized alternation and alternation (nondeterminism) for realtime multicounter machines. The separation results among determinism, non-determinism, and alternation for realtime multicounter machines have been proved in [4, 10].

Section 2 of this paper is devoted to the formal definitions of synchronized alternating devices working in realtime. In Section 3, we show that, for any positive integer \( k \), realtime one-way synchronized alternating one-counter automata (lsaca(1, real)s) are more powerful than realtime one-way nondeterministic \( k \)-counter automata. Section 3 also shows that there exists a language accepted by a lsaca(1, real), but not accepted by any real-time one-way alternating multicounter (multi-stack-counter) automata. From this result, we see that, for each \( k \geq 1 \), realtime one-way synchronized alternating \( k \)-counter (\( k \)-stack-counter) automata are more powerful than realtime one-way alternating \( k \)-counter (\( k \)-stack-counter) automata. Finally, we show that realtime one-way synchronized alternating finite automata (lsafa(real)s) recognize exactly regular sets and, thus, lsaca(1, real)s are more powerful than lsafa(real)s.

2. Definitions

We assume here that the reader is familiar with the alternation concept introduced by Chandra et al. [2], and we refer to [3, 7, 12] for the formal definition of the concept of synchronized alternation. Here, we stress only on the definition of synchronized alternating devices working in realtime. (“Realtime” means that the machine moves its reading head on the input tape to the right in each computation step.) Given a real-time alternating machine type \( M \), we shall augment it by a finite synchronization
alphabet. An internal state of such an augmented (synchronized alternating) machine can be either an internal state of \( M \) or a pair (internal state of \( M \), synchronizing symbol). The latter is called a synchronizing state. As usual, for alternating machines, the states of \( M \) are partitioned into universal, existential, accepting, and rejecting states. We use the usual notation of a configuration and the computation step relation \( \rightarrow_M \) for the machine, and we call the configuration universal, existential, or synchronizing in correspondence to the type of the internal state of this configuration. The initial configuration and the accepting ones are defined as usual for the particular type of the machine.

In order to avoid misunderstandings, we give a precise definition of accepting computations of a realtime synchronized alternating machine \( M \). We assume that the endmarker \( S \) is attached to the right of the input tape of \( M \), and \( M \) makes at most one step on \$ without moving its reading head.

**Definition 2.1.** The full configuration tree of a realtime synchronized alternating machine RSAM \( M \) on an input word \( w \) is a finite labelled tree \( T_M(w) \) such that

(i) each node \( v \) of \( T_M(w) \) is labelled by some configuration \( c(v) \) of \( M \),
(ii) for the root \( v_0 \), \( c(v_0) \) is the initial configuration of \( M \) on \( w \), and
(iii) node \( v_2 \) is a direct descendant of node \( v_1 \) iff \( c(v_1) \rightarrow_M c(v_2) \).

**Observation 2.2.** The full configuration tree \( T_M(w) \) can be divided into at most \( n + 2 \) levels \( L_0, L_1, \ldots, L_n, L_{n+1} \) (where \( n \) is the length of \( w \)) according to the distance to the root, and each configuration in the \( i \)th level \( L_i \) has positioned its input head on the \((i + 1)\)st symbol (from the left) of \( w \), where the \( i \)th symbol of \( w \) is the right endmarker \$ for \( i = n + 1, n + 2 \).

**Definition 2.3.** A computation tree of an RSAM \( M \) on an input word \( w \) is a subtree \( T' \) of the full configuration tree \( T_M(w) \) of \( M \) on \( w \) such that

(i) each node in \( T' \) labelled by a universal configuration has the same direct descendants as in \( T_M(w) \),
(ii) each node \( v \) in \( T' \) labelled by an existential configuration has exactly one descendant if \( v \) has at least one descendant in \( T_M(w) \), and no descendant if \( v \) has no descendant in \( T_M(w) \), and
(iii) for each level \( L_i \) of the nodes of \( T' \) with the distance \( i - 1 \) to the root of \( T' \), the following holds: for any nodes \( v_1 \) and \( v_2 \) in \( L_i \), if \( c(v_1) \) is a synchronizing configuration with a synchronizing symbol \( S \), then \( c(v_2) \) is also a synchronizing configuration with the synchronizing symbol \( S \).

**Definition 2.4.** An accepting computation of an RSAM \( M \) on an input word \( w \) is a computation tree of \( M \) on \( w \) whose leaves are all labelled by accepting configurations. The word \( w \) is accepted by \( M \) if there exists an accepting computation of \( M \) on \( w \). We denote the set of all the words accepted by \( M \) by \( L(M) \).
We refer to [4] for definitions of a realtime one-way non-deterministic \( k \)-counter automaton, denoted by \( \lnca(k, \text{real}) \), to [1] for definitions of realtime one-way nondeterministic \( k \)-stack-counter automaton, denoted by \( \lnsca(k, \text{real}) \), to [10] for definitions of realtime one-way alternating \( k \)-counter automaton, denoted by \( \lac(k, \text{real}) \), to [11] for definitions of realtime one-way alternating \( k \)-stack-counter automaton, denoted by \( \lasca(k, \text{real}) \). By \( \lsaca(k, \text{real}) \) (\( \lsasca(k, \text{real}) \)) we denote the synchronized version of \( \lac(k, \text{real}) \) (\( \lasca(k, \text{real}) \)). Thus, for example, a realtime one-way synchronized alternating one-counter automaton is denoted by \( \lsaca(1, \text{real}) \). Further, we denote by \( \lsafa(\text{real}) \) a realtime one-way synchronized alternating finite automaton. By \( \lnca(k, \text{real}) \) (\( \lnsca(k, \text{real}) \), \( \lac(k, \text{real}) \), \( \lasca(k, \text{real}) \), \( \lsaca(k, \text{real}) \), \( \lsasca(k, \text{real}) \), \( \lsafa(\text{real}) \)) we denote the class of sets accepted by \( \lnca(k, \text{real}) \) (\( \lnsca(k, \text{real}) \), \( \lac(k, \text{real}) \), \( \lasca(k, \text{real}) \), \( \lsaca(k, \text{real}) \), \( \lsasca(k, \text{real}) \), \( \lsafa(\text{real}) \)).

3. Results

We first show that \( \lsaca(1, \text{real}) \)s can simulate realtime one-way nondeterministic multicounter automata.

**Theorem 3.1.** For each \( k \geq 1 \), \( \lnca(k, \text{real}) \) \( \subseteq \) \( \lsaca(1, \text{real}) \).

**Proof.** Let \( A \) be a \( \lnca(k, \text{real}) \) accepting a language \( L(A) \) over an alphabet \( \Sigma \). Without loss of generality, we may assume that \( A \) makes its first computation step deterministically without changing the contents of any of its counters for each input word. Let \( Q_A \) be the set of states of \( A \) with the initial state \( q_0 \). Let

\[
\delta_A \subseteq (Q_A \times \{0, 1\}^k \times \Sigma) \times (Q_A \times \{0, 1\}^k \times \{\text{right}\}) \\
\cup (Q_A \times \{0, 1\}^k \times \{\text{S}\}) \times (Q_A \times \{0, 1\}^k \times \{\text{no move}\})
\]

be the transition function of \( A \), and let \( b_{\text{max}} = \max_{v \in D} \{\text{card}((u \in Q_A \times \{0, 1\}^k \times \{\text{right, no move}\} | (v, u) \in \delta_A))\} \), where \( D = Q_A \times \{0, 1\}^k \times (\Sigma \cup \{\text{S}\}) \), be the upper bound on the number of nondeterministic branches (possible actions) from any input argument \( v \in D \) of \( A \). Let us also assume that for each input argument \( v \in D \) all the possible actions (which belong to \( Q_A \times \{0, 1\}^k \times \{\text{right, no move}\} \)) from \( v \) are sequentially ordered, i.e., we can assign to each such action its order number.

Now, let us construct a \( \lsaca(1, \text{real}) \) \( B \) simulating \( A \). Let the finite set of synchronizing symbols of \( B \) be \( S_B = \{(a_1, a_2, \ldots, a_k, d) | \forall i(1 \leq i \leq k) [a_i \in \{0, 1\}] \) and \( d \in \{1, \ldots, b_{\text{max}}\} \} \). Let

\[
Q_B = Q_A \times \{1, 2, \ldots, k\} \cup Q_A \times \{1, 2, \ldots, k\} \times S_B,
\]

be the set of states of \( B \), and let the only one universal state of \( B \) be the initial state \((q_0, 1)\). The idea of the simulation of \( A \) by \( B \) is as follows. Assume that after reading the
Realtime one-counter automata

first input symbol $A$ goes deterministically from the initial state $q_0$ to a state $p$ without changing the contents of any of its counters. $B$ simulates this step by a universal branching of $B$ into $k$ processes $B_1, B_2, \ldots, B_k$, where $B_i$ is in the state $(p, i)$ for each $i \in \{1, \ldots, k\}$. The idea of the simulation consists in simulating the contents of the $i$th counter of $A$ by the contents of the counter of $B_i$ during the whole computation. The processes $B_1, B_2, \ldots, B_k$ will communicate by synchronization in such a way that each of these processes working in parallel will be able to correctly guess the whole actual argument $v \in D$ of $A$ in each computation step.

Let us assume that after the $(j-1)$st computation step $A$ read the $j$th symbol "a" on the input tape in a state $q$, and whether its counters are empty or not be described by the vector $(b_1, \ldots, b_k) \in \{0, 1\}^k$. Further, assume that for each $i \in \{1, \ldots, k\}$, $B_i$ reads the $j$th input symbol "a" in a state $(q, i, s)$, and the contents of its counter be the same as the contents of the $i$th counter of $A$ which is characterized by $b_i \in \{0, 1\}$. Now, each $B_i$ existentially chooses one of at most $2^{k-1} \times b_{\text{max}}$ possible actions, each corresponding to one element from the following subset, $S^i$, of $S$: $S^i = \{(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_k, r) \mid \forall t (1 \leq t \leq k, t \neq i) [a_t \in \{0, 1\}] \text{ and } (r \in \{1, \ldots, b_{\text{max}}\}]$. If $B_i$ chooses one action in which $B_i$ takes the synchronizing symbol $(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_k, r) \in S^i$, then $B_i$ follows the $r$th nondeterministic choice of $A (\delta_A)$ for the argument $u = (q, a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_k, a)$, i.e., $B_i$ changes the state $(q, i, s)$ into the state $(p, i, s')$, and it changes the contents of its counter by $c_i \in \{-1, 0, 1\}$, if $(p, c_1, \ldots, c_k, \text{right})$ is the $r$th possible action of $A$ for the argument $u$.

Now, let us show that $B$ simulates $A$ correctly. From the definition of the computation tree of an RSAM and from the fact that $B$ has only one universal branching into $B_1, \ldots, B_k$, we see that each computation tree of $B$ consists exactly of $k$ disjoint paths leading from the root of the full configuration tree of $B$ to some leaves, and each path corresponds to one of the nondeterministically chosen computations of some $B_i$. Since all the vertices of the $m$th level of the full configuration tree are labelled by synchronizing configurations for $m \geq 3$, each level of a computation tree of $B$ has to be labelled by synchronizing configurations with the same synchronizing symbol. Thus, the only synchronizing symbols which can be the same for all the $k$ vertices in the $j$th level labelled by the actual configurations of the processes $B_1, B_2, \ldots, B_k$ are synchronizing symbols in $\bigcap_{1 \leq i \leq k} S^i = \{(b_1, \ldots, b_k, r) \mid r \in \{1, \ldots, b_{\text{max}}\}]$. It implies that only correct guessings $(b_1, \ldots, b_k)$ of the actual argument of $A$ can appear in any computation tree of $B$. On the other hand, $B$ is able to simulate any of the possible nondeterministic choices of $A$ in the $j$th step by using the synchronizing symbol $(b_1, \ldots, b_k, r)$ for the $r$th possible choice. Since no two vertices in the same level of an computation tree of $B$ may be labelled by two different synchronizing symbols $(b_1, \ldots, b_k, u)$ and $(b_1, \ldots, b_k, v)$ for $u \neq v$, it is clear that all processes $B_1, B_2, \ldots, B_k$ in a computation tree of $B$ simulate the same nondeterministic decision of $A$ made in the $j$th step. Clearly, in this way we have exactly one computation tree of $B$ for each one of the computation paths of $A$. So, there is an accepting computation of $A$ on the given input $w$ iff there is an accepting computation of $B$ on $w$. □
We next show that there exists a language accepted by a \( \text{lsaca}(1, \text{real}) \), but not accepted by any realtime one-way alternating multi-stack-counter automaton.

**Theorem 3.2.** There exists a language which is in \( \text{1SACA}(1, \text{real}) \) but not in \( \bigcup_{1 \leq k < \infty} \text{1ASCA}(k, \text{real}) \).

**Proof.** Let \( L = \{ w \# w_1 \# w_2 \ldots \# w_r \mid w \in \{0, 1\}^+ \} \) with \( r \geq 1 \), \( \forall i (1 \leq i \leq r) [w_i \in \{0, 1\}^+] \) and \( \exists j (1 \leq j \leq r) [w_j = w] \). It is shown in [11] that \( L \) is not in \( \bigcup_{1 \leq k < \infty} \text{1ASCA}(k, \text{real}) \).

On the other hand, \( L \) is accepted by a \( \text{1SACA}(1, \text{real}) \) \( M \) which acts as follows. Suppose that an input word \( x = w \# w_1 \# w_2 \ldots \# w_r \) (where \( w \in \{0, 1\}^+ \), \( r \geq 1 \), and \( w_i \in \{0, 1\}^+ (1 \leq i \leq r) \)) is presented to \( M \). (Input words in different form from the above can easily be rejected by \( M \).) Let the length of the initial segment \( w \) of \( x \) be \( n \). While reading the initial segment \( w \), \( M \) generates \( n \) processes \( M_1, M_2, \ldots, M_n \) by using universal branches in such a way that for each \( i (1 \leq i \leq n) \), the \( i \)th process \( M_i \), picks up the \( i \)th symbol \( w(i) \) of \( w \) and stores \( i \) in its counter just after its input head read through \( w \). Then each process \( M_i \) existentially chooses some \( j_i (1 \leq j_i \leq r) \), enters a synchronizing state on the first symbol of the subword \( w_{j_i} \) of \( x \), moves into \( w_{j_i} \), checks by using the symbol \( w(i) \) picked up and the contents \( i \) of its counter if the \( i \)th symbols of \( w \) and \( w_{j_i} \) are the same, and enters an accepting state only if this check is successful. It will be obvious that \( L = L(M) \). (Note that since \( M \) is a realtime machine, all the processes \( M_1, M_2, \ldots, M_n \) must enter the synchronizing states with the same synchronizing symbol on the same position on the input \( x \) (i.e., \( j_1 = j_2 = \cdots = j_n \)), if \( M \) accepts \( x \).)

It is shown in [1] that \( \text{1NSCA}(k, \text{real}) \subseteq \text{1NCA}(2k, \text{real}) \) for each positive integer \( k \). From this fact and from Theorems 3.1 and 3.2, we have Corollary 3.3.

**Corollary 3.3.**

1. \( \bigcup_{1 \leq k < \infty} \text{1NCA}(k, \text{real}) = \bigcup_{1 \leq k < \infty} \text{1NSCA}(k, \text{real}) \neq \text{1SACA}(1, \text{real}) \).

2. For each \( k \geq 1 \), \( \text{1ACA}(k, \text{real}) \neq \text{1ASCA}(k, \text{real}) \) and \( \text{1ASCA}(k, \text{real}) \neq \text{1SASCA}(k, \text{real}) \).

It is shown in [10, 11] that \( \text{1ACA}(k, \text{real}) \neq \text{1ACA}(k+1, \text{real}) \) and \( \text{1ASCA}(k, \text{real}) \neq \text{1ASCA}(k+1, \text{real}) \) for each \( k \geq 1 \). Unfortunately, it is unknown whether or not \( \text{1SACA}(k, \text{real}) \neq \text{1SACA}(k+1, \text{real}) \) and \( \text{1SASCA}(k, \text{real}) \neq \text{1SASCA}(k+1, \text{real}) \) for each \( k \geq 1 \). It is also unknown whether \( \text{1ACA}(k+1, \text{real}) - \text{1SACA}(k, \text{real}) \neq \emptyset \) and \( \text{1ASCA}(k+1, \text{real}) - \text{1SASCA}(k, \text{real}) \neq \emptyset \) for each \( k \geq 1 \), i.e., whether one additional counter for \( \text{1aca} \) (\( \text{1saca} \)) can bring more computational power than synchronized alternation.

Now, we are only able to show that one counter is better than no counter for realtime synchronized alternating automata, as the following theorem shows.

**Theorem 3.4.** \( \text{1SAFA}(\text{real}) = \mathcal{R} \), where \( \mathcal{R} \) denotes the class of all regular sets.

**Proof.** Obviously, \( \mathcal{R} \subseteq \text{1SAFA}(\text{real}) \). Now, let us show that any \( \text{1safa}(\text{real}) \) \( A = (\Sigma, K, \delta_A, F, q_0) \) can be simulated by a nondeterministic finite automaton \( B = (\Sigma, P(K) \cup \{Q\}, \)
δ_B, P(F), {q_0}), where P(K) (P(F)) is the power set of the set K (F) of states (final states) of A, and Q is a new state. For each b ∈ Σ and for each Z ∈ P(K) consisting either only of nonsynchronizing states or only of synchronizing states with the same synchronizing symbol, let δ_B(Z, b) = \{ U \mid \delta_A(q, b) \subseteq U \} for each universal state q ∈ Z, and exactly one state r_p ∈ δ_A(q, b) is in U for each existential state p ∈ Z. Let δ_B(Z, b) = \{ Q \} for each b ∈ Σ and for each Z containing two synchronizing states from K with different synchronizing symbols. One can easily see that B simulates A by going from one level of a computation tree t of A (represented by the set of all states appearing in the configurations of this level) to the next level of t. The correctness of this simulation procedure follows from the fact that if a level of a computation tree of A contains several times the same configuration c (i.e., the same state) assigned to different nodes, then it is sufficient to simulate only the computation continuing from any one of these nodes labelled by c (if one finds a suitable computation subtree rooted by one of these nodes, then this computation subtree can be used for any other node labelled by c).

The last open problem formulated in this note follows from the proof of Theorem 3.4. The nondeterministic finite automaton B simulating the 1safa(real) A has exponential number of states in comparison with A. Does there exist a more effective simulation procedure of 1safa(real) or does there exist a language requiring the exponential increase of the number of states of the simulating nondeterministic finite automaton? How about the simulation of 1safa(real)s by deterministic finite automata?

Finally, we note that by allowing the linear time to 1safa’s they recognize the context-sensitive language \{ wcw \mid w \in \{0, 1\}^* \}. Thus, we conjecture that a 1safa(real) is the only one “natural” synchronized alternating device whose power is restricted to the recognition of regular sets only.

References