ELSEVIER

Theoretical Computer Science 289 (2002) 191–204

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Augmenting the discrete timed automaton with other data structures ☆

Oscar H. Ibarra *, Jianwen Su

*Department of Computer Science, University of California, Santa Barbara, CA 93106, USA*

Received July 2000; accepted May 2001
Communicated by A. Salomaa

**Abstract**

We describe a general automata-theoretic approach for analyzing the verification problems of discrete timed automata (i.e., timed automata with integer-valued clocks) augmented with various data structures. Formally, let $\mathscr{C}$ be a class of nondeterministic machines with reversal-bounded counters and possibly other data structures (e.g., a pushdown stack, a queue, a read-write work-tape, etc.). Let $A$ be a discrete timed automaton and $M$ be a machine in $\mathscr{C}$. Denote by $A \oplus M$ the combined automaton, i.e., $A$ augmented with $M$ (in some precise sense to be defined). We show that if $\mathscr{C}$ has a decidable emptiness problem, then the (binary, forward, backward) reachability, safety, and invariance for $A \oplus M$ are solvable. We give examples of such $\mathscr{C}$'s and exhibit some new properties of discrete timed automata that can be verified. We also briefly consider reachability in discrete timed automata operating in parallel. 

## 1. Introduction

Ever since the introduction of the model of a timed automaton [1], there have been many studies that extend the expressive power of the model (e.g. [2, 4, 5, 7, 10]). For instance [2] considers models of hybrid systems of finite automata supplied with (unbounded) discrete data structures and continuous variables and obtains decidability results for several classes of systems with control variables and observation variables. Comon and Jurski [5, 4] shows that the binary reachability of timed automata is expressible in the additive theory of the reals. Dang et al. [7] characterizes the binary reachability of discrete timed automata (i.e., timed automata with integer-valued clocks) augmented with a pushdown stack, while Ibarra et al. [10] looks at queue-connected discrete timed automata.

☆ Supported in part by NSF grants IRI-9700370 and IIS-9817432.
* Tel.: +1-805-893-4171; fax: +1-805-893-8553.
*E-mail addresses:* ibarra@cs.ucsb.edu (O.H. Ibarra), su@cs.ucsb.edu (J. Su).

0304-3975/02/$ - see front matter 
PII: S0304-3975(01)00269-9

In this paper, we extend the ideas in [7, 10] and describe a general automata-theoretic approach for analyzing the verification problems of discrete timed automata augmented with various data structures. Formally, let $\mathscr{C}$ be a class of nondeterministic machines with reversal-bounded counters (i.e., each counter can be incremented or decremented by 1 and tested for zero, but the number of alternations between nondecreasing mode and nonincreasing mode is bounded by a constant, independent of the computation) and possibly other data structures, e.g., a pushdown stack, a queue, a read-write worktape, etc. Let $A$ be a discrete timed automaton and $M$ be a machine in $\mathscr{C}$. Denote by $A \oplus M$ the combined automaton, i.e., $A$ augmented with $M$ (in some precise sense to be defined). We show that if $\mathscr{C}$ has a decidable emptiness problem, then the (binary, forward, backward) reachability, safety, and invariance for $A \oplus M$ are also solvable. We give examples of such $\mathscr{C}$'s and exhibit some new properties of discrete timed automata that can be verified:

1. For example, let $A$ be a discrete timed automaton with $k$ clocks. For a given computation of $A$, let $r_i$ be the number of times clock $i$ resets, $i = 1, \ldots, k$. Suppose we are interested in computations of $A$ in which the $r_i$'s satisfy a Presburger formula $f$, i.e., we are interested in the set $Q$ of pairs of configurations $(\alpha, \beta)$ such that $\alpha$ can reach $\beta$ in a computation in which the clock resets satisfy $f$. (A configuration of $A$ is a pair $(q, U)$, where $q$ is a state and $U$ is the set of clock values.) We can show that $Q$ is Presburger. One can also put other constraints, like introducing a parameter $t_i$ for each clock $i$, and consider computations where the first time $i$ resets to zero is before (or after) time $t_i$. Then $Q(t_1, \ldots, t_k)$ is Presburger.
2. As another example, suppose we are interested in the set $S$ of pairs of configurations $(\alpha, \beta)$ of a discrete timed automaton $A$ such that there is a computation path (i.e., sequence of states) from $\alpha$ to $\beta$ that satisfies a property that can be verified by a machine in a class $\mathscr{C}$. If $\mathscr{C}$ has a decidable emptiness problem, then $S$ is effectively computable. For example, suppose that the property is for the path to contain three nonoverlapping subpaths (i.e., segments of computation) which go through the same sequence of states, and the length of the subpath is no less than $\frac{1}{5}$ of the length of the entire path. We can show that $S$ is computable.

The constraints in 1 and 2 can be combined; thus, we can show that the set of pairs of configurations that are in both $Q$ and $S$ is computable.

3. We can equip the discrete timed automaton with one-way write-only tapes which the automaton can use to record certain information about the computation of the system (and perhaps even require that the strings appearing in these tapes satisfy some properties). Such systems can effectively be analyzed.

Finally, we briefly look at reachability in machines (i.e., $A_1 \oplus M_1$ and $A_2 \oplus M_2$) operating in parallel.

## 2. Combining discrete timed automata with other machines

A timed automaton [1] is a finite-state machine augmented with finitely many real-valued clocks. All the clocks progress synchronously with rate 1, except that a clock

can be reset to 0 at some transition. Here, we only consider integer-valued clocks. A *clock constraint* is a Boolean combination of *atomic clock constraints* in the following form: $x \# c$, $x - y \# c$, where $\#$ denotes $\leqslant, \geqslant, <, >$, or $=$, $c$ is an integer, $x, y$ are integer-valued clocks. Let $\mathscr{L}_X$ be the set of all clock constraints on clocks $X$. Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ the set of nonnegative integers. Formally, a *discrete timed automaton* $A$ is a tuple $\langle S, X, E \rangle$ where

1. $S$ is a finite set of (*control*) *states*,
2. $X$ is a finite set of *clocks* with values in $\mathbb{N}$, and
3. $E \subseteq S \times 2^X \times \mathscr{L}_X \times S$ is a finite set of *edges* or *transitions*.

Each edge $\langle s, \lambda, l, s' \rangle$ in $E$ denotes a transition from state $s$ to state $s'$ with *enabling condition* $l \in \mathscr{L}_X$ and a set of clock resets $\lambda \subseteq X$. Note that $\lambda$ may be empty. The meaning of a one-step transition along an edge $\langle s, \lambda, l, s' \rangle$ is as follows:

- The state changes from $s$ to $s'$.
- Each clock changes. If there are no clock resets on the edge, i.e., $\lambda = \emptyset$, then each clock $x \in X$ progresses by one time unit. If $\lambda \neq \emptyset$, then each clock $x \in \lambda$ is reset to 0 while each $x \notin \lambda$ remains unchanged.
- The enabling condition $l$ is satisfied.

The notion of a discrete timed automaton defined above is slightly different, but easily shown equivalent to the standard definition of a (discrete) timed automaton in [1] (see [7]).

Now consider a class $\mathscr{C}$ of acceptors, where each machine $M$ in the class is a nondeterministic finite automaton augmented with finitely many counters, and possibly other data structures. Thus, $M = \langle Q, \Sigma, q_0, F, K, D, \delta \rangle$, where $Q$ is the state set, $\Sigma$ is the input alphabet, $q_0$ is the start state, $F$ is the set of accepting states, $K$ is the set of counters, $D$ the other data structures, and $\delta$ is the transition function. In the move $\delta(q, a, s_1, \ldots, s_k, loc) = \{t_1, \ldots, t_m\}$,

- $q$ is the state, $a$ is $\varepsilon$ or a symbol in $\Sigma$, $s_i$ is the status of counter $i$ (i.e., zero or non-zero), and $loc$ is the "local" portion of the data structure(s) $D$ that influences (affects) the move. For example, if $D$ is a pushdown stack, then $loc$ is the top of the stack; if $D$ is a two-way read-write tape, then $loc$ is the symbol under the read-write head; if $D$ is a queue, then $loc$ is the symbol in the front of the queue or $\varepsilon$ if the queue is empty. Note that $D$ can be a combination of several data structures (e.g., several stacks and queues).
- $t_1, \ldots, t_m$ are the choices of moves (note that $M$ is nondeterministic). Each $t_i$ is of the form $(p, d_1, \ldots, d_k, act)$, which means increment counter $i$ by $d_i$ (1, 0, or $-1$), perform $act$ on $loc$, and enter state $p$. For example if $D$ is a pushdown stack, $act$ pops the top symbol and pushes a string (possibly empty) onto the stack; if $D$ is a two-way read-write tape, $act$ rewrites the symbol under the read-write head and moves the head one cell to the left, right, or remains on the current cell; if $D$ is a queue, then $act$ deletes $loc$ (if not $\varepsilon$) from the front of the queue, and possibly adds a symbol to the rear of the queue.

Note that the counters can only hold nonnegative integers. There is no loss of generality since the states can remember the signs.

The language accepted by $M$ is denoted by $L(M)$. We will only be interested in $\mathscr{C}$'s with a decidable emptiness problem. This is the problem of deciding for a given acceptor in $\mathscr{C}$, whether $L(M)$ is empty. Since the emptiness problem for finite automata augmented with two counters is undecidable [11], we will need to put some restrictions on the operation of the counters.

Let $r$ be a nonnegative integer. We say that a counter is $r$-reversal if the counter changes mode from nondecreasing to nonincreasing and vice-versa at most $r$ times, independent of the computation. So, for example, a counter whose values change according to the pattern

0 1 1 2 3 3 3 4 5 5 5 4 3 2 1 1 0 0 1 1 2 3 3

is 2-reversal. When we say that the counters are *reversal-bounded*, we mean that we are given an integer $r$ such that each counter is $r$-reversal. From now on, we will assume that the acceptors in $\mathscr{C}$ have reversal-bounded counters.

We can extend the acceptors in $\mathscr{C}$ to *multitape* acceptors by providing them with multiple one-way read-only input tapes. Thus, a $k$-tape acceptor now accepts a $k$-tuple of words (strings). We call the resulting class of acceptors $\mathscr{C}(k)$. The emptiness problem for $\mathscr{C}(k)$ is deciding for a given $k$-tape acceptor $M$, whether it accepts an empty set of $k$-tuples of strings. We denote $\mathscr{C}(1)$ simply by $\mathscr{C}$. One can easily show the following:

**Theorem 1.** *If the emptiness problem for $\mathscr{C}$ is decidable, then the emptiness problem for $\mathscr{C}(k)$ is decidable.*

**Proof.** We give a proof for the case $k=2$. Let $M$ be a 2-tape acceptor in $\mathscr{C}(2)$. We may assume without loss of generality that the two tapes of $M$ use disjoint input alphabets. We construct an acceptor $M'$ in $\mathscr{C}$ such that $L(M')$ is empty if and only if $L(M)$ is empty. The idea of the construction is as follows: If $(x_1,x_2)$ is an input to $M$, then the input to $M'$ is a string $x$ which is some interlacing of the symbols in $x_1$ and $x_2$ (i.e., $x$ is a shuffle of $x_1$ and $x_2$). Thus $x$ with the symbols in $x_1$ ($x_2$) deleted reduces to $x_2$ ($x_1$). Clearly $M'$ can simulate the actions of the two input heads of $M$ on input $x$. □

In the rest of the paper, we will assume that $\mathscr{C}$ has a decidable emptiness problem. In the area of verification, we are mostly interested in the "behavior" of machines rather than their language-accepting capabilities. When dealing with machines in $\mathscr{C}$ *without* inputs, we shall refer to them simply as machines. Thus, when we say "a machine $M$ in $\mathscr{C}$", we mean that $M$ has *no* input tape.

Let $A$ be a discrete timed automaton and $M$ a machine in class $\mathscr{C}$ (hence, $M$ has no input tape!). Let $A \oplus M$ be the machine obtained by augmenting $A$ with $M$. So, e.g., if $M$ is a machine with a pushdown stack and reversal-bounded counters, then $A \oplus M$ will be a discrete pushdown timed automaton with reversal-bounded counters. We will describe more precisely how $A \oplus M$ operates later. A configuration $\alpha$ of $A \oplus M$ is a

5-tuple $(s, U, q, V, v(D))$, where $s$ and $U$ are the state and clock values of $A$, and $q$, $V$, $v(D)$ are the state, counter values, and data structure values of $M$ (e.g., if $D$ is a pushdown stack, then $v(D)$ is the content of the stack; if $D$ is a queue, then $v(D)$ is the content of the queue). Let $Reach(A \oplus M)$ be the set of all pairs of configurations $(\alpha, \beta)$ such that $\alpha$ can reach $\beta$. This set is the binary reachability of $A \oplus M$. We assume that the configurations are represented as strings over some alphabet, where the components of a configuration are separated by markers, and the clock and counter values represented in unary. We also assume that each of the following tasks can be implemented on a machine $M'$ in $\mathscr{C}$:

1. $M'$, when given a configuration $\alpha = (s, U, q, V, v(D))$ of $A \oplus M$ on its input tape, can represent this configuration in its counters and data structures, i.e., $M'$ can read $\alpha$ and record the states $s$ and $q$, store the set of values of $U$ and $V$ in appropriate counters, and store $v(D)$ in its data structures.
2. $M'$, when given a configuration $\alpha$ on its input tape, can check if $\alpha$ represents its current configuration (this task is the converse of 1).

In the following, $A$ is a discrete timed automaton and $M$ is a machine in $\mathscr{C}$; FCA refers to a nondeterministic finite automaton (acceptor) augmented with reversal-bounded counters.

**Theorem 2.** *We can effectively construct a* 2*-tape acceptor in* $\mathscr{C}(2)$ *accepting Reach* $(A \oplus M)$.

Note that the input to the 2-tape acceptor is a pair of configurations $(\alpha, \beta)$, where $\alpha$ ($\beta$) is on the first (second) tape. We illustrate the proof in the next section for a particular class $\mathscr{C}$.

**Theorem 3.** *If* $I$ (*the initial set*) *and* $P$ (*the unsafe set*) *are two sets of configurations of* $A \oplus M$, *let BAD be the set of all configurations in* $I$ *that can reach configurations in* $P$. *If* $I$ *and* $P$ *can be accepted by FCAs, then we can effectively construct an acceptor in* $\mathscr{C}$ *accepting BAD. Hence, nonsafety is decidable with respect to* $P$.

**Proof.** Let $M_I$ and $M_P$ be FCAs accepting $I$ and $P$, respectively. From Theorem 2, we can construct a 2-tape acceptor $B$ in $\mathscr{C}(2)$ accepting $Reach(A \oplus M)$. By using additional counters, we can modify $B$ to a 2-tape acceptor $B'$ which also checks that $\alpha$ ($\beta$) on tape1 (tape2) is accepted by $M_I$ ($M_P$). Now construct from $B'$ an acceptor $B''$ which deletes the second tape. Clearly, $L(B'') = BAD$. Then $A \oplus M$ is unsafe if and only if $L(B'')$ is nonempty, which is decidable by Theorem 1. □

Since the complement of a language accepted by a deterministic FCA can also be accepted by an FCA [8], we also have:

**Theorem 4.** *If* $I$ *and* $P$ (*the safe set*) *are two sets of configurations of* $A \oplus M$, *let GOOD be the set of all configurations in* $I$ *that can only reach configurations in* $P$.

*If $I$ can be accepted by an FCA and $P$ can be accepted by a deterministic FCA, then we can decide whether $GOOD = I$. Hence, invariance is decidable with respect to $P$.*

We can show that forward reachability is computable.

**Theorem 5.** *Let $I$ be a set of configurations accepted by an FCA. We can effectively construct an acceptor in $\mathscr{C}$ accepting $post^*(A \oplus M, I) =$ the set of all configurations reachable from configurations in $I$.*

**Proof.** Let $M_I$ be an FCA accepting $I$. As in Theorem 3, we can construct a 2-tape acceptor $B'$ in $\mathscr{C}(2)$ accepting the set of all pairs of configurations $(\alpha, \beta)$ in $Reach(M)$ such that $\alpha$ is accepted by $M_I$. We can then construct from $B'$ an acceptor $B''$ in $\mathscr{C}$ which deletes the first tape, and $L(B'') = post^*(A \oplus M, I)$. □

Similarly, for backward reachability we have:

**Theorem 6.** *Let $I$ be a set of configurations accepted by an FCA. We can effectively construct an acceptor in $\mathscr{C}$ accepting $pre^*(A \oplus M, I) =$ the set of all configurations that can reach configurations in $I$.*

We can equip $A \oplus M$ with a one-way input tape. In order to do this, we can simply change the format of the transition edge of $A$ by a 5-tuple $\langle s, \lambda, l, s', a \rangle$ in $E$, where $a$ denotes an input symbol or $\varepsilon$ (the null string). The meaning of this edge is like before, but now $A$ can read a symbol or a null string at each transition. We also define a subset of the states of $A$ as accepting states. Then $A \oplus M$ becomes an acceptor. Note that $A$ and $M$ will now start on some prescribed initial configurations (e.g., $A$ is initialized to its start state with all clocks zero, $M$ is initialized to its start state with all counters zero and the other data structures properly initialized). We will prove the following in the next section.

**Theorem 7.** *It is decidable to determine, given an acceptor $A \oplus M$, whether $A \oplus M$ accepts the empty set.*

One can extend the $A \oplus M$ acceptor to have multiple input tapes. Then, similar to Theorem 1, we have:

**Corollary 1.** *It is decidable to determine, given a multitape acceptor $A \oplus M$, whether $A \oplus M$ accepts the empty set.*

We can also equip the multitape $A \oplus M$ acceptor with one-way output tapes. But, clearly, these output tapes can also be viewed as input tapes (since writing can be simulated by reading). Hence, the analysis of a multi-input-tape multi-output-tape $A \oplus M$ reduces to the analysis of multi-input-tape $A \oplus M$.

## 3. Examples of $\mathscr{C}$

We illustrate the proof of Theorem 2 for the class $\mathscr{C}$, where each machine is a nondeterministic machine with a pushdown stack and finitely many reversal-bounded counters. Call a machine in this class a PCM, and PCA when it has an input tape (i.e., it is an acceptor). It is known that the emptiness problem for PCAs is decidable [8]. Let $A$ be a discrete timed automaton and $M$ be a PCM. We describe precisely how $A \oplus M$ operates.

A configuration of the timed automaton $A$ is of the form $(s, U)$, where $s$ is the state and $U$ is the set of clock values. Now machine $M$ has states, pushdown stack, and reversal-bounded counters. A move of $M$ is defined by a transition function $\delta$. If $\delta(q, Z, s_1, \ldots, s_k) = \{t_1, \ldots, t_m\}$, then

- $q$ is the state, $Z$ is the topmost symbol, and $s_i$ is the status of counter $i$ (i.e., zero or non-zero).
- $t_1, \ldots, t_m$ are the choices of moves (note that $M$ is nondeterministic). Each $t_i$ is of the form $(p, w, d_1, \ldots, d_k)$, which means pop $Z$ and push string $w$ (which is possibly empty) onto the stack, increment counter $i$ by $d_i$ (1, 0, or $-1$), and enter state $p$.

A configuration of $M$ can be represented by a tuple of the form $(q, V, w)$, where $q$ is a state, $V$ is the set of values of the counters, and $w$ is the content of the stack with the rightmost symbol at the top of the stack.

A transition of the combined machine $A \oplus M$ is now a tuple $\langle s, \lambda, l, s', ENTER(M, R)\rangle$, where $\langle s, \lambda, l, s'\rangle$ is as in a timed automaton. The combined transition is now carried out in two stages. Like before, $A$ (the timed automaton component of the combined machine) makes the transition based on $\langle s, \lambda, l, s'\rangle$. It then transfers control to machine $M$ by executing the command $ENTER(M, R)$, where $R$ is a one-step transition rule: $R(s, \lambda, l, s', q, Z, s_1, \ldots, s_k) = \{t_1, \ldots, t_m\}$. Note that the outcome of this transition (i.e., the right side of the rule) not only depends on $\langle s, \lambda, l, s'\rangle$, but also on the current state, status of the counters, and the topmost symbol of the stack. This $R$ is then followed by a sequence of transitions by $M$ (using the transition function $\delta$). Thus the use of $ENTER(M, R)$ allows the combined machine to update the configuration of $M$ through a sequence of $M$'s transitions. After some amount of computation, $M$ returns control to $A$ by entering a special state or command $RETURN$. When this happens, $A$ will now be in state $s'$. Thus the computation of $A \oplus M$ is like in a timed automaton, except that between each transition of $A$, the system calls $M$ to do some computation.

A configuration of the system is a tuple of the form $\alpha = (s, U, q, V, w)$. Thus, a configuration is a result of an execution of a (possibly empty) sequence of ($ENTER$, $RETURN$) commands. Note that a configuration can be represented as a string where the clock values $U$ and counter values $V$ are represented in unary and the components of the tuple separated by markers.

As defined earlier, the binary reachability is $Reach(A \oplus M) = the$ set of all pairs of configurations $(\alpha, \beta)$, where $\alpha$ can reach $\beta$. We will show that $Reach(A \oplus M)$ can be accepted by a 2-tape PCA. Note that the input to the acceptor is a pair of strings $(\alpha, \beta)$, where $\alpha$ ($\beta$) is on the first (second) tape.

First we note that we can view the clocks in a discrete timed automaton $A$ as counters, which we shall also refer to as clock-counters. In a reversal-bounded multicounter machine, only *standard tests* (comparing a counter against 0) and *standard assignments* (increment or decrement a counter by 1, or simply nochange) are allowed. But clock-counters in $A$ do not have standard tests nor standard assignments. The reasons are as follows. A clock constraint allows comparison between two clocks like $x_2 - x_1 > 7$. Note that using only standard tests we cannot directly compare the difference of two clock-counter values against an integer like 7 by computing $x_2 - x_1$ in another counter, since each time this computation is done, it will cause at least a counter reversal, and the number of such tests during a computation can be unbounded. The clock progress $x := x + 1$ is standard, but the clock reset $x := 0$ is not. Since there is no bound on the number of clock resets, clock-counters may not be reversal-bounded (each reset causes a counter reversal).

We first prove an intermediate result. Define a semi-PCA as a PCA which, in addition to a stack and reversal-bounded counters, has clock-counters that use nonstandard tests and assignments as described in the preceding paragraph.

**Lemma 1.** *We can effectively construct, given a discrete timed automaton $A$ and PCM $M$, a 2-tape semi-PCA $B$ accepting Reach$(A \oplus M)$.*

**Proof.** We describe the construction of the 2-tape semi-PCA $B$. Given a pair of configurations $(\alpha, \beta)$ on its two input tapes, $B$ first copies $\alpha$ into its counters and stack (these include the clock-counters). Then $B$ simulates the ("alternating" mode of) computation of $A \oplus M$ starting from configuration $\alpha$ as described above. It is clear that $B$ can do this. After some time, $B$ guesses that it has reached the configuration $\beta$. It then checks that the values of the counters and stack match those on the second input tape. $B$ accepts if the check succeeds. However, there is a slight complication because the pushdown stack content is in "reverse". If the stack content on the second tape is written in reversed, there is no problem. One can get around this difficulty if the comparison of the stack content with the second tape is done *during* the simulation instead of waiting until the end of the simulation. This involves guessing, for each position of the stack, the last time $M$ rewrites this position, i.e., that the symbol would not be rewritten further in reaching configuration $\beta$. So, e.g., if on stack position $p$, the symbol changes are $Z_1, \ldots, Z_k$ for the entire computation, then $Z_k$ is the last symbol written on the position, and $B$ checks after $Z_k$ is written that the $p$th position of the stack word in $\beta$ is $Z_k$. $M$ marks $Z_k$ in the stack and makes sure that this symbol is never popped or rewritten in the rest of the computation. □

The next lemma uses a technique from [7] (see also [9]).

**Lemma 2.** *We can effectively construct from the 2-tape semi-PCA $B$, a 2-tape PCA $C$ equivalent to $B$.*

**Proof.** The 2-tape PCA $C$ operates like $B$, but the simulation of $A \oplus M$ differs in the way $A$ is simulated. Let $A$ have clock-counters $x_1, \ldots, x_k$. Let $m$ be one plus the maximal absolute value of all the integer constants that appear in the tests (i.e., the clock constraints on the edges of $A$ in the form of Boolean combinations of $x_i \# c$, $x_i - x_j \# c$ with $c$ an integer). Denote the finite set $\{-m, \ldots, 0, \ldots, m\}$ by $[m]$. Define two finite tables with entries $a_{ij}$ and $b_i$ for $1 \leqslant i, j \leqslant k$. Each entry can be regarded as a finite state variable with states in $[m]$. Intuitively, $a_{ij}$ is used to record the difference between two clock values of $x_i$ and $x_j$, and $b_i$ is used to record the clock value of $x_i$. During the computation of $A$, when the difference $x_i - x_j$ (or the value $x_i$) goes above $m$ or below $-m$, $a_{ij}$ (or $b_i$) stays the same as $m$ or $-m$. The procedure for updating the entries is given below, where "$\oplus 1$" means adding one if the result does not exceed $m$, else it keeps the same value. "$\ominus 1$" means subtracting one if the result is not less than $-m$, else it keeps the same value. We modify $A$ as follows. Consider a transition edge in $A$. If on the edge the set of clock resets $\lambda = \emptyset$, the entries are updated by adding the following instructions for each $1 \leqslant i \leqslant k$:

- $a_{ij} := a_{ij}$ for each $1 \leqslant j \leqslant k$. Recall that all the clocks progress after this edge; thus, the difference is unchanged.
- $b_i := b_i \oplus 1$. That is, clocks progress by one time unit.

If the set of clock resets is $\lambda \neq \emptyset$, the entries are updated by adding the following instructions for each $1 \leqslant i, j \leqslant k$:

- $a_{ij} := 0$ if $i \in \lambda$ and $j \in \lambda$. In this case, both clocks $x_i$ and $x_j$ reset to 0.
- $a_{ij} := -b_j$ if $i \in \lambda$ and $j \notin \lambda$. In this case, $x_i$ resets but $x_j$ does not. So the difference should be $-x_j$.
- $a_{ij} := b_i$ if $i \notin \lambda$ and $j \in \lambda$.
- $a_{ij} := a_{ij}$ if $i \notin \lambda$ and $j \notin \lambda$.

We then add the following instructions:

- $b_i := b_i$ if $x_i \notin \lambda$.
- $b_i := 0$ if $x_i \in \lambda$.

The initial values of $a_{ij}$ and $b_i$ can be constructed directly from the values $\alpha_{x_i}$ of clocks $x_i$ in configuration $\alpha$, for each $1 \leqslant i, j \leqslant k$:

- $a_{ij} := \alpha_{x_i} - \alpha_{x_j}$ if $|\alpha_{x_i} - \alpha_{x_j}| \leqslant m$,
- $a_{ij} := m$ if $\alpha_{x_i} - \alpha_{x_j} > m$,
- $a_{ij} := -m$ if $\alpha_{x_i} - \alpha_{x_j} < -m$,

and, noticing that clocks are nonnegative,

- $b_i := \alpha_{x_i}$ if $\alpha_{x_i} \leqslant m$,
- $b_i := m$ if $\alpha_{x_i} > m$.

$C$ simulates $A$ exactly except that it uses $a_{ij} \# c$ for the test $x_i - x_j \# c$ and $b_i \# c$ for the test $x_i \# c$, with $-m < c < m$. One can prove (by induction) that doing this is valid: Each time after $C$ updates the entries by executing a transition, $x_i - x_j \# c$ iff $a_{ij} \# c$, and $x_i \# c$ iff $b_i \# c$, for all $1 \leqslant i, j \leqslant k$ and for each integer $c \in [m-1]$.

Thus clock-counter comparisons are replaced by finite table look-up and, therefore, nonstandard tests are not present in $C$. Finally, we show how nonstandard assignments of the form $x_i := 0$ (clock resets) in machine $C$ can be avoided.

Clearly after eliminating the clock comparisons, the clock-counters in $C$ do not participate in any tests except:

- at the beginning of the simulation when the initial values of the $x_i$'s are used to compute the initial values of the $a_{ij}$'s and the $b_i$'s as described above;
- at the end of the simulation when the final values of the $x_i$'s are compared with the second input tape to check whether they match those in $\beta$.

Thus, for each $x_i$, during the simulation of $A$ but before the last reset of $x_i$, the actual value of $x_i$ is irrelevant. We describe how to construct a 2-tape PCA $D$ from $C$ such that in the simulation of $A$, no nonstandard assignment is used. For each clock $x_i$ in $A$, there are two cases. The first case is when $x_i$ will not be reset during the entire simulation of $C$. The second case is when $x_i$ will be reset. $D$ guesses the case for each $x_i$. For the first case, $x_i$ is already reversal-bounded, since the nonstandard assignment $x_i := 0$ is not used. For the second case, $D$ first decrements $x_i$ to 0. Then $D$ simulates $C$. Whenever a clock progress $x_i := x_i + 1$ or a clock reset $x_i := 0$ is being executed by $A$, $D$ keeps $x_i$ as 0. But, at some point when a clock reset $x_i := 0$ is being executed by $A$, $D$ guesses that this is the last clock reset for $x_i$. After this point, $D$ faithfully simulates a clock progress $x_i := x_i + 1$ executed by $A$, and a later execution of a clock reset $x_i := 0$ in $A$ will cause $D$ to abort abnormally (since the guess of the last reset of $x_i$ was wrong). Thus $D$ uses only standard assignments $x_i := x_i + 1$, $x_i := x_i$, and $x_i := x_i - 1$ initially to bring $x_i$ to 0 (for the second case). □

From the above lemmas, we have:

**Theorem 8.** *We can effectively construct, given a discrete timed automaton $A$ and a PCM $M$, a 2-tape PCA accepting Reach($A \oplus M$).*

One can generalize Theorem 8. Extend a PCA acceptor by allowing the machine to have multiple pushdown stacks. Thus the machine will have multiple reversal-bounded counters and multiple stacks (ordered by name, say $S_1, \ldots, S_m$). The operation of the machine is restricted in that it can only read the topmost symbol of the *first* nonempty stack. Thus a move of the machine would depend only on the current state, the input symbol (or $\varepsilon$), the status of each counter (zero or nonzero), and the topmost symbol of the first stack, say $S_i$, that is not empty (initially, all stacks are set to some starting top symbol). The action taken in the move consists of the input being consumed, each counter being updated $(+1, -1, 0)$, the topmost symbol of $S_i$ being popped and a string (possibly empty) being pushed onto each stack, and the next state being entered. This acceptor, call it MPCA, was studied in [6] as a generalization of a PCA [8] and a generalization of a multipushdown acceptor [3]. Thus an MPCA with only one stack reduces to a PCA.

By combining the techniques in [8] and [3], it was shown in [6] that the emptiness problem for MPCAs is decidable. An MPCA without an input tape will be called an MPCM. By a construction similar to that of Theorem 8, we can prove the next result. Note that checking that the contents of the stacks at the end of the simulation are the same as the stack words in the target configuration does not require the latter to be in reverse (or need special handling), since we can first reverse the stack contents by using another set of pushdown stacks and then check that they match the stack words in the target configuration.

**Theorem 9.** *We can effectively construct, given a discrete timed automaton $A$ and an MPCM $M$, a 2-tape MPCA accepting Reach$(A \oplus M)$.*

Other examples of classes $\mathscr{C}$ that can be shown to have a decidable emptiness problem are given below. Thus, the results in Section 2 apply.

1. Nondeterministic machines with reversal-bounded counters and a two-way read/write worktape that is restricted in that the number of times the head crosses the boundary between any two adjacent cells of the worktape is bounded by a constant, independent of the computation (thus, the worktape is finite-crossing). There is no bound on how long the head can remain on a cell [9].
2. Nondeterministic machines with reversal-bounded counters and a queue that is restricted in that the number of alternations between nondeletion phase and noninsertion phase is bounded by a constant [9]. A nondeletion (noninsertion) phase is a period consisting of insertions (deletions) and no-changes, i.e., the queue is idle. Without the restriction emptiness is undecidable since it is known that a finite-state machine with an unrestricted queue can simulate a Turing machine.

Finally, as mentioned in the paragraph preceeding Theorem 7, we can provide the machine $A \oplus M$ with an input tape. The language accepted by such an acceptor can be shown to be accepted by an acceptor $M'$ which belongs to the same class as $M$ (the simulation is similar to the one described in Lemmas 1 and 2). Thus, Theorem 7 follows.

## 4. Applications

In this section we exhibit some properties of timed automata that can be verified using the results above.

**Example 1.** (Real-time) pushdown timed systems with "observation" counters were studied in [2]. The purpose of these counters is to record information about the evolution of the system and to reason about certain properties (e.g., number of occurrences of certain events in some computation). The counters do not participate in the dynamic of the system, i.e., they are never tested by the system. A transition edge specifies for each observation counter an integral value (positive, negative, zero) to be added to the

counter. Of interest are the values of the counters when the system reaches a specified configuration. It was shown in [2] that "region" reachability is decidable for these systems.

Clearly, for the discrete case, such a system can be simulated by the machine $A \oplus M$ described in the previous section. We associate in $M$ two counters for each observation counter: one counter keeps track of the positive increases and the other counter keeps track of the negative increases. When the target configuration is reached, the difference can be computed in one of the counters. Note that the sign of the difference can be specified in another counter, which is set to 0 for negative and 1 for positive. Thus, from Theorems 2–6, (binary, forward, backward) reachability, safety, and invariance are solvable for these systems.

**Example 2.** Let $A$ be a discrete timed automaton and $M$ be a nondeterministic pushdown machine with reversal-bounded counters. For a given computation of $A \oplus M$, let $r_i$ be the number of times clock $x_i$ resets. Suppose we are interested in computations in which the $r_i$'s satisfy a Presburger formula $f$, i.e., we are interested in $(\alpha, \beta)$ in $Reach(A \oplus M)$ such that $\alpha$ can reach $\beta$ in a computation in which the clock resets satisfy $f$. It is known that a set of $k$-tuples is definable by a Presburger formula $f$ if and only if it is definable by a reversal-bounded multicounter machine [8]. (Thus, a machine $M_f$ with no input tape but with reversal-bounded counters can be effectively constructed from $f$ such that when the values of the first $k$ counters are set to the $k$-tuple and all the other counters are initially zero, $M_f$ enters an accepting state if and only if the $k$-tuple satisfies $f$. In fact, $M_f$ can be made deterministic [8].) It follows that we can construct a 2-tape pushdown acceptor with reversal-bounded counters $M'$ accepting the set $Q$ of pairs of configurations $(\alpha, \beta)$ in $Reach(A \oplus M)$ such that $\alpha$ can reach $\beta$ in a computation in which the clock resets satisfy $f$. One can also put other constraints, like introducing a parameter $t_i$ for each clock $i$, and consider computations where the first time $i$ resets to zero is before (or after) time $t_i$. We can construct a 3-tape acceptor $M''$ from $M'$ accepting $Q(t_1, \ldots, t_k)$. $M''$ first reads the parameters $t_i$'s (which are given on the third input tape) and then simulates $M'$, checking that the constraint on the first time clock $i$ resets is satisfied. Note that if $M$ has no pushdown stack, then $Q$ and $Q(t_1, \ldots, t_k)$ are Presburger.

**Example 3.** As another example, suppose we are interested in the set $S$ of pairs of configurations $(\alpha, \beta)$ of a discrete timed automaton $A$ such that there is a computation path (i.e., sequence of states) from $\alpha$ to $\beta$ that satisfies a property that can be verified by an acceptor in a class $C$. If $C$ has a decidable emptiness problem, then $S$ is effectively computable. For example, suppose that the property is for the path to contain three nonoverlapping subpaths (i.e., segments of computation) which go through the same sequence of states, and the length of the subpath is no less than $\frac{1}{5}$ of the length of the entire path. Thus if $p$ is the computation path, there exist subpaths $p_1, \ldots, p_7$ (some may be null) such that $p = p_1 p_2 p_3 p_4 p_5 p_6 p_7$, where $p_2$, $p_4$, and $p_6$ go through the

same sequence of states, and length of $p_2$ = length of $p_4$ = length of $p_6$ is no less than $\frac{1}{5}$ of the length of $p$. We can check this property by incorporating a finite-crossing read-write tape to the machine (actually, the head need only make 5 crossings on the read-write tape).

**Example 4.** We can equip $A \oplus M$ with one-way write-only tapes which the machine can use to record certain information about the computation of the system (and perhaps even requiring that the strings appearing in these tapes satisfy some properties). From Corollary 1, such systems can effectively be analyzed.

## 5. Reachability in parallel discrete timed automata

The technique of using the reversal-bounded counters to record and compare various integers (like the running times of the machines) in the proofs in Section 3 can be used to decide some reachability questions concerning machines operating in parallel. We give two examples below.

Let $A_1, A_2$ be discrete timed automata and $M_1, M_2$ be PCMs. Recall from Section 3 that a configuration of $A_i \oplus M_i$ is a 5-tuple $\alpha_i = (s_i, U_i, q_i, V_i, w_i)$. Suppose we are given a pair of configurations $(\alpha_1, \beta_1)$ of $A_1 \oplus M_1$ and a pair of configurations $(\alpha_2, \beta_2)$ of $A_2 \oplus M_2$, and we want to know if $A_i \oplus M_i$ when started in configuration $\alpha_i$ can reach configuration $\beta_i$ at some time $t_i$, with $t_1$ and $t_2$ satisfying a given linear relation $L(t_1, t_2)$ definable by a Presburger formula. (Thus, e.g., if the linear relation is $t_1 = t_2$, then we want to determine if $A_1 \oplus M_1$ when started in configuration $\alpha_1$ reaches $\beta_1$ at the same time that $A_2 \oplus M_2$ when started in $\alpha_2$ reaches $\beta_2$.) This reachability question is decidable. The idea is the following. First note that we can incorporate a counter in $M_i$ that records the running time $t_i$ of $A_i \oplus M_i$. Let $Z_i$ be a 2-tape PCA accepting $R(A_i \oplus M_i)$. We construct a 4-tape PCA $Z$ which, when given $\alpha_1, \beta_1, \alpha_2, \beta_2$ in its 4 tapes, first simulates the computation of $Z_1$ to check that $\alpha_1$ can reach $\beta_1$, recording the running time $t_1$ (which is in configuration $\beta_1$) of $A_1 \oplus M_1$ in a counter. $Z$ then simulates $Z_2$. Finally, $Z$ checks that the running times $t_1$ and $t_2$ satisfy the given linear relation (which can be verified since Presburger formulas can be evaluated by nondeterministic reversal-bounded multicounter machines). Since the emptiness problem for PCAs is decidable, decidability of reachability follows.

We can allow the machines $A_1 \oplus M_1$ and $A_2 \oplus M_2$ to share a common input tape, i.e., each machine has a one-way read-only input head (see the paragraph preceding Theorem 7). A configuration $\alpha_i$ will now be a 7-tuple $\alpha_i = (s_i, U_i, q_i, V_i, w_i, h_i)$, $h_i$ is the position of the input head on the common input $x$. One can show that if both $A_1 \oplus M_1$ and $A_2 \oplus M_2$ have a one-turn stack (or an unrestricted counter), then reachability is undecidable, even if they have no reversal-bounded counters and the linear relation is $t_1 = t_2$. However, if only one of $A_1 \oplus M_1$ and $A_2 \oplus M_2$ has an unrestricted pushdown stack, then reachability is decidable. Again, the idea is to construct a 5-tape PCA which, when given $\alpha_1, \beta_1, \alpha_2, \beta_2, x$, first simulates $M_1$ and $M_2$ in parallel on the input $x$.

If one of the machines, e.g., $M_1$ advances its input head to the next input symbol, but $M_2$ has not yet read the current input symbol, $M$ does not advance its input head and "suspends" the simulation of $M_1$ until $M_2$ has read the current symbol or $M$ guesses that $M_2$ will not be reading further on the input to reach the target configuration.

Note that the above results generalize to any number, $k$, of machines $A_i \oplus M_i$ $(i = 1, \ldots, k)$ operating in parallel.

## 6. Conclusions

We showed that a discrete timed automaton augmented with a machine with reversal-bounded counters and possibly other data structures from a class $\mathscr{C}$ of machines can be effectively analyzed with respect to reachability, safety, and other properties if $\mathscr{C}$ has a decidable emptiness problem. We gave examples of such $\mathscr{C}$'s and examples of new properties of discrete timed automata that can be verified. We also showed that reachability in parallel machines can be effectively decided. It would be interesting to look for other classes of $\mathscr{C}$'s with decidable emptiness problem.

## References

[1] R. Alur, D. Dill, Automata for modeling real-time systems, Theoret. Comput. Sci. 126 (2) (1994) 83–236.
[2] A. Bouajjani, R. Echahed, R. Robbana, in: P.J. Antsalilis, W. Kohn, M.O. Lemmon, A.N. Nerode, S. Sastry (Eds.) On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures, In Hybrid Systems II, Lecture Notes in Computer Science, Vol. 999, Springer, Berlin, 1995.
[3] A. Cherubini, L. Breveglieri, C. Citrini, S. Crespi Reghizzi, Multi-push-down languages and grammars, Int. J. Foundations Comput. Sci. 7 (3) (1996) 253–291.
[4] H. Comon, Y. Jurski, Multiple counters automata, safety analysis and Presburger arithmetic, Proc. 10th Int. Conf. on Computer Aided Verification, 1998, pp. 268–279.
[5] H. Comon, Y. Jurski, Timed automata and the theory of real numbers, Proc. 10th Int. Conf. on Concurrency Theory, 1999, pp. 242–257.
[6] Z. Dang, Verification and Debugging of Infinite State Real-time Systems, Ph.D. Thesis, University of California, Santa Barbara, 2000.
[7] Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su, Binary reachability analysis of discrete pushdown timed automata, Int. Conf. on Computer Aided Verification, 2000, pp. 69–84.
[8] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. Assoc. Comput. Machin. 25 (1) (1978) 116–133.
[9] O.H. Ibarra, T. Bultan, J. Su, Reachability analysis for some models of infinite-state transition systems, Proc. 10th Int. Conf. on Concurrency Theory, 2000, pp. 183–198.
[10] O.H. Ibarra, Z. Dang, P. San Pietro, Queue-Connected Discrete Timed Automata, Theoret. Comput. Sci., submitted for publication.
[11] M. Minsky, Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines, Ann. Math. 74 (1961) 437–455.