Binary Reachability Analysis of Discrete Pushdown Timed Automata

Zhe Dang*, Oscar H. Ibarra**, Tevfik Bultan***,
Richard A. Kemmerer* and Jianwen Su**

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

Abstract. We introduce discrete pushdown timed automata that are timed automata with integer-valued clocks augmented with a pushdown stack. A configuration of a discrete pushdown timed automaton includes a control state, finitely many clock values and a stack word. Using a pure automata-theoretic approach, we show that the binary reachability (i.e., the set of all pairs of configurations (α, β), encoded as strings, such that α can reach β through 0 or more transitions) can be accepted by a nondeterministic pushdown machine augmented with reversal-bounded counters (NPCM). Since discrete timed automata with integer-valued clocks can be treated as discrete pushdown timed automata without the pushdown stack, we can show that the binary reachability of a discrete timed automaton can be accepted by a nondeterministic reversal-bounded multicounter machine. Thus, the binary reachability is Presburger. By using the known fact that the emptiness problem is decidable for reversal-bounded NPCMs, the results can be used to verify a number of properties that can not be expressed by timed temporal logics for discrete timed automata and CTL* for pushdown systems.

1 Introduction

After the introduction of efficient automated verification techniques such as symbolic model-checking [16], finite state machines have been widely used for modeling reactive systems. Due to the limited expressiveness, however, they are not suitable for specifying most infinite state systems. Thus, searching for models to represent more general transition systems and analyzing the decidability of their verification problems such as reachability or model-checking is an important research issue. In this direction, several models have been investigated such as pushdown automata[4, 12, 17], timed automata[2] (and real-time logics[3, 1, 14]), and various approximations on multicounter machines[9, 7].

* This work is supported in part by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0207.
** This work is supported in part by NSF grant IRI-9700370.
*** This work is supported in part by NSF grant CCR-9970976.

A pushdown system is a finite state machine augmented by a pushdown stack. On the other hand, a timed automaton can be regarded as a finite state machine with a number of real-valued clocks. All the clocks progress synchronously with rate 1, and a clock can be reset to 0 at some transition. Each transition also comes with an enabling condition in the form of clock constraints (i.e., Boolean combinations of $x\#c$ and $x - y\#c$ where x and y are clocks, c is an integer constant, and $\#$ denotes $>, <$ or $=$. Such constraints are also called regions.). A standard region technique [2] (and more recent techniques[6, 18]) can be used to analyze region reachability.

In this paper, we consider integer-valued clocks. We call a timed automaton with integer-valued clocks a *discrete timed automaton*. A strictly more powerful system can be obtained by combining a pushdown system with a discrete timed automaton. That is, a discrete timed automaton is augmented with a pushdown stack (i.e., a discrete pushdown timed automaton). We give a characterization of binary reachability $\rightsquigarrow$, defined as the set of pairs of configurations (control state and clock values, plus the stack word if applicable) (α, β) such that α can reach β through 0 or more transitions. Binary reachability characterization is a fundamental step towards developing a model checking algorithm for discrete pushdown timed automata. From classical automata theory, it is known that the binary reachability of pushdown automata is context-free. For timed automata (with either real-valued clocks or integer-valued clocks), the region technique is not enough to give a characterization of the binary reachability. Recently, Comon et. al. [10] showed that the binary reachability of timed automata (with real-valued clocks) is expressible in the additive theory of reals. They show that a timed automaton with real-valued clocks can be flattened into one without nested cycles. Their technique also works for discrete timed automata. However, it is not easy to deduce a characterization of the binary reachability of discrete pushdown timed automata by combining the above results. The reason is, as pointed out in their paper, this flattening destroys the structure of the original automaton. That is, the flattened timed automaton accepts different sequences of transitions, though the binary reachability is still the same. Thus, their approach cannot be used to show the binary reachability of the discrete pushdown timed automata proposed in this paper, since by flattening the sequence of stack operations cannot be maintained. A class of Pushdown Timed Systems (with continuous clocks) was discussed in [5]. However, that paper focuses on region reachability instead of binary reachability.

In this paper, we develop a new automata-theoretic technique to characterize the binary reachability of a discrete pushdown timed automaton. Our technique does not use the region technique [2] nor the flattening technique [10]. Instead, a nondeterministic pushdown multicounter machine (NPCM), which is a nondeterministic pushdown automaton with counters, is used. Obviously, without restricting the counter behaviors, even the halting problem is undecidable, since machines with two counters already have an undecidable halting problem. An NPCM is reversal-bounded if the number of counter reversals (a counter changing mode between nondecreasing and nonincreasing and vice-versa) is bounded

by some fixed number independent of the computation. We show that the binary reachability of a discrete pushdown timed automaton can be accepted by a reversal-bounded nondeterministic pushdown multicounter machine. We also discuss the safety analysis problem. That is, given a property P and an initial condition I, which are two sets of configurations of a discrete pushdown timed automaton $\mathcal{A}$, determine whether, starting from a configuration in I, $\mathcal{A}$ can only reach configurations in P. Using the above characterization and the known fact that the emptiness problem for reversal-bounded NPCMs is decidable, we show that the safety analysis problem is decidable for discrete pushdown timed automata, as long as both the safety property and the initial condition are accepted by nondeterministic reversal-bounded multicounter machines.

It is known that Presburger relations can be accepted by reversal-bounded multicounter machines. Therefore, it is immediate that the safety analysis problem is decidable as long as both the safety property and the initial condition are Presburger formulas on clocks. A discrete timed automaton can be treated as a discrete pushdown timed automaton without the pushdown stack. We can show that the binary reachability of a discrete timed automaton can be accepted by a reversal-bounded nondeterministic multicounter machine (i.e., a reversal-bounded NPCM without the pushdown stack). That is, the binary reachability of a discrete timed automaton is Presburger. This result shadows the result in [10] that the binary reachability of a timed automaton with real-valued clocks is expressible in the additive theory of reals, although our approach is totally different.

The characterization of $\rightsquigarrow$ for discrete pushdown timed automata will lead us to formulate a model checking procedure for a carefully defined temporal logic. The logic can be used to reason about a class of timed pushdown processes. Due to space limitation, we omit it here. In fact, the binary reachability characterization itself already demonstrates a wide range of safety properties that can be verified for discrete pushdown timed automata. We will show this by investigating a number of examples of properties at the end of the paper.

2 Discrete Pushdown Timed Automata

A timed automaton [2] is a finite state machine augmented with a number of real-valued clocks. All the clocks progress synchronously with rate 1, except a clock can be reset to 0 at some transition. In this paper, we 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 $\leq, \geq, <, >$, or $=$, c is an integer, x, y are integer-valued clocks. Let $\mathcal{L}_X$ be the set of all clock constraints on clocks X. Let $\mathbf{Z}$ be the set of integers with $\mathbf{Z}^+$ for nonnegative integers. Formally, a *discrete timed automaton* $\mathcal{A}$ is a tuple $\langle S, X, E \rangle$ where S is a finite set of *(control) states.* X is a finite set of *clocks* with values in $\mathbf{Z}^+$. $E \subseteq S \times 2^X \times \mathcal{L}_X \times S$ is a finite set of *edges* or *transitions.* Each edge $\langle s, \lambda, l, s' \rangle$ denotes a transition from state s to state s' with *enabling condition* $l \in \mathcal{L}_X$ and a set of clock resets $\lambda \subseteq X$. Note that λ may be empty. Also note that since each pair of states may have

more than one edge between them, $\mathcal{A}$ is, in general, nondeterministic.

The semantics is defined as follows. $\alpha \in S \times (\mathbf{Z}^+)^{|X|}$ is called a *configuration* with α_x being the value of clock x and α_q being the state under this configuration. $\alpha \rightarrow^{\langle s,\lambda,l,s'\rangle} \alpha'$ denotes a one-step transition along an edge $\langle s, \lambda, l, s'\rangle$ in $\mathcal{A}$ satisfying

- The state s is set to a new location s', i.e., $\alpha_q = s, \alpha'_q = s'$.
- Each clock changes according to the edge given. If there are no clock resets on the edge, i.e., $\lambda = \emptyset$, then clocks progress by one time unit, i.e., for each $x \in X$, $\alpha'_x = \alpha_x + 1$. If $\lambda \neq \emptyset$, then for each $x \in \lambda$, $\alpha'_x = 0$ while for each $x \notin \lambda$, $\alpha'_x = \alpha_x$.
- The enabling condition is satisfied, that is, $l(\alpha)$ is true.

We simply write $\alpha \rightarrow \alpha'$ if α can reach α' by a one-step transition. A *path* $\alpha_0 \cdots \alpha_k$ satisfies $\alpha_i \rightarrow \alpha_{i+1}$ for each i. Also write $\alpha \rightsquigarrow^{\mathcal{A}} \beta$ if α reaches β through a path. Given a set P of configurations of $\mathcal{A}$, write the *preimage* $Pre^*(P)$ of P as the set of configurations that can reach a configuration in P, i.e.,

$$Pre^*(P) =_{def} \{\alpha : \text{ for some } \beta \in P, \alpha \rightsquigarrow^{\mathcal{A}} \beta\}.$$

The following figure shows an example of a discrete timed automaton with two clocks x_1 and x_2. The following sequence of configurations is a path: $\langle s_0, x_1 = 0, x_2 = 0\rangle, \langle s_1, x_1 = 0, x_2 = 0\rangle, \langle s_0, x_1 = 1, x_2 = 1\rangle, \langle s_1, x_1 = 1, x_2 = 0\rangle$.

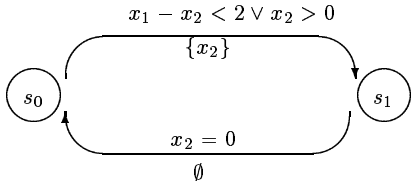

Figure 1. An example discrete timed automaton

The above defined $\mathcal{A}$ is a little different from the standard (discrete) timed automaton given in [2]. In that model, each state is assigned with a clock constraint called an *invariant* in which $\mathcal{A}$ can remain in the same control state with all the clocks synchronously progressing with rate 1 as long as the invariant is satisfied. It is easy to see that, when integer-valued clocks are considered, $\mathcal{A}$'s remaining in a state can be replaced by a self-looping transition with the invariant as the enabling condition and without clock resets. Each execution of such a transition causes all the clocks to progress by one time unit. Another difference is that in a standard timed automaton a state transition takes no time, even when the transition has no clock resets. In order to translate a standard timed automaton to our definition, we introduce a dummy clock. Thus, for each state transition t in a standard timed automaton the translated transition t' is exactly the same except the dummy clock is reset in t'. Thus, doing this will ensure that all clock values remain the same when t has no clock resets. Thus, standard timed automata can be easily transformed into the ones defined above. Since the paper focuses on binary reachability, the ω-language accepted by a timed

automaton is irrelevant here. Thus, event labels in a standard timed automaton are not considered in this paper.

Discrete timed automata can be further extended by allowing a pushdown stack. A *discrete pushdown timed automaton* $\mathcal{A}$ is a tuple $\langle \Gamma, S, X, E \rangle$ where Γ is the stack alphabet, and S, X, E are the same as in the definition of a discrete timed automaton except that each edge includes a stack operation. That is, each edge e is in the form of $\langle s, \lambda, (\eta, \eta'), l, s' \rangle$ where $s, s' \in S$, $\lambda \subseteq X$ is the set of clock resets and $l \in \mathcal{L}_X$ is the enabling condition. The stack operation is characterized by a pair (η, η') with $\eta \in \Gamma$ and $\eta' \in \Gamma^*$. That is, replacing the top symbol of the stack η by a word η'. A configuration $\alpha \in (\mathbf{Z}+)^{|X|} \times S \times \Gamma^*$ with $\alpha_w \in \Gamma^*$ indicating the stack content. $\alpha \rightsquigarrow^{\mathcal{A}} \beta$ can be similarly defined assuming that the stack contents in α and β are consistent with the sequence of stack operations along the path.

This paper focuses on the characterization of binary reachability $\rightsquigarrow^{\mathcal{A}}$ for both discrete timed automata and discrete pushdown timed automata. Before we proceed to show the results, some further definitions are needed.

A *nondeterministic multicounter machine (NCM)* is a nondeterministic machine with a finite set of *(control) states* $Q = \{1, 2, \cdots, |Q|\}$, and a finite number of counters $x_1, \cdots, x_k$ with integer counter values. Each counter can add 1, subtract 1, or stay unchanged. Those counter assignments are called *standard assignments*. M can also test whether a counter is equal to, greater than, or less than an integer constant. Those tests are called *standard tests*.

An NCM can be augmented with a pushdown stack. A *nondeterministic pushdown multicounter machine (NPCM)* M is a nondeterministic machine with a finite set of *(control) states* $Q = \{1, 2, \cdots, |Q|\}$, a pushdown stack with stack alphabet Γ, and a finite number of counters $x_1, \cdots, x_k$ with integer counter values. Both assignments and tests in M are standard. In addition, M can pop the top symbol from the stack or push a word in Γ^* on the top of the stack. It is well-known that counter machines with two counters have undecidable halting problem, and obviously the undecidability holds for machines augmented with a pushdown stack. Thus, we have to restrict the behaviors of the counters. One such restriction is to limit the number of reversals a counter can make. A counter is *n-reversal-bounded* if it changes mode between nondecreasing and nonincreasing at most n times. For instance, the following sequence of a counter values: $0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 3, 2, 1, 1, 1, 1, \cdots$ demonstrates only one counter reversal. A counter is *reversal-bounded* if it is *n-reversal-bounded* for some n. We note that a reversal-bounded M (i.e., each counter in M is reversal-bounded) does not necessarily limit the number of moves or the number of reachable configurations to be finite.

Let $(j, v_1, \cdots, v_k, w)$ denote the *configuration* of M when it is in state $j \in Q$, counter x_i has value $v_i \in \mathbf{Z}$ for $i = 1, 2, \cdots, k$, and the string in the pushdown stack is $w \in \Gamma^*$ with the rightmost symbol being the top of the stack. Each integer counter value v_i can be represented by a unary string 0^{v_i} (1^{v_i}) when v_i positive (negative). Thus, a configuration $(j, v_1, \cdots, v_k, w)$ can be represented as a string by concatenating the unary representations of each j,$v_1, \cdots, v_k$ as

well as the string w with a separator $\# \notin \Gamma$. For instance, (1,2,-2,w) can be represented by $0^1\#0^2\#1^2\#w$. Similarly, an integer tuple $(v_1, \cdots, v_k)$ can also be represented by a string. Thus, in this way, a set of configurations and a set of integer tuples can be treated as sets of strings, i.e., a language. It is noticed that a configuration α of a discrete (pushdown) timed automaton $\mathcal{A}$ can be similarly encoded as a string $[\alpha]$.

Note that the above defined M does not have an input tape; in this case it is used as a system specification rather than a language recognizer, in which we are more interested in the behaviors that M generates. When a NPCM (or NCM) M is used as a language recognizer, we attach a separate one-way read-only input tape to the machine and assign a state in Q as the final state. M *accepts* an input iff it can reach the final state. When M is reversal-bounded, the emptiness problem, i.e., whether M accepts some input, is known to be decidable,

Theorem 1. *The emptiness problem for reversal-bounded nondeterministic pushdown multicounter machines with a one-way input tape is decidable [15].*

It has been shown in [13] that the emptiness problem for reversal-bounded nondeterministic multicounter machines (NCMs) with one-way input is decidable in n^{ckr} time for some constant c, where n is the size of the machine, k is the number of counters, and r is the reversal-bound on each counter. We believe that a similar bound could be obtained for the case of NPCMs.

Actually, Theorem 1 can be strengthened for the case of NCMs:

Theorem 2. *A set of n-tuples of integers is definable by a Presburger formula iff it can be accepted by a reversal-bounded nondeterministic multicounter machine [15].*

A language is *bounded* if there exist finite words $w_1, \cdots, w_n$ such that each element can be represented as $w_1^* \cdots w_n^*$. A nondeterministic reversal-bounded multicounter machine can be made deterministic on bounded languages.

Theorem 3. *If a bounded language L is accepted by a nondeterministic reversal-bounded multicounter machine, then L can also be accepted by a deterministic reversal-bounded multicounter machine [15].*

For an NPCM M, we can define the *preimage* $Pre^*(P)$ of a set of configurations P similarly to be the set of all predecessors of configurations in P, i.e., $Pre^*(P) = \{t \mid t$ can reach some configuration t' in P in 0 or more moves$\}$. Recently, we have shown [11] that $Pre^*(P)$ can be accepted by a reversal-bounded NPCM assuming that M is reversal-bounded and P is accepted by a reversal-bounded nondeterministic multicounter machine.

Theorem 4. *Let M be a reversal-bounded nondeterministic pushdown multicounter machine. Suppose a set of configurations P is accepted by a reversal-bounded nondeterministic multicounter machine. Then $Pre^*(P)$ with respect to M can be accepted by a reversal-bounded nondeterministic pushdown multicounter machine.*

3 Main Results

Let $\mathcal{A}$ be a discrete pushdown timed automaton with clocks $x_1, \cdots, x_k$. The binary reachability $\leadsto^{\mathcal{A}}$ can be treated as a language $\{[\alpha]\#[\beta]^r : \alpha \leadsto^{\mathcal{A}} \beta\}$ where $[\alpha]$ is the string encoding of configuration α, $[\beta]^r$ is the reverse string encoding of configuration β. [4] The two encodings are separated by a delimiter "#". The main result claims that the binary reachability $\leadsto^{\mathcal{A}}$ can be accepted by a reversal-bounded NPCM using standard tests and assignments. $\mathcal{A}$ itself can be regarded as an NPCM, when we refer to a clock as a counter. However, tests in $\mathcal{A}$ as clock constraints are not standard tests. Furthermore, $\mathcal{A}$ is not reversal-bounded since clocks can be reset for an unbounded number of times.

The proof of the main result proceeds as follows. We first show that $\leadsto^{\mathcal{A}}$ can be accepted by a reversal-bounded NPCM using nonstandard tests and assignments. Then, we show that these nonstandard tests can be made standard. Finally, these nonstandard assignments can be simulated by standard ones. Throughout the two simulations the counters remain reversal-bounded.

First, we show that clocks $x_1, \cdots, x_k$ in $\mathcal{A}$ can be translated into reversal-bounded ones. Let $y_0, y_1, \cdots, y_k$ be another set of clocks such that $x_i = y_0 - y_i$ $(1 \leq i \leq k)$. Let $\mathcal{A}'$ be a discrete pushdown timed automaton that is exactly the same as $\mathcal{A}$, except

- $\mathcal{A}'$ has clock y_0 that never resets. Intuitively, the *now-clock* y_0 denotes current time.
- Each y_i with $1 \leq i \leq k$ denotes the (last) time when a reset of clock x_i happens. Thus, each reset of clock x_i on an edge is replaced by updating y_i to the current time, i.e., $y_i := y_0$. If x_i does not reset on an edge, the value of y_i is unchanged. Also, only if there is no clock reset on an edge, add an assignment $y_0 := y_0 + 1$ to the edge to indicate that the now-clock progresses with one time unit. Only these assignments can change y_0.
- the enabling condition on each edge of $\mathcal{A}$ is replaced by substituting x_i with $y_0 - y_i$. Note that the enabling conditions $x_i \# c$ and $x_i - x_j \# c$ become $y_0 - y_i \# c$ and $y_j - y_i \# c$, respectively. Thus, the resulting enabling conditions are Boolean combinations of $y_i - y_j \# c$ with $0 \leq i, j \leq k$ and c being an integer constant.

Counters $y_0, y_1, \cdots, y_k$ in $\mathcal{A}'$ do not reverse. The reason is that assignments that change the counter values are only in the form of: $y_0 := y_0 + 1$ and $y_i := y_0$ for $1 \leq i \leq k$, and there is no way that a counter y_i decreases. For a configuration α of $\mathcal{A}$ and $u \in \mathbf{Z}^+$, write α^u to be a configuration of $\mathcal{A}'$ such that $\alpha^u_{y_0} = u$ (y_0's value is u), and for each $1 \leq i \leq k$, $\alpha^u_{y_i} = u - \alpha_{x_i}$ (y_i is the translation of x_i), and $\alpha^u_w = \alpha_w$ (the stack content is the same). Also write max_α to be the maximal value of clocks α_{x_i} in α (note that, each α_{x_i} is nonnegative by definition). Thus, α^{max_α} is the configuration α^u of $\mathcal{A}'$ with y_0's value being max_α. It follows directly, by induction on the length of a path, that the binary reachability of $\mathcal{A}$ can be characterized by that of $\mathcal{A}'$ as follows.

[4] The reason that we use the reverse encoding of β will become apparent in the Proof of Theorem 6.

Theorem 5. *For any pair of configurations α and β of a discrete pushdown timed automaton $\mathcal{A}$, the following holds,*

$$\alpha \rightsquigarrow^{\mathcal{A}} \beta \text{ iff there exist } v \in \mathbf{Z}^+ \text{ with } v \geq max_\alpha \text{ such that } \alpha^{max_\alpha} \rightsquigarrow^{\mathcal{A}'} \beta^v.$$

From the above theorem, it suffices for us to investigate the binary reachability of $\mathcal{A}'$. As mentioned above, $\mathcal{A}'$ is an NPCM with reversal-bounded counters $y_0, y_1, \cdots, y_k$. However, instead of standard tests, $\mathcal{A}'$ has tests that check an enabling condition by comparing the difference of two counters against an integer constant. Also the assignments include only $y_0 := y_0 + 1$ and $y_i := y_0$ for $1 \leq i \leq k$ in $\mathcal{A}'$, which are not standard assignments. The following theorem says the nonstandard tests can be made standard.

Theorem 6. *The binary reachability $\rightsquigarrow^{\mathcal{A}'}$ of $\mathcal{A}'$ can be accepted by a reversal-bounded NPCM using standard tests and nonstandard assignments that are of the form $y_0 := y_0 + 1$ and $y_i := y_0$ with $1 \leq i \leq k$.*

Proof. We construct the reversal-bounded NPCM as required. Given a pair of string encodings of configurations $\alpha^{\mathcal{A}'}$ and $\beta^{\mathcal{A}'}$ (separated by a delimiter "#" not in the stack alphabet, also recall that the encoding of $\beta^{\mathcal{A}'}$ has the stack word in $\beta^{\mathcal{A}'}$ reversed.) of $\mathcal{A}'$ on M's one-way input tape, M first copies $\alpha^{\mathcal{A}'}$ into its $k+1$ counters $y_0, y_1, \cdots, y_k$ and the stack. Thus, M's input head stops at the beginning of $\beta^{\mathcal{A}'}$. M starts simulating $\mathcal{A}'$ as follows with the stack operations in $\mathcal{A}'$ being exactly simulated on its own stack. Tests in $\mathcal{A}'$ are Boolean combinations of $y_i - y_j \# c$ for $0 \leq i, j \leq k$. Using only standard tests, M cannot directly compare the difference of two counter values against an integer c by storing $y_i - y_j$ in another counter, since each time this "storing" is done it will cause at least a counter reversal, and we don't have a bound on the number of such tests. In the following, we provide a technique to avoid such nonstandard tests. Assume m is one plus the maximal absolute value of all the integer constants that appear in the tests in $\mathcal{A}'$. Denote the finite set $[m] =_{def} \{-m, \cdots, 0, \cdots, m\}$. M uses its finite control to build a finite table. For each pair of counters y_i and y_j with $0 \leq i, j \leq k$, there is a pair of entries a_{ij} and b_{ij}. Each entry can be regarded as finite state control variable with states in $[m]$. Intuitively, a_{ij} is used to record the difference between the values of two counters y_i and y_j. b_{ij} is used to record the "future" value of the difference when a clock assignment $y_i := y_0$ occurs in the future. During the computation of $\mathcal{A}'$, when the difference goes beyond m or below $-m$, a_{ij} stays the same as m or $-m$. M uses $a_{ij} \# c$ to do a test $y_i - y_j \# c$. Doing this is always valid, as we will show later. Thus, M only uses standard tests. Below, "ADD 1" means adding one if the result does not exceed m, otherwise it keeps the same value. "SUBTRACT 1" means subtracting one if the result is not less than $-m$, otherwise it keeps the same value. In the following, we show how to construct the table. When assignment $y_0 := y_0 + 1$ is being executed by $\mathcal{A}'$, M updates the table as follows, for each $0 \leq i, j \leq k$:

- a_{ij} stays the same if $i > 0$ and $j > 0$. That is, the now-clock's progressing does not affect the difference between two non-now-clocks,

- a_{ij} ADD 1 if $i = 0$ and $j > 0$, noticing that y_i is the now-clock and y_j is a non-now-clock (thus it remains unchanged),
- a_{ij} SUBTRACT 1 if $i > 0$ and $j = 0$, noticing that y_j is the now-clock and y_i is a non-now-clock (thus it remains unchanged),
- a_{ij} is always 0 if $i = 0$ and $j = 0$. The difference between two identical now-clocks is always 0.

After updating all a_{ij}, entries b_{ij} are updated as below, for each $0 \leq i, j \leq k$,

- $b_{ij} := a_{0j}$. Thus b_{ij} is the value of $y_i - y_j$ assuming currently there is a jump $y_i := y_0$.

It is noticed that an edge in $\mathcal{A}'$ cannot contain two forms of assignment, i.e., both $y_0 := y_0 + 1$ and $y_i := y_0$. Let $\tau \subseteq \{y_1, \cdots, y_k\}$ denote assignments $y_i := y_0$ for $i \in \tau$ on an edge being executed by $\mathcal{A}'$. M updates the table as follows, for each $0 \leq i, j \leq k$:

- $a_{ij} := 0$ if $i, j \in \tau$, noticing that both y_i and y_j are currently the same value as the now-clock y_0,
- $a_{ij} := b_{ij}$ if $i \in \tau$ and $j \notin \tau$, noticing that y_i currently is the same value of the now-clock y_0 and the difference $y_i - y_j$ is prestored as b_{ij},
- $a_{ij} := -b_{ji}$ if $i \notin \tau$ and $j \in \tau$, noticing that $y_i - y_j = -(y_j - y_i)$,
- a_{ij} stays the same if $i \notin \tau$ and $j \notin \tau$, since clocks outside τ are not changed.

After updating all a_{ij}, entries b_{ij} are updated as follows, for each $0 \leq i, j \leq k$:

- $b_{ij} := 0$ if $i, j \in \tau$, noticing that both y_i and y_j are currently the same value as the now-clock y_0,
- $b_{ij} := a_{0j}$ if $i \in \tau$ and $j \notin \tau$, noticing that y_i currently is the same value as the now-clock y_0,
- $b_{ij} := -a_{0i}$ if $i \notin \tau$ and $j \in \tau$, noticing that y_j currently is set to the now-clock y_0,
- b_{ij} stays the same if $i \notin \tau$ and $j \notin \tau$, noticing that b_{ij} represents $y_0 - y_j$ and in fact the two clocks y_0 and y_j are unchanged after the transition.

The initial values of a_{ij} and b_{ij} can be constructed directly from $\alpha^{\mathcal{A}'}$ as follows, for each $0 \leq i, j \leq k$:

- $a_{ij} := \alpha^{\mathcal{A}'}_{y_i} - \alpha^{\mathcal{A}'}_{y_j}$ if $|\alpha^{\mathcal{A}'}_{y_i} - \alpha^{\mathcal{A}'}_{y_j}| \leq m$,
- $a_{ij} := m$ if $\alpha^{\mathcal{A}'}_{y_i} - \alpha^{\mathcal{A}'}_{y_j} > m$,
- $a_{ij} := -m$ if $\alpha^{\mathcal{A}'}_{y_i} - \alpha^{\mathcal{A}'}_{y_j} < -m$,

and for each $0 \leq i, j \leq k$: $b_{ij} := a_{0j}$.

M then simulates $\mathcal{A}'$ exactly except using $a_{ij} \# c$ for a test $y_i - y_j \# c$ in $\mathcal{A}'$, with $-m < c < m$. Then, we claim that doing this by M is valid,

Claim. Each time after M updates the table by executing a transition, $y_i - y_j \# c$ iff $a_{ij} \# c$, and $y_0 - y_j \# c$ iff $b_{ij} \# c$, for all $0 \leq i, j \leq k$ and for each integer $c \in [m - 1]$.

Proof of the Claim. We prove it by induction. Obviously, the Claim holds for the initial values of all clocks y_i (in configuration $\alpha^{\mathcal{A}'}$) and the corresponding entries a_{ij} and b_{ij}, by the choice of m. Suppose that $\mathcal{A}'$ is currently at configuration γ and the Claim holds. Thus, for all $0 \leq i, j \leq k$ and for each integer $c \in [m-1]$, $\gamma_{y_i} - \gamma_{y_j} \# c$ iff $a_{ij} \# c$ and $\gamma_{y_0} - \gamma_{y_j} \# c$ iff $b_{ij} \# c$ hold. Therefore, γ satisfies an enabling condition in $\mathcal{A}'$ iff the entries a_{ij} satisfy the same enabling condition by replacing $y_i - y_j$ with a_{ij}, noticing that m is chosen such that it is greater than the absolute value of any constant in all the enabling conditions in $\mathcal{A}'$. Assume γ satisfies the enabling condition on an edge e and $\mathcal{A}'$ will execute e next. Thus, M, using the entries a_{ij} to test the enabling condition, will also execute the same edge. We use γ' to denote the configuration after executing the edge, and use a'_{ij} and b'_{ij} to denote the table entries after executing the edge. We need to show, for all $0 \leq i, j \leq k$ and for each integer $c \in [m-1]$,

(*) $\quad \gamma'_{y_i} - \gamma'_{y_j} \# c$ iff $a'_{ij} \# c$

and

(**) $\quad \gamma'_{y_0} - \gamma'_{y_j} \# c$ iff $b'_{ij} \# c$

hold. There are two cases to be considered according to the form of the assignment. Suppose the assignment on e is a clock progress $y_0 := y_0 + 1$. After this assignment, $\gamma'_{y_0} = \gamma_{y_0} + 1$ and $\gamma'_{y_i} = \gamma_{y_i}$ for each $1 \leq i \leq k$. On the other hand, according to the updating algorithm above, a'_{ij} are updated for each $0 \leq i, j \leq k$ as follows, depending on the case. There are four subcases:

- If $i > 0$ and $j > 0$, then $\gamma'_{y_i} = \gamma_{y_i}$, $\gamma'_{y_j} = \gamma_{y_j}$, $a'_{ij} = a_{ij}$. The claim (*) holds trivially.
- If $i = 0$ and $j > 0$, then $\gamma'_{y_i} = \gamma_{y_i} + 1$, $\gamma'_{y_j} = \gamma_{y_j}$, $a'_{ij} = a_{ij}$ ADD 1. Since y_0 is the only now-clock, all $\gamma_{y_i} - \gamma_{y_j}$, $\gamma'_{y_i} - \gamma'_{y_j}$, a'_{ij} and a_{ij} are nonnegative. It suffices to show for any $c \geq 0, c \in [m-1]$, the claim holds. In fact, $\gamma'_{y_i} - \gamma'_{y_j} \# c$ iff $\gamma_{y_i} - \gamma_{y_j} \# c - 1$ iff $a_{ij} \# c - 1$ iff $a_{ij} + 1 \# c$. Also, $a_{ij} + 1 \# c$ iff $a'_{ij} \# c$, by separating the cases for $a_{ij} = m$ and $a_{ij} < m$, and noticing that $c < m$. Thus, (*) holds, i.e., $\gamma'_{y_i} - \gamma'_{y_j} \# c$ iff $a'_{ij} \# c$.
- If $i > 0$ and $j = 0$, similar as above.
- If $i = 0$ and $j = 0$, the Claim (*) holds trivially.

Noticing that under the assignment $y_0 := y_0 + 1$, $b'_{ij} := a'_{0j}$. Thus, (**) can be shown using (*).

When the assignment is in the form of $y_i := y_0$ for $y_i \in \tau \subseteq \{y_1, \cdots, y_k\}$, (note that in this case, the now clock does not progress, i.e., $\gamma'_{y_0} = \gamma_{y_0}$) there are four cases to consider in order to show (*) for all $0 \leq i, j \leq k$,

- If $i, j \in \tau$, then $\gamma'_{y_i} = \gamma'_{y_j} = \gamma_{y_0}$, $a'_{ij} = 0$ and therefore $\gamma'_{y_i} - \gamma'_{y_j} = a'_{ij} = 0$. Thus, the Claim (*) trivially holds.
- If $i \in \tau$ and $j \notin \tau$, then, $\gamma'_{y_i} = \gamma_{y_0}$, $\gamma'_{y_j} = \gamma_{y_j}$, $a'_{ij} = b_{ij}$. Thus, for each $c \in [m-1]$, $\gamma'_{y_i} - \gamma'_{y_j} \# c$ iff $\gamma_{y_0} - \gamma_{y_j} \# c$ iff (induction hypothesis) $b_{ij} \# c$ iff $a'_{ij} \# c$. The Claim (*) holds.
- If $i \notin \tau$ and $j \in \tau$, similar as above.
- If $i \notin \tau$ and $j \notin \tau$, then, $\gamma'_{y_i} = \gamma_{y_i}$, $\gamma'_{y_j} = \gamma_{y_j}$, and $a'_{ij} = a_{ij}$. Thus, the Claim (*) holds trivially.

Now we prove Claim (**) under this assignment for τ. Again, there are four cases to consider:

- If $i, j \in \tau$, then $b'_{ij} = 0$, noticing that $\gamma'_{y_j} = \gamma_{y_0}$ and $\gamma'_{y_0} = \gamma_{y_0}$, Claim (**) holds.
- If $i \in \tau$ and $j \notin \tau$, then, $b'_{ij} = a'_{0j}$. Claim (**) holds directly from Claim (*).
- If $i \notin \tau$ and $j \in \tau$, similar as above.
- If $i \notin \tau$ and $j \notin \tau$, then, $b'_{ij} = b_{ij}$. In fact, $\gamma'_{y_0} = \gamma_{y_0}$, $\gamma'_{y_j} = \gamma_{y_j}$. Thus, Claim (**) holds directly from the induction hypothesis.

This ends the proof of the Claim. Thus, it is valid for M to use $a_{ij}\#c$ to do each test $y_i - y_j \# c$. At some point, M guesses that it has reached the configuration $\beta^{\mathcal{A}'}$ by comparing the counter values and the stack content with $\beta^{\mathcal{A}'}$ through reading the rest of the input tape. M accepts iff such a comparison succeeds. Clearly M accepts $\rightsquigarrow^{\mathcal{A}'}$. There is a slight problem when M compares its own stack content with the one $\beta^{\mathcal{A}'}_w$ on the one-way input tape in $\beta^{\mathcal{A}'}$ by popping the stack. The reason is that popping the stack contents reads the reverse of the stack content. However, recall that the encoding of the stack word $\beta^{\mathcal{A}'}_w$ on the input tape is reversed. Thus, such a comparison can be proceeded. □

Assignments in M constructed in the above proof, in the form of, $y_0 := y_0 + 1$ and $y_i := y_0$ with $1 \leq i \leq k$, are still not standard. We will now show that these assignments can be made standard, while the machine is still reversal-bounded. Let M' be an NPCM that is exactly the same as M. M' simulates M's computation from the configuration $\alpha^{\mathcal{A}'}$. Initially, each $y_i := \alpha^{\mathcal{A}'}_{y_i}$ as we indicated in the above proof. However, each time that M executes an assignment $y_0 := y_0 + 1$, M' increases **all** the counters by 1, i.e., $y_i := y_i + 1$ for each $0 \leq i \leq k$. When M executes an assignment $y_i := y_0$, M' does nothing. The stack operations in M are faithfully simulated by M' on its own stack. For each $1 \leq i \leq k$, at some point, either initially or at the moment $y_i := y_0$ is being executed by M, M' guesses (only once for each i) that y_i has already reached the value given in $\beta^{\mathcal{A}'}$. After such a guess for i, an execution of $y_0 := y_0 + 1$ will not cause $y_i := y_i + 1$ as indicated above (i.e., y_i will no longer be incremented). However, after such a guess for i, a later execution of $y_i := y_0$ in M will cause M' to abort abnormally (without accepting the input). At some point after **all** $1 \leq i \leq k$ have been guessed, M' guesses that it has reached the configuration $\beta^{\mathcal{A}'}$. Then, M' compares its current configuration with the one on the rest of the input tape $\beta^{\mathcal{A}'}$ (recall that the stack word in $\beta^{\mathcal{A}'}$ is reversed on the input tape.). M' accepts iff such a comparison succeeds. Clearly, M' uses only assignments $y_0 := y_0 + 1$ and $y_i := y_i + 1$ for $1 \leq i \leq k$. Thus, M' is also reversal-bounded and accepts $\rightsquigarrow^{\mathcal{A}'}$. Therefore,

Theorem 7. *The binary reachability $\rightsquigarrow^{\mathcal{A}'}$ of $\mathcal{A}'$ can be accepted by a reversal-bounded NPCM using standard tests and assignments.*

Combining the above theorem with Theorem 5 and noticing that v in Theorem 5 can be guessed, it follows immediately that,

Theorem 8. *The binary reachability of a discrete pushdown timed automaton can be accepted by a reversal-bounded NPCM using standard tests and assignments.*

A discrete timed automaton is a special case of a discrete pushdown timed automaton without the pushdown stack. The above proofs still work for discrete timed automata without considering stack operations. That is,

Theorem 9. *The binary reachability of a discrete timed automaton can be accepted by a reversal-bounded multicounter machine using standard tests and assignments.*

Combining the above theorem and Theorem 2, it is immediate that the binary reachability of a discrete timed automaton is Presburger over clocks. This result shadows the result in [10] that the binary reachability of a timed automaton with real-valued clocks is expressible in the additive theory of reals. However, our proof is totally different from the flattening technique in [10].

4 Verification Results

The importance of the characterization of $\rightsquigarrow^{\mathcal{A}}$ for a discrete pushdown timed automaton $\mathcal{A}$ is that the emptiness of reversal-bounded NPCMs is decidable from Theorem 1. In this section, we will formulate a number of properties that can be verified for discrete pushdown timed automata.

We first need some notation. We use $\alpha, \beta \cdots$ to denote variables ranging over configurations. We use $\mathbf{q}, \mathbf{x}, \mathbf{w}$ to denote variables ranging over control states, clock values and stack words respectively. Note that α_{x_i}, α_q and α_w are still used to denote the value of clock x_i, the control state and the stack word of α. We use a count variable $\#_a(\mathbf{w})$ to denote the number of occurrences of a character $a \in \Gamma$ in a stack word variable $\mathbf{w}$. An NPCM-term t is defined as follows:[5]

$$t ::= n \mid \mathbf{q} \mid \mathbf{x} \mid \#_a(\alpha_w) \mid \alpha_{x_i} \mid \alpha_q \mid t - t \mid t + t$$

where n is an integer and $a \in \Gamma$. An NPCM-formula f is defined as follows:

$$f ::= t > 0 \mid t \bmod n = 0 \mid \neg f \mid f \vee f$$

where $n \neq 0$ is an integer[6]. Thus, f is a Presburger formula over control state variables, clock value variables and count variables. Let F be a formula in the following format:

$$\bigvee (f_i \wedge \ \alpha^i \rightsquigarrow \beta^i)$$

[5] Control states can be interpreted over a bounded range of integers. In this way, an arithmetic operation on control states is well-defined.

[6] We use $(i, j, \mathbf{w})$ to denote the i-th character of $\mathbf{w}$ is the j-th symbol in Γ. Then, Theorem 10 is still correct when we include (i, j, α_w) as an atomic formula in an NPCM-formula with $i, j \in \mathbf{Z}^+$.

where f_i, α^i and β^i are a number of NPCM-formulas and configuration variables. Write $\exists F$ to be a closed formula such that each free variable in F is existentially quantified. Then, the property $\exists F$ can be verified. [7]

Theorem 10. *The truth value of $\exists F$ with respect to a discrete pushdown timed automaton $\mathcal{A}$ for any NPCM-formula F is decidable.*

Proof. Consider each disjunctive subformula $f_i \wedge \alpha^i \rightsquigarrow \beta^i$ in F. Since f_i is Presburger, (the domain of) f_i can be accepted by a NCM M_{f_i} from Theorem 2, and further from Theorem 3, since the domain of f_i can be encoded as a set of integer tuples (thus, bounded), M_{f_i} can be made deterministic. From Theorem 8, we can construct a reversal bounded NPCM accepting (the domain of) $\alpha^i \rightsquigarrow \beta^i$. It is obvious that $f_i \wedge \alpha^i \rightsquigarrow \beta^i$ can be accepted by a reversal-bounded NPCM M_i by "intersecting" the two machines. Now, F is a union of M_{f_i}. Since reversal-bounded NPCMs are closed under union [15], we can construct a reversal-bounded M to accept F. Since $\exists F = false$ is equivalent to testing the emptiness of M, the theorem follows from Theorem 1. □

Although $\exists F$ cannot be used to specify liveness properties, it can be used to specify many interesting safety properties. For instance, the following property:

"for any configurations α and β with $\alpha \rightsquigarrow^{\mathcal{A}} \beta$, clock x_2 in β is the sum of clocks x_1 and x_2 in α, and symbol a appears in β twice as many times as symbol b does in α."

This property can be expressed as, $\forall\alpha\forall\beta(\alpha \rightsquigarrow^{\mathcal{A}} \beta \rightarrow (\beta_{x_2} = \alpha_{x_1} + \alpha_{x_2} \wedge \#_a(\beta_w) = 2\#_b(\alpha_w)))$. The negation of this property is equivalent to $\exists F$ for some F. Thus, it can be verified. We also need to point out that

- Even without clocks, $\#_a(\beta_w) = 2\#_b(\alpha_w)$ indicate a nonregular set of stack word pairs. Thus, this property cannot be verified by the model checking procedures for pushdown systems [4, 12, 17],
- Even without the pushdown stack, $\beta_{x_2} = \alpha_{x_1} + \alpha_{x_2}$ is not a clock region [2]. Thus, the classical region techniques (include [5] for Pushdown Timed Systems) can not verify this property. This is also pointed out in [10].

Note that in an NPCM-formula, the use of a stack word is limited to count the occurrences of a symbol, e.g., $\#_a(\alpha_w)$. In fact we can have the following more general use. Given P and I, two sets of configurations of a discrete pushdown timed automaton $\mathcal{A}$. If, starting from a configuration in I, $\mathcal{A}$ can only reach configurations in P, then P is a *safety property* with respect to the initial condition I. The *safety analysis problem* is whether P is a safety property with respect to the initial condition I, given P and I. The following theorem asserts that the safety analysis problem is decidable for discrete pushdown timed automata,

[7] If stack words in α^i and β^i are bounded, i.e., in the form of $w_1^* \cdots w_k^*$ with $w_1, \cdots, w_k$ fixed, then F can be further extended to allow disjunctions *and* conjunctions over $(f_i \wedge \alpha^i \rightsquigarrow \beta^i)$ formulas and the following theorem still holds. The reason is that reversal-bounded NPCMs are closed under conjunction when languages are bounded [15].

when both P and I are bounded languages and are accepted by reversal-bounded NCMs.

Theorem 11. *The safety analysis problem is decidable for discrete pushdown timed automata where both the safety property and the initial condition are bounded languages and accepted by reversal-bounded nondeterministic multi-counter machines.*

Proof. Let $\mathcal{A}$ be a discrete pushdown timed automaton. Let P and I be accepted by reversal-bounded NCMs M_P and M_I, respectively. Note that P is a safety property with respect to the initial condition I iff $I \cap Pre^*(\neg P) = \emptyset$. That is, if P is a safety property, then, starting from a configuration, $\mathcal{A}$ can not reach a configuration that is in the complement $\neg P$ of P. From Theorem 3, since P and I are bounded, both M_P and M_I can be made deterministic. Thus, $\neg P$ can also be accepted by a reversal-bounded NCM. Therefore, from Theorem 8 and Theorem 4, we can construct a reversal-bounded NPCM M_{Pre^*} accepting $Pre^*(\neg P)$. It is obvious that $I \cap Pre^*(\neg P)$ can also be accepted by a reversal-bounded NPCM M' by "intersecting" M_I and M_{Pre^*}. The theorem follows by noticing that $I \cap Pre^*(\neg P) = \emptyset$, i.e., testing the emptiness of M' is decidable from Theorem 1. □

Thus, from the above theorem, the following property can be verified:

"starting from a configuration α with the stack word $a^n b^{2n}$ for some n, $\mathcal{A}$ can only reach a configuration β satisfying: the clock x_1 in β is the sum of clock x_2 and x_3 in β, and the stack word is $a^n b^{n+2m}$ for some n and m."
The reason is that $a^n b^{2n}$ and $a^n b^{n+2m}$ can be encoded as Presburger tuples (thus bounded). Therefore they can be accepted by reversal-bounded NCMs.

Now let's look at $\mathcal{A}$ without a pushdown stack, i.e., a discrete timed automaton. Obviously, the above two theorems still hold for such $\mathcal{A}$. However, since $\rightsquigarrow^{\mathcal{A}}$ now is Presburger from Theorem 9, we can do more. An NCM-term t is defined as follows:

$$t ::= n \mid \mathbf{q} \mid \mathbf{x} \mid \alpha_{x_i} \mid \alpha_q \mid t - t \mid t + t$$

where n is an integer and $a \in \Gamma$. An NCM-formula f is defined as follows:

$$f ::= t > 0 \mid \neg f \mid f \vee f \mid \alpha \rightsquigarrow^{\mathcal{A}} \beta \mid \forall \alpha(f) \mid \forall \mathbf{x}(f) \mid \forall \mathbf{q}(f).$$

Thus, f is a Presburger formula over control state variables, clock value variables and configuration variables. Thus, if f is closed (i.e., without free variables), then the truth value of f is decidable since f is Presburger. Thus, a property formulated as a closed NCM-formula can be verified. Even clocks are real values, this is still true from [10]. The following is an example property which can be verified:

"for all configuration α there exists a configuration β such that $\alpha \rightsquigarrow^{\mathcal{A}} \beta$ and the clock x_1 in β is the sum of clocks x_1 and x_2 in α."
This property can be written in the following closed NCM-formula,

$$\forall \alpha \exists \beta (\alpha \rightsquigarrow^{\mathcal{A}} \beta \ \wedge \ \beta_{x_1} = \alpha_{x_1} + \alpha_{x_2}).$$

NCM-formulas can be extended by considering event labels as follows. Consider a discrete timed automata $\mathcal{A}$. Recall that an edge in $\mathcal{A}$ does not have a label. Now we assume that, just as a standard timed automaton, each edge in $\mathcal{A}$ is labeled by a letter in a finite alphabet Γ. Denote $R(\alpha, \beta, \mathbf{n}_\Gamma)$ to be a predicate meaning α can reach β through a path that for each $a \in \Gamma$, the number of occurrence of label a in the path is equal to $\mathbf{n}_a$ with $\mathbf{n}_\Gamma$ being an array of $\mathbf{n}_a$ for $a \in \Gamma$. By introducing a new counter for each $a \in \Gamma$ and increasing the counter whenever M executes a transition labeled by a, we can construct a reversal-bounded NCM M as in the proof of Theorem 9 (and the actual proof is in Theorem 8.). From Theorem 2, we have,

Theorem 12. *R is Presburger.*

Thus, we can add atomic terms $\mathbf{n}_a$ for $a \in \Gamma$ and an atomic formula $R(\alpha, \beta, \mathbf{n}_\Gamma)$ to the above definition of NCM-formulas. Then such closed NCM-formulas can be verified for discrete timed automata with labels. The following is an example property:

"For any configuration α there exists a configuration β such that if the clock x_1 in β is the sum of clocks x_1 and x_2 in α, then α can reach β through a path with the number of transitions labeled by a being twice the number of transitions labeled by b."

This property can be written in a closed NCM-formula,

$$\forall\alpha\exists\beta(\beta_{x_1} = \alpha_{x_1} + \alpha_{x_2} \rightarrow \exists\mathbf{n}_a\exists\mathbf{n}_b(R(\alpha, \beta, \mathbf{n}_a, \mathbf{n}_b) \wedge \mathbf{n}_a = 2\mathbf{n}_b)).$$

Thus, it can be verified.

5 Conclusion

We consider discrete pushdown timed automata that are timed automata with integer-valued clocks augmented with a pushdown stack. Using a pure automata-theoretic approach, we show that the binary reachability can be accepted by a reversal-bounded nondeterministic pushdown multicounter machine. The proof reveals that, by replacing enabling conditions with a finite table, the control part (testing clock constraints) and the clock behaviors (clock progresses and resets) can be separated for a discrete (pushdown) timed automaton, while maintaining the structure of the automaton. By using the known fact that the emptiness problem for reversal-bounded nondeterministic pushdown multicounter machines is decidable, we show a number of properties that can be verified.

Binary reachability characterization is a fundamental step towards a more general model-checking procedure for discrete pushdown timed automata. It is immediate to see that the region reachability in [2] can be similarly formulated by using Theorem 11, as long as stack words are regular. Thus, we can use the idea in [4] as well as in [12] to demonstrate a subset of μ-calculus that has a decidable decision procedure for a class of timed pushdown processes. We plan to investigate this issue in future work. In the future we would also like to

investigate the complexity of the verification procedures we have developed in this paper. The techniques in this paper will be used in the implementation of a symbolic model checker for real-time specifications written in the specification language ASTRAL [8].

Thanks to anonymous reviewers for a number of useful suggestions.

References

1. R. Alur, C. Courcoibetis, and D. Dill, *"Model-checking in dense real time," Information and Computation*, **104** (1993) 2-34
2. R. Alur and D. Dill, *"Automata for modeling real-time systems," Theoretical Computer Science*, **126** (1994) 183-236
3. R. Alur, T. A. Henzinger, *"A really temporal logic," J. ACM*, **41** (1994) 181-204
4. A. Bouajjani, J. Esparza, and O. Maler, *"Reachability Analysis of Pushdown Automata: Application to Model-Checking,"*, In *CONCUR'97*, LNCS 1243, pp. 135-150
5. A. Bouajjani, R. Echahed, and R. Robbana, *"On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures,"*, In *Hybrid System II*, LNCS 999, 1995, pp. 64-85
6. G. Behrmann, K. G. Larsen, J. Pearson, C. Weise and W. Yi, *"Efficient timed reachability analysis using clock difference diagrams,"*, In *CAV'99*, LNCS 1633, pp. 341-353
7. B. Boigelot and P. Wolper, *"Symbolic verification with periodic sets,"* In *CAV'94*, LNCS 818, pp. 55-67
8. A. Coen-Porisini, C. Ghezzi and R. Kemmerer, *"Specification of real-time systems using ASTRAL," IEEE Transactions on Software Engineering*, **23** (1997) 572-598
9. H. Comon and Y. Jurski, *"Multiple counters automata, safety analysis and Presburger arithmetic,"* In *CAV'98*, LNCS 1427, pp. 268-279.
10. H. Comon and Y. Jurski, *"Timed Automata and the Theory of Real Numbers,"* In *CONCUR'99*, LNCS 1664, pp. 242-257
11. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su *"Safety property analysis of reversal-bounded pushdown multicounter machines,"*, manuscript, 1999
12. A. Finkel, B. Willems and P. Wolper *"A direct symbolic approach to model checking pushdown systems,"* In *INFINITY'97*.
13. E. Gurari and O. Ibarra, *"The Complexity of Decision Problems for Finite-Turn Multicounter Machines," J. Computer and System Sciences*, **22** (1981) 220-229
14. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. *"Symbolic Model Checking for Real-time Systems," Information and Computation*, **111** (1994) 193-244
15. O. H. Ibarra, *"Reversal-bounded multicounter machines and their decision problems," J. ACM*, **25** (1978) 116-133
16. K. L. McMillan. *"Symbolic model checking - an approach to the state explosion problem,"* PhD thesis, Carnegie Mellon University, 1992.
17. I. Walukiewicz, *"Pushdown processes: games and model checking,"* In *CAV'96*, LNCS 1102, pp. 62-74
18. F. Wang, *"Efficient Data Structure for Fully Symbolic Verification of Real-Time Software Systems ,"* In *TACAS'00*, to appear.

This article was processed using the LaTeX macro package with LLNCS style