On Removing the Pushdown Stack in Reachability Constructions [⋆]

Oscar H. Ibarra[1] and Zhe Dang[2]

[1]Department of Computer Science
University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu

[2]School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164, USA
zdang@eecs.wsu.edu

Abstract. A discrete pushdown timed automaton is a pushdown machine with integer-valued clocks. It has been shown recently that the binary reachability of a discrete pushdown timed automaton can be accepted by a 2-tape pushdown acceptor with reversal-bounded counters. We improve this result by showing that the stack can be removed from the acceptor, i.e., the binary reachability can be accepted by a 2-tape finite-state acceptor with reversal-bounded counters. We also obtain similar results for more general machine models. Our characterizations can be used to verify certain properties concerning these machines that were not verifiable before using previous techniques. We are also able to formulate a subset of Presburger LTL that is decidable for satisfiability-checking with respect to these machines.

1 Introduction

Developing verification techniques for infinite-state systems is an important ongoing effort, motivated to a large extent by the successes of model-checking techniques for finite-state systems [23]. Unlike for finite-state systems, there is a decidability boundary for infinite-state systems: machines with two counters (i.e., "Minsky machines") are Turing complete. Therefore, we must seek a balance between the computing power of infinite-state systems and their decidability.

Many infinite-state models have been shown decidable for various model-checking problems. These models include timed automata [1], pushdown automata and pushdown processes [3, 14, 12], various versions of counter machines [6, 9, 11, 13, 22], and various queue machines [2, 4, 19, 20, 24].

Pushdown systems are of particular interest, since, in practice, they are related to pushdown processes and, in theory, they are well studied in automata theory. A pushdown machine can be obtained by augmenting a finite-state machine with a pushdown stack. A configuration of a pushdown machine without an input tape (PM), is a string $\alpha = wq$, where w is the stack content and q is the state (we assume that the stack alphabet is disjoint from the state set). If M is a PM and S is a set of configurations, define the

[⋆] Supported in part by NSF grant IRI-9700370

backward and forward reachability sets of M with respect to S by: $pre^*(M,S) = \{\alpha|$ configuration α can reach some configuration in $S\}$ and $post^*(M,S) = \{\alpha|$ configuration α is reachable from some configuration in $S\}$. It is known that if S is regular, then $pre^*(M,S)$ and $post^*(M,S)$ are also regular (see, e.g., [3, 12, 14]). One can also show that the binary reachability of M, $Binary(M) = \{(\alpha,\beta)|\beta$ is reachable from $\alpha\}$, can be accepted by a 2-tape FA, i.e., a finite-state acceptor with two one-way input tapes. (Note that a 1-tape FA is the usual finite automaton.)

A PM augmented with finitely many real-valued clocks is called a pushdown timed automaton, which is a generalization of a timed automaton [1]. It is discrete if the clocks can only assume nonnegative integer values (definitions are in Section 4). A configuration now includes the clock values written in unary. It is easy to show that in general, the binary reachability of a (discrete) pushdown timed automaton cannot be accepted by a 2-tape FA. Characterizations of the binary reachability of pushdown timed automata with discrete or real-valued clocks have recently been given in [10, 8]. In particular, it was shown in [10] (see also [21]) that the binary reachability of a discrete pushdown timed automaton can be accepted by a 2-tape pushdown acceptor augmented with reversal-bounded counters. A counter (which, we assume w.l.o.g., can only store nonnegative integers, since the sign can be remembered in the states) is reversal-bounded if it can be tested for zero and can be incremented or decremented by one, but the number of alternations between nondecreasing mode and nonincreasing mode in any computation is bounded by a given constant; e.g., a counter whose values change according to the pattern $0\ 1\ 1\ 2\ 3\ 4\ \underline{4\ 3}\ 2\ 1\ \underline{0\ 1}\ \underline{1\ 0}$ is 3-reversal, where the reversals are underlined. It follows that the backward and forward reachability sets can be accepted by (1-tape) pushdown acceptors with reversal-bounded counters. These results and the fact that the emptiness problem for multitape pushdown acceptors with reversal-bounded counters is decidable [16] have been used recently to prove the decidability of certain verification problems for infinite-state transition systems [10, 17, 18, 21, 22].

In this paper, we improve the above results by showing that the pushdown stack can be *removed* from the acceptors. Specifically, we show that the binary (backward or forward) reachability can be accepted by a 2-tape (1-tape) finite-state acceptor with reversal-bounded counters. In fact, we show that the results hold, even if the discrete pushdown timed automaton is augmented with reversal-bounded counters. Note that equipping the pushdown timed automaton with counters is an important and nontrivial generalization, since it is known that reversal-bounded counters can "verify" Presburger relations on clock values [16].

The results in this paper can be used to verify properties that were not verifiable before using previous techniques. For example, consider the set $W = pre^*(M_1, pre^*(M_2, R_2) \cap R_1)$, where M_1 and M_2 are discrete pushdown timed automata with the same state set, the same pushdown alphabet, and the same clock names, and R_1 and R_2 are two sets of configurations accepted by finite-state acceptors augmented with reversal-bounded counters. We know from [10] that $pre^*(M_2, R_2)$ can be accepted by a pushdown acceptor with reversal-bounded counters. Without using our current result, a direct construction of a machine accepting W would require two stacks: one for the machine accepting $pre^*(M_2, R_2)$, and the other is for M_1 itself. This seems to show that the emptiness of W may be undecidable, since a 2-stack machine is equivalent to a Turing machine. However, it follows from our results that W can be accepted by a finite-state acceptor with reversal-bounded counters; hence, the emptiness of W is decidable. As an application, consider the satisfiability-checking (the dual of model-checking) of a property $\diamond(P_1 \wedge \diamond P_2)$ concerning a discrete pushdown timed automaton with reversal-bounded counters M, where P_1 and P_2 are Presburger formulas on stack symbol counts and clock and counter values. This problem is reducible to checking the emptiness of $pre^*(M, pre^*(M, P_2) \cap P_1)$, which we now know is decidable.

We also look at discrete timed automata with clocks, reversal-bounded counters, and a read/write worktape (instead of a pushdown stack), but restricted to be *finite-crossing*, i.e., in any computation, the number of times the read/write head crosses the boundary between any two adjacent worktape cells is bounded by a given constant. We show that the binary (backward or forward) reachability set of this machine can also be accepted by a 2-tape (1-tape) finite-state acceptor with reversal-bounded counters. This improves the corresponding results in [18] where the the acceptors needed a finite-crossing read/write tape. Note that without without the "finite-crossing" requirement, the model becomes a Turing machine.

We will use the following notation. We use the suffix 'M' to indicate that the model has no input tape and 'A' when the model has one-way input tape(s). All models are nondeterministic.

1. PM: Pushdown machine
2. PA: Pushdown acceptor
3. PCM: Pushdown machine with reversal-bounded counters
4. PCA: Pushdown acceptor with reversal-bounded counters
5. FM: Finite-state machine
6. FA: Finite-state acceptor
7. CM: Finite-state machine with reversal-bounded counters
8. CA: Finite-state acceptor with reversal-bounded counters
9. WCM: Finite-state machine with a read/write worktape and reversal-bounded counters
10. WCA: Finite-state acceptor with a read/write worktape and reversal-bounded counters
11. k-tape PCA (FA, CA, WCA) is a PCA (FA, CA, WCA) with k input tapes, with one head per tape. A 1-tape PCA (FA, ...) will simply be referred to as a PCA (FA, ...)
12. PTCM (WTCM) is a PCM (WCM) augmented with discrete clocks.

The paper has four sections in addition to this section. Section 2 shows that the binary reachability of a PCM can be accepted by a 2-tape CA and that the backward and forward reachability sets can be accepted by CAs. Section 3 shows that these results hold for finite-crossing WCMs. Section 4 generalizes the results to PCMs and finite-crossing WCMs with "clocks" (i.e., the timed versions of the models). Finally, Section 5 proposes a subset of Presburger LTL whose satisfiability-checking is decidable.

2 PCMs

We first look at the simple case of a PM (pushdown machine without counters). We assume that the pushdown stack has a "bottom" symbol B_0, and is associated with two kinds of stack operations: push(q, Z, q'), i.e., push symbol Z onto the stack and switch from state q to state q', and pop(q, Z, q'), i.e., pop the top Z from the stack and switch from state q to state q'. Replacing the top symbol of the stack with another symbol can be implemented by a push followed by a pop.

Let M be a PM. Define predicates push^* and pop^* as follows: $\text{push}^*(q, T, Z, q')$ is true, if there is a sequence of moves of M such that, starting from state q with stack top symbol T, M does not pop this T, and the last move is a push of Z on top of T ending in state q' (Notice that, prior to this last move, the sequence may involve many pushes/pops.) Similarly, $\text{pop}^*(q, Z, T, q')$ is true, if there is a sequence of moves of M such that, starting from state q with stack top symbol T (and Z the symbol directly under T), M does not pop this Z and the result of the moves makes this Z the top of the stack and state q'. We also define $\text{stay}^*(q, T, q')$ to be true if M, starting from state q with stack top symbol T, can reach state q' without performing any stack operations.

Lemma 1. *Given a PM M, we can effectively compute the predicates* push^*, pop^*, *and* stay^*.

Define $Binary(M, S, T) = \{(\alpha, \beta) |$ configuration $\alpha \in S$ can reach configuration $\beta \in T\}$. When $S = T =$ the sets of all valid configurations, $Binary(M, S, T)$ will simply be written $Binary(M)$.

Theorem 1. *$Binary(M)$ of a PM M can be accepted by a 2-tape FA.*

Corollary 1. *Let M be a PM, and, S and T be sets of configurations of M accepted by FAs. Then $Binary(M, S, T)$ can be accepted by a 2-tape FA.*

The backward and forward reachability sets of a PM M with respect to a regular set of configurations S is regular [3, 14, 12]. This result is easily obtained from the corollary above.

Corollary 2. *Let M be a PM and S be a set of configurations of M accepted by an FA A_S. Then pre$^*(M, S) = \{\alpha |$ configuration α can reach some configuration β in $S\}$ and post$^*(M, S) = \{\beta |$ configuration β is reachable from some configuration α in $S\}$ can be accepted by FAs.*

We now consider the PCMs. The reversal-bounded counters in the PCMs complicate the constructions, since now we have to incorporate counters into the acceptors of the reachability sets. We need to prove some intermediate results.

Let $\mathbb{N}$ be the set of nonnegative integers and n be a positive integer. A subset S of $\mathbb{N}^n$ is a *linear set* if there exist vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, ..., \boldsymbol{v}_t$ in $\mathbb{N}^n$ such that

$$S = \{\boldsymbol{v} \mid \boldsymbol{v} = \boldsymbol{v}_0 + a_1\boldsymbol{v}_1 + \cdots + a_t\boldsymbol{v}_t, \forall 1 \leqslant i \leqslant t, a_i \in \mathbb{N}\}.$$

The vectors $\boldsymbol{v}_0$ (the constant vector) and $\boldsymbol{v}_1, ..., \boldsymbol{v}_t$ (the periods) are called *generators*. A set S is *semilinear* if it is a finite union of linear sets. Semilinear sets are precisely the sets definable by *Presburger formulas*[15].

There is a simple automaton that characterizes semilinear sets. Let M be a nondeterministic finite-state machine (*without* an input tape) with n counters for some $n \geqslant 1$.

The computation of M starts with all the counters zero and the automaton in the start state. An atomic move of M consists of incrementing at most one counter by 1 and changing the state (decrements are not allowed). An n-tuple $\boldsymbol{v} = (i_1, \ldots, i_n) \in \mathbb{N}^n$ is generated by M if M, when started from its initial configuration, halts in an accepting state with $\boldsymbol{v}$ as the contents of the counters. The set of all n-tuples generated by M is denoted by $G(M)$. We call this machine a C-generator. If the C-generator is augmented with a pushdown stack, the machine is called a PC-generator. Notice that counters in a generator are nondecreasing, i.e., 0-reversal-bounded. We will need the following lemma, which can easily be shown using the results in [16].

Lemma 2. *The following statements are equivalent for $S \subseteq \mathbb{N}^n$: a) S is a semilinear set; b) S can be generated by a C-generator; c) S can be generated by a PC-generator.*

Consider a PCM M with n counters. A configuration of M is now represented as a string $\alpha = wqd_1^{x_1}d_2^{x_2}...d_n^{x_n}$, where w is the stack content, q is the state, $d_1, ..., d_n$ are distinct symbols, and $x_1, x_2, ..., x_n$ are the values of the counters (thus the counter values are represented in unary). We will show that the binary reachability $Binary(M)$ can be accepted by a 2-tape CA.

To simplify matters, we convert M to another PCM M' with many more counters than M. Assume M starts from a configuration α and reaches another configuration β. M' operates like M, except that the counters can make at most one reversal. M' simulates M faithfully, except that when a counter c of M makes a reversal from nonincreasing to increasing or c starts to increment before any decrements were made after starting from configuration α, M' suspends the simulation but continues decreasing this counter to zero while simultaneously increasing a new counter c' (starting at zero). When c reaches zero, c' has the old value of c before the simulation was suspended. M' then resumes the simulation with c' taking the role of c. If c' later reverses from nonincreasing to increasing, a new counter c'' is deployed like before. In this way, each counter c of M making r reversals can be replaced by $(r+1)/2$ counters $c_1, ..., c_{(r+1)/2}$, where each one makes at most one reversal. Moreover, a configuration α of M translates to a corresponding configuration of M', where the value of a counter c of M is identified with the value of one of the counters $c_1, ..., c_{(r+1)/2}$. Clearly, if we can construct a 2-tape CA A' to accept $Binary(M')$, we can modify A' to a 2-tape CA A to accept $Binary(M)$.

Let M be a PCM. From the discussion above, we assume that the counters have been normalized, i.e., during the computation from one configuration to another, each counter c behaves as one of the following five patterns:

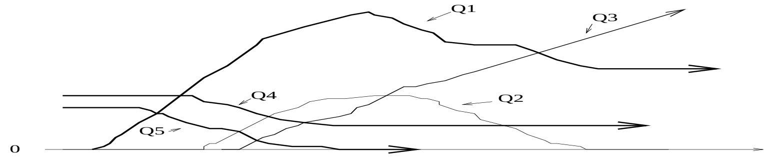

Fig. 1. Behavior patterns for a normalized counter

- c starts at zero, becomes positive, reverses (i.e., decrements), but remains positive.
- c starts at zero, becomes positive, reverses, becomes zero (and remains zero).
- c starts at zero, becomes positive, and does not reverse.
- c starts at a positive value, remains nonincreasing and positive.
- c starts at a positive value, remains nonincreasing, becomes zero (and remains zero).

We do not include the case when a counter remains at zero during the entire computation, since this can be simulated by an increment by 1 followed by a decrement by 1. Call the behaviors above Q_1, Q_2, Q_3, Q_4 and Q_5, respectively.

Consider a counter c that has behavior Q_1. During the computation, c makes a mode change at three different instances: when it started at 0, became positive, and when it reversed. We denote these instances by $0, +, rev$. Note that c is positive at the end of the computation, since it has behavior Q_1. In the construction to be described below, c will be simulated by two counters, c^+ and c^-, the first to record increments and the second to record decrements. If c is tested for zero during any segment of the simulation, the simulator assumes that it is zero before the mode changes to $+$ and positive after the mode has changed to $+$. (Note that the simulator knows when the mode changes.) At the end of the entire simulation, the simulator verifies that c is indeed positive by checking that $c^+ - c^-$ is positive.

Similarly, for a counter c with behavior Q_2, the mode-change instances are: $0, +,$ $rev, zero$. As in the above case, c will be simulated by two counters c^+ and c^-, and the simulator's action when testing for zero is like in the above case before the mode changes to $zero$. The point when the counter becomes zero (i.e. the mode changes to $zero$) is "guessed" by the simulator. After the mode has changed to $zero$, the simulator assumes that the counter is always zero when it is being tested (and c^+ and c^- will remain the same in the rest of the computation). At the end of the simulation, the simulator verifies that c is zero by checking that $c^+ = c^-$.

For the case of a counter c with behavior Q_3, the mode-change instances are $0, +$. Like in the case for Q1, the simulator assumes the counter is zero before the mode changes to $+$ and positive after the mode has changed to $+$. Then c^+ is exactly c, and c^- will remain zero during the entire simulation. Note that for this case, there is nothing to verify at the end of the simulation.

For the case for Q_4, the mode-change instance is rev. Counter c stays positive and c^+ will remain zero during the entire computation. The simulator checks that c^- is less than the starting value of c.

For the case of a counter c with behavior Q_5, the mode-change instances are rev, $zero$. The simulator assumes the counter is positive before the mode changes to $zero$. Notice that the point that c becomes zero can be guessed by the simulator as described in the case for Q2. c^+ will remain zero during the entire simulation. Then the simulator checks that c^- is exactly the starting value of c.

When we say that a counter starts with mode m and ends with mode m' in a certain segment of the computation, we mean: 1. The counter is already in mode m, i.e., the mode-change to m has already been executed earlier. 2. If $m' \neq m$, the mode-change to m' occurs during the segment of computation under consideration.

In describing a subcomputation of the machine, we refer to $\langle c, Q_i, m, m' \rangle$ as a mode vector for c, and this means that counter c has behavior Q_i ($i = 1, 2, 3, 4, 5$)

and in the subcomputation, c starts with mode m and ends with mode m'. We denote $\langle c, Q_i, m, m' \rangle$ simply as $V(c, Q_i)$, when m and m' are understood.

Let M be a PCM with n counters: $c_1, ..., c_n$. We associate with each counter c two counters c^+ and c^-. Given $Q_{i_1}, ..., Q_{i_n}, q, Z, q'$ (each $i_j \in \{1, 2, 3, 4, 5\}$), define a set of $2n$-tuples of nonnegative integers $\text{push}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, Z, q')$ as follows: $(u_1, ..., u_n, v_1, ..., v_n)$ is in $\text{push}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, Z, q')$ if there is a sequence of moves of M such that,

1. the computation starting from state q with stack top symbol T, M will not pop out this T and the last move is a push of Z onto the stack and end with state q' (Notice that, prior to this last move, the sequence may involve many pushes/pops.).
2. The computation remains within the specified mode vectors of the counters.
3. For $i = 1, .., n$, u_i (v_i) is the number of times counter i is incremented (decremented) by 1. So, for example, for $V(c_1, Q_2, 0, 0)$, $u_1 = 0$ and $v_1 = 0$; for $V(c_1, Q_2, 0, +)$, $u_1 > 0$ and $v_1 = 0$; for $V(c_1, Q_2, 0, rev)$, $u_1 > 0$ and $v_1 > 0$; for $V(c_1, Q_2, rev, rev)$, $u_1 = 0$ and $v_1 \geqslant 0$; for $V(c_1, Q_2, rev, zero)$, $u_1 = 0$ and $v_1 \geqslant 0$; for $V(c_1, Q_2, zero, zero)$, $u_1 = 0$ and $v_1 = 0$, etc.

Thus, $\text{push}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, Z, q')$ gives separate counts of the total increments and total decrements for each counter of M during the computation.

Similarly to $\text{pop}^*(q, Z, T, q')$ and $\text{stay}^*(q, T, q')$ in Section 2, we can define the sets $\text{pop}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), Z, T, q')$ and $\text{stay}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, q')$.

Lemma 3. *We can construct C-generators for* $\text{push}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, Z, q')$, $\text{pop}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), Z, T, q')$, *and* $\text{stay}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, q')$.

Proof. First we construct a PC-generator B with $2n$ counters which simulates the computation of M starting in state q with its stack top T. During the simulation, B makes sure that items 1 and 2 are satisfied. The simulation halts when M writes Z on the top of symbol T and moves right in state q'. From Lemma 2, B can be converted to an equivalent C-generator for

$$\text{push}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, Z, q').$$

Similarly, we can construct C-generators for

$$\text{pop}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), Z, T, q')$$

and $\text{stay}^*(q, V(c_1, Q_{i_1}), ..., V(c_n, Q_{i_n}), T, q')$. ∎

For notational convenience, $(V(c_1, Q_{i_1}), \cdots, V(c_n, Q_{i_n}))$ will simply be denoted by V and will be called a global mode vector. Note that there are only a finite number of distinct global mode vectors. We use $A_{\text{push}}(q, V, T, Z, q')$, $A_{\text{pop}}(q, V, Z, T, q')$, and $A_{\text{stay}}(q, V, T, q')$ to denote the C-generators for $\text{push}^*(q, V, T, Z, q')$, $\text{pop}^*(q, V, Z, T, q')$ and $\text{stay}^*(q, V, T, q')$, respectively.

Let V and V' be two global mode vectors. Let $\langle c, Q_i, m, m' \rangle$ be a mode vector for c in V and $\langle c, Q_j, m'', m''' \rangle$ the corresponding mode vector for c in V'. We say that V

and V' are compatible with respect to counter c if $Q_i = Q_j$, $m' = m''$, and m''' is a proper mode for Q_i (so, e.g., rev and $zero$ are not proper for Q_3; $zero$ is not proper for Q_1). Two global mode vectors V and V' are compatible if they are compatible with respect to every counter c. We are now ready to prove:

Theorem 2. *$Binary(M)$ of a PCM M can be accepted by a 2-tape CA.*

Proof. From definition, $Binary(M) = \{(\alpha, \beta)|$ configuration α can reach configuration β in $M\}$. We construct a 2-tape CA B to accept $Binary(M)$. The specifications of all the C-generators $A_{\text{push}}(q, V, T, Z, q')$, $A_{\text{pop}}(q, V, Z, T, q')$, and $A_{\text{stay}}(q, V, T, q')$ are incorporated in the states of B. We describe the operation of B when given configurations α and β on its first and second input input tapes, respectively. Let $\alpha = wqd_1^{x_1} \cdots d_n^{x_n}$ and $\beta = w'q'd_1^{x'_1} \cdots d_n^{x'_n}$. Let $w = Z_1 \cdots Z_k$ and $w' = Z'_1 \cdots Z'_{k'}$.

B reads the two tapes in parallel and makes sure that the symbol under head 1 is the same as the symbol under head 2. Nondeterministically, B starts to operate in the following way. Assume that both heads are at the m-th ($m \geqslant 1$) cell with $Z_1 \cdots Z_{m-1} = Z'_1 \cdots Z'_{m-1}$. There are four cases to consider (nondeterministically chosen):

Case 1. $m \leqslant k$ and $m \leqslant k'$.
Case 2. $m \leqslant k$ and $m = k' + 1$.
Case 3. $m = k + 1$ and $m \leqslant k'$.
Case 4. $m = k + 1 = k' + 1$.

Consider Case 1. B operates in two phases. In the first phase, B reads the rest of the first input tape and guesses a sequence of pop-generators (when $m = 1$, treat Z_{m-1} as the stack bottom B_0)

$$A_{\text{pop}}(q_0, V_0, Z_{m-1}, Z_m, q_1), \cdots, A_{\text{pop}}(q_{k-m}, V_{k-m}, Z_{k-1}, Z_k, q_{k-m+1})$$

such that V_{i+1} and V_i are compatible and each $\text{pop}^*(q_{i+1}, V_{i+1}, Z_{i+m-1}, Z_{i+m}, q_i)$ is not empty for $i = 0, \cdots, k - m$, and $q_{k-m+1} = q$. Further, V_{k-m} is consistent with the starting counter values $x_1, \cdots, x_n$, e.g., if the behavior of counter c_1 in V_{k-m} is Q2 (c_1 starts from 0), then x_1 must be 0. Note that each counter c in M is associated with two counters c^+ and c^- in the C-generators to keep track of the increments and decrements in counter c. In order to decide the counter values at the beginning of the second phase below, B guesses the value y_i for each counter c_i, and verifies, at the end of phase 1 by using auxiliary counters, that $y_i + \Sigma c_i^- - \Sigma c_i^+ = x_i$ where Σc_i^+ (resp. Σc_i^-) is the total increments (decrements) made to counter c_i for all the pop generators in phase 1. Note that "increments" in each pop generator essentially means "decrements" to y_i, since the pop generators are supposed to change the values of the c_i's from x_i's to y_i's. Doing these ensures that configuration $Z_1 \cdots Z_k qd_1^{x_1} \cdots d_n^{x_n}$ can reach the intermediate configuration γ (i.e., $Z_1 \cdots Z_{m-1}q_0d_1^{y_1} \cdots d_n^{y_n}$) through a sequence of moves that never pops symbol Z_{m-1} out.

Now, B starts phase 2, with counter values $y_1, \cdots, y_n$ for counters $c_1, \cdots, c_n$ in M. B then reads the rest of the second input tape and guesses a sequence of push generators

$$A_{\text{push}}(p_0, U_0, Z'_{m-1}, Z'_m, p_1), \cdots, A_{\text{push}}(p_{k'-m}, U_{k'-m}, Z'_{k'-1}, Z'_{k'}, p_{k'-m+1})$$

such that $p_0 = q_0$ and, V_0 and U_0 are compatible (i.e., M continues its computation from the intermediate configuration γ that was reached from the starting configuration α). B also checks that U_i and U_{i+1} are compatible and each push$^*(p_i, U_i, Z'_{i+m-1}, Z'_{i+m}, p_{i+1})$ is not empty for $i = 0, \cdots, k' - m$, and $p_{k'-m+1} = q'$. In order to verify the intermediate configuration γ can reach configuration β, B needs to check (similar to phase 1) that $y_i - \Sigma c_i^- + \Sigma c_i^+ = x'_i$ where Σc_i^+ (resp. Σc_i^-) is the total increments (decrements) made to counter c_i for all the push generators in phase 2. Finally, B needs to check that the ending counter values $x'_1, \cdots, x'_n$ are consistent with the last mode vector $U_{k'-m}$. For instance, if counter c_1 has behavior pattern Q4 in $U_{k'-m}$, then x'_1 must be positive. B accepts if all the guesses are successful, i.e., α can reach β.

Cases 2 - 4 are handled similarly, where the C-generators $A_{\text{stay}}(q, V, T, q')$ for $\text{stay}^*(q, V, T, q')$ are also used in the construction.

Hence, $Binary(M)$ can be accepted by a 2-tape CA B. ∎

As in Corollaries 1 and 2, we have:

Corollary 3. *Let M be a PCM and S and T be sets of configurations of M accepted by CAs. Then $Binary(M, S, T)$ can be accepted by a 2-tape CA, and $pre^*(M, S)$ and $post^*(M, S)$ can be accepted by CAs.*

3 Finite-Crossing WCMs

Let M be a finite-crossing WCM with n counters. A configuration of M is represented as a string $\alpha = w_1 q w_2 d_1^{x_1} d_2^{x_2} ... d_n^{x_n}$, where $w = w_1 w_2$ is the content of the read/write worktape with the head at the leftmost symbol of w_2, q is the state, $d_1, ..., d_n$ are distinct symbols, and $x_1, x_2, ..., x_n$ are the values of the counters. We can prove the following:

Theorem 3. *Let M be a finite-crossing WCM. Then $Binary(M)$ can be accepted by a 2-tape CA.*

Corollary 4. *Let M be a finite-crossing WCM and S and T be sets of configurations of M accepted by CAs. Then $Binary(M, S, T)$ can be accepted by a 2-tape CA, and $pre^*(M, S)$ and $post^*(M, S)$ can be accepted by CAs.*

Corollary 5. *Let M be a finite-crossing WCM without counters (i.e., the only memory structure is a finite-crossing read/write worktape) and S and T be regular sets. Then $Binary(M, S, T)$ can be accepted by a 2-tape FA, and $pre^*(M, S)$ and $post^*(M, S)$ can be accepted by FAs.*

4 WCMs and PCMs with Clocks

A timed automaton is a finite-state machine *without* an input tape augmented with finitely many real-valued unbounded clocks [1]. All the clocks progress synchronously with rate 1, except that when a nonempty subset of clocks are reset to 0 at some transition, the other clocks do not progress. A transition between states fires if a clock constraint is satisfied. 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 clocks. Here we only consider integer-valued clocks, i.e., discrete timed automata. A discrete pushdown timed automaton (finite-crossing worktape timed automaton) is a discrete time automaton with a pushdown stack (finite-crossing read/write tape). We can further generalize these models by augmenting them with reversal-bounded counters, call them PTCM and finite-crossing WTCM, respectively. Thus a PTCM (finite-crossing WTCM) is a PCM (finite-crossing WCM) with clocks. A configuration of a PTCM (finite-crossing WTCM) now contains the values of the clocks. It is known that the binary reachability of a PTCM (finite-crossing WTCM) can be accepted by a 2-tape PCA (finite-crossing WCA) [10, 21]. The results in the previous section can be generalized:

Theorem 4. *Let M be a PTCM (or a finite-crossing WTCM). Then $Binary(M)$ can be accepted by a 2-tape CA.*

Corollary 6. *Let M be a PTCM (or a finite-crossing WTCM) and S and T be sets of configurations of M accepted by CAs. Then $Binary(M, S, T)$ can be accepted by a 2-tape CA, and $pre^*(M, S)$ and $post^*(M, S)$ can be accepted by CAs.*

5 Model-Checking and Satisfiability-Checking

It is important to formulate what kinds of temporal properties are decidable for the machine models discussed in this paper. Given a machine (a PM, PCM, finite-crossing WCM, or its timed version) M and a configuration α, we use α_{c_i} to denote the value of counter c_i in α, $\alpha_{\#_a}$ to denote the number of appearances of symbol a in the stack/tape content in α, $\alpha_{\mathbf{q}}$ to denote the state in α. Let P be a Presburger formula on variables α_{c_i}, $\alpha_{\#_a}$, and $\alpha_{\mathbf{q}}$. Since the solutions of P can be accepted by a deterministic CA [16], it is obvious that the set of configurations satisfying P can be accepted by a deterministic CA. Particularly, if P is a Boolean combination of atomic formulas like $x > k$, $x = k$, where x is a variable (α_{c_i}, $\alpha_{\#_a}$, or $\alpha_{\mathbf{q}}$), and k is an integer, then P is called a *regular* formula. Obviously, the set of configurations satisfying a regular formula A can be accepted by a FA.

Now, we describe a (subset of a) Presburger linear temporal logic $\mathcal{L}$ as follows. This logic is inspired by the recent work of Comon and Cortier [5] on model-checking a special form of counter automata without nested cycles. Formulas in $\mathcal{L}$ are defined as:

$$f ::= \ P \mid A \mid P \wedge f \mid f \vee f \mid \ \circ f \mid A \; U \; f$$

where P is a Presburger formula, A is a regular formula, $\circ$ and U stand for *next* and *until*, respectively. Formulas in $\mathcal{L}$ are interpreted on (finite) sequences p of configurations of M in a usual way. We use p^i to denote the sequence resulting from the deletion of the first i configurations from p. We use p_i to indicate the i-th element in p. The satisfiability relation $\models$ is recursively defined as follows, for each sequence p and for each formula $f \in \mathcal{L}$ (written $p \models f$):

$p \models P$ if $p_1 \in P$,

$p \models A$ if $p_1 \in A$,

$p \models P \wedge f$ if $p \models P$ and $p \models f$,
$p \models f_1 \vee f_2$ if $p \models f_1$ or $p \models f_2$,
$p \models \circ f$ if $p^1 \models f$,
$p \models A\ U\ f$ if there exists j (which is not greater than the length of p) such that $p^j \models f$ and $\forall k < j (p^k \models A)$.

We use the convention that $\Diamond f$ (eventual) abbreviates $(true\ U\ f)$. The satisfiability-checking problem is to check whether there is an execution p of M satisfying $p \models f$, for a given M and $f \in \mathcal{L}$. The model-checking problem, which is the dual of the satisfiability-checking problem, is to check whether for all execution p of M satisfying $p \models f$, for a given M and $f \in \mathcal{L}$. The results of this paper show that:

Theorem 5. *The satisfiability-checking problem is decidable for $\mathcal{L}$ with respect to the following machine models: PM, PCM, finite-crossing WCM, and their timed versions.*

Proof. (sketch) Given $f \in \mathcal{L}$, we use $[f]$ to denote the set of p_1 such that $p \models f$. For each of the machine models, we will show $[f]$ can be accepted by a CA. Therefore, the theorem follows by noticing that the satisfiability-checking problem is equivalent to testing the emptiness of the CA, which is decidable.

We will only look at PCM; all the other models can be handled similarly. The proof is based upon an induction on the structure of $\mathcal{L}$. Obviously, $[P]$ and $[A]$ can be accepted by CAs; so can $[f_1 \vee f_2]$ if both $[f_1]$ and $[f_2]$ can. $[P \wedge f]$ can be accepted by a CA, since $[f]$ can be accepted by a CA and $[P]$ can be accepted by a deterministic CA. For the case of $[A\ U\ f]$, notice that the set $[A\ U\ f]$ is very similar to $Pre^*(M, [f])$ – the only difference is that $[A\ U\ f]$ further requires that each intermediate configuration on the path leading to $[f]$ to be in A. This requirement can be easily fulfilled by slightly modify M, thanks to the fact that A is regular. Therefore, Corollary 3 still applies to show that $[A\ U\ f]$ can be accepted by a CA. The case for $[\circ f]$ is simpler, since we only look at one move. ∎

$\mathcal{L}$ is quite powerful. For instance, it can express a property like $\Diamond(P_1 \wedge \Diamond P_2)$. We should point out that without using the results in this paper, this property cannot be checked. For instance, the timed version of PM was studied in [10]. In that paper, it was shown that $[P_1 \wedge \Diamond P_2]$ can be accepted by a PCA – this is bad, since it is not possible to characterize $[\Diamond(P_1 \wedge \Diamond P_2)]$ from here (a machine accepting $[\Diamond(P_1 \wedge \Diamond P_2)]$ may need two stacks (i.e., Turing): one stack is for the PM, the other is for $[P_1 \wedge \Diamond P_2]$). But now, we have a stronger characterization for $[P_1 \wedge \Diamond P_2]$: it can be accepted by a CA. Therefore, the results in this paper give a CA characterization for $[\Diamond(P_1 \wedge \Diamond P_2)]$.

Since the model-checking problem is the dual of the satisfiability-checking problem, we conclude that

Theorem 6. *The model-checking problem is decidable for $\neg\mathcal{L}$ (taking negation of each formula in $\mathcal{L}$) with respect to the following machine models: PM, PCM, finite-crossing WCM, and their timed versions.*

Acknowledgment: We wish to thank Nicholas Tran for helpful discussions concerning this work.

References

1. R. Alur and D. Dill. *"The theory of timed automata," TCS*, 126(2):183-236, 1994
2. P. Abdulla and B. Jonsson. *"Verifying programs with unreliable channels," Information and Computation*, 127(2): 91-101, 1996
3. A. Bouajjani, J. Esparza, and O. Maler. *"Reachability analysis of pushdown automata: application to model-Checking," CONCUR'97*, **LNCS** 1243, pp. 135-150
4. G. Cece and A. Finkel. *"Programs with Quasi-Stable Channels are Effectively Recognizable," CAV'97*, **LNCS** 1254, pp. 304-315
5. H. Comon and V. Cortier. *"Flatness is not a weakness," Proc. Computer Science Logic*, 2000
6. H. Comon and Y. Jurski. *"Multiple counters automata, safety analysis and Presburger arithmetic," CAV'98*, **LNCS** 1427, pp. 268-279
7. H. Comon and Y. Jurski. *"Timed automata and the theory of real numbers," CONCUR'99*, **LNCS** 1664, pp. 242-257
8. Z. Dang. *"Binary reachability analysis of timed pushdown automata with dense clocks," CAV'01*, **LNCS** 2102, pp. 506-517
9. Z. Dang and O. H. Ibarra. *"Liveness verification of reversal-bounded counter machines with a free counter,"* submitted, 2001
10. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer and J. Su. *"Binary Reachability Analysis of Discrete Pushdown Timed Automata," CAV'00*, **LNCS** 1855, pp. 69-84
11. Z. Dang, P. San Pietro, and R. A. Kemmerer. *"On Presburger Liveness of Discrete Timed Automata," STACS'01*, **LNCS** 2010, pp. 132-143
12. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. *"Efficient Algorithms for Model Checking Pushdown Systems," CAV'00*, **LNCS** 1855, pp. 232-247
13. A. Finkel and G. Sutre. *"Decidability of Reachability Problems for Classes of Two Counter Automata," STACS'00*, **LNCS** 1770, pp. 346-357
14. A. Finkel, B. Willems, and P. Wolper. *"A direct symbolic approach to model checking pushdown systems," INFINITY'97*
15. S. Ginsburg and E. Spanier. *"Bounded Algol-like languages," Transactions of American Mathematical Society*, 113, pp. 333-368, 1964.
16. O. H. Ibarra. *"Reversal-bounded multicounter machines and their decision problems," J. ACM*, **25** (1978) 116-133
17. O. H. Ibarra. *"Reachability and safety in queue systems with counters and pushdown stack," Proceedings of the International Conference on Implementation and Application of Automata*, pp. 120-129, 2000.
18. O. H. Ibarra, T. Bultan, and J.Su. *"Reachability analysis for some models of infinite-state transition systems," CONCUR'00*, pp. 183-198, 2000.
19. O. H. Ibarra. *"Reachability and safety in queue systems with counters and pushdown stack," Proceedings of the International Conference on Implementation and Application of Automata*, pp. 120-129, 2000
20. O. H. Ibarra, Z. Dang, and P. San Pietro, *"Verification in Loosely Synchronous Queue-Connected Discrete Timed Automata,"* submitted. 2001
21. O. H. Ibarra and J. Su. *"Generalizing the discrete timed automaton," Proceedings of the International Conference on Implementation and Application of Automata*, 206-215, 2000.
22. O. H. Ibarra, J. Su, T. Bultan, Z. Dang, and R. A. Kemmerer. *"Counter Machines: Decidable Properties and Applications to Verification Problems,", MFCS'00*, **LNCS 1893**, pp. 426-435
23. K. L. McMillan. *"Symbolic model-checking - an approach to the state explosion problem,"* PhD thesis, Department of Computer Science, Carnegie Mellon University, 1992
24. W. Peng and S. Purushothaman. *"Analysis of a Class of Communicating Finite State Machines," Acta Informatica*, 29(6/7): 499-522, 1992