Safety Verification for Two-Way Finite Automata with Monotonic Counters [⋆]

Oscar H. Ibarra[1⋆⋆], Zhe Dang[2] and Zhi-Wei Sun[3]

[1]Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

[2]School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164, USA

[3] Department of Mathematics
Nanjing University
Nanjing 210093, China

Abstract. We look at a model of a two-way nondeterministic finite automaton augmented with monotonic counters operating on inputs of the form $a_1^{i_1}...a_n^{i_n}$ for some fixed n and distinct symbols $a_1, ..., a_n$, where $i_1, ..., i_n$ are nonnegative integers. Our results concern the following Presburger safety verification problem: Given a machine M, a state q, and a Presburger relation E over counter values, is there $(i_1, ..., i_n)$ such that M, when started in its initial state on the left end of the input $a_1^{i_1}...a_n^{i_n}$ with all counters initially zero, reaches some configuration where the state is q and the counter values satisfy E? We give positive and negative results for different variations and generalizations of the model (e.g., augmenting the model with reversal-bounded counters, discrete clocks, etc.). In particular, we settle an open problem in [10].

1 Introduction

Recently, there has been significant progress in automated verification techniques for finite-state systems. One such technique, called model-checking [5, 6, 22, 20, 23], explores the state space of a finite-state system and checks that a desired temporal property is satisfied. In recent years, model-checkers like SMV [20] and SPIN [16] have been successful in some industrial-level applications. Successes in finite-state model-checking have inspired researchers to develop model-checking techniques for infinite-state systems (e.g., arithmetic programs that contain integer variables and parameters).

However, for infinite-state systems, a fundamental problem - the decidability problem - should be addressed before any attempts to develop automatic verification procedures. It is well known that in general, it is not possible to automatically verify whether

⋆ The work by Oscar H. Ibarra has been supported in part by NSF Grants IIS-0101134 and CCR02-08595.

⋆⋆ Corresponding author (`ibarra@cs.ucsb.edu`).

an arithmetic program with two integer variables (i.e., a Minsky machine) is going to halt [21]. Therefore, a key aspect of the research on infinite-state system verification is to identify what kinds of practically important infinite-state models are decidable with respect to a particular form of properties.

Counter machines are considered a natural model for specifying reactive systems containing integer variables. They have also been found to have a close relationship to other popular models of infinite-state systems, such as timed automata [1]. Since general counter machines like Minsky machines are undecidable for verification of any nontrivial property, studying various restricted models of counter machines may help researchers to obtain new decidable models of infinite-state systems and identify the boundary between decidability and undecidability.

In this paper, we study a model of a two-way nondeterministic finite automaton augmented with monotonic counters operating on inputs of the form $a_1^{i_1}...a_n^{i_n}$ for some fixed n and distinct symbols $a_1, ..., a_n$, where $i_1, ..., i_n$ are nonnegative integers. Our results concern the following Presburger safety verification problem: Given a machine M, a state q, and a Presburger relation E over counter values, is there $(i_1, ..., i_n)$ such that M, when started in its initial state on the left end of the input $a_1^{i_1}...a_n^{i_n}$ with all counters initially zero, reaches some configuration where the state is q and the counter values satisfy E? We give positive and negative results for different variations and generalizations of the model (e.g., augmenting the model with reversal-bounded counters, discrete clocks, etc.). In particular, we settle an open problem in [10]. In the sequel, we refer to the tuples of counter values that can be reached by M (on all inputs) as the reachable set at state q, or simply reachability set when q is understood.

Some of our decidable results are quite powerful in the sense that the underlying machines may have nonsemilinear (i.e., non Presburger) reachability sets. This is in contrast to most existing decidable results in model-checking infinite-state systems (e.g., pushdown systems [4]) that essentially have semilinear reachability sets. For instance, consider a nondeterministic transition system M that is equipped with a number of monotonic counters $\boldsymbol{X} = X_1, ..., X_k$ and a number of nonnegative integer parameterized constants $A_1, ..., A_n$. M has a finite number of control states. On a transition from one control state to another, M is able to increment the counter synchronously (i.e., $\boldsymbol{X} := \boldsymbol{X} + \delta$ that increments every X_i by the same amount δ, where $\delta = 0, 1$, or some A_j), test the truth value of a Presburger formula on $\boldsymbol{X}$ and $A_1, \ldots, A_n$, or reset a counter to 0. M is a form of a generalized timed automaton (with discrete clocks, Presburger enabling conditions, and parameterized durations) [10] in disguise. We further require that each counter is reset for at most a fixed number of times (such as 10). We show in this paper that the safety verification problem is decidable for such M. It is easy to see that M, with these reset-bounded counters, can exhibit a reachability set satisfying $A_j | X_i$, which is not Presburger.

2 Preliminaries

Let $\mathbf{N}$ be the set of nonnegative integers and $c \in \mathbf{N}$. A c-counter machine is a two-way nondeterministic finite automaton with input endmarkers (two-way NFA) augmented with c counters, each of which can be incremented by 1, decremented by 1, and tested for zero. We assume, w.l.o.g., that each counter can only store a nonnegative integer, since the sign can be stored in the states. If r is a nonnegative integer, let 2NCM(c,r) denote the class of c-counter machines where each counter is r *reversal-bounded,* i.e., it makes at most r alternations between nondecreasing nonincreasing modes in any computation; 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. For convenience, we sometimes refer to a machine in the class as a 2NCM(c,r). Note that the counters in a 2NCM(c,0) are monotonic (i.e., nondecreasing). A 2NCM(c,r) is *finite-crossing* if there is a positive integer c such that in any computation, the input head crosses the boundary between any two adjacent cells of the input no more than c times. Note that a 1-crossing 2NCM(c,r) is a one-way nondeterministic finite automaton augmented with c r-reversal counters. 2NCM(c) will denote the class of c-counter machines whose counters are r-reversal bounded for some given r. For deterministic machines, we use 'D' in place of 'N'. If M is a machine, $L(M)$ denotes the language it accepts.

We will need the following result from [17].

Theorem 1. *There is a* fixed r *such that the emptiness problem for 2DCM(2,r) over bounded languages is undecidable.*

We will also need the following results.

Theorem 2. *The emptiness problem is decidable for the following machine classes: (a) 2DCM(1) [18], (b) 2NCM(1) over a bounded language [8], (c) 2NCM(c) over a unary alphabet (i.e., over a bounded language on 1 letter) for every c [18], and (d) finite-crossing 2NCM(c) for every c [17, 15].*

Let Y be a finite set of variables over integers. For all integers a_y, with $y \in Y$, b and c (with $b > 0$), $\sum_{y \in Y} a_y y < c$ is an *atomic linear relation* on Y and $\sum_{y \in Y} a_y y \equiv_b c$ is a *linear congruence* on Y. A *linear relation* on Y is a Boolean combination (using $\neg$ and $\wedge$) of atomic linear relations on Y. A *Presburger formula* on Y is the Boolean combination of atomic linear relations on Y and linear congruences on Y. A set P is *Presburger-definable* if there exists a Presburger formula $\mathcal{F}$ on Y such that P is exactly the set of the solutions for Y that make $\mathcal{F}$ true. It is well known that Presburger formulas are closed under quantification.

Let k be a positive integer. A subset S of $\mathbf{N}^k$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^k$ such that $S = \{v \mid v = v_0 + a_1 v_1 + \ldots + a_t v_t, \forall 1 \leq i \leq t, a_i \in \mathbf{N}\}$. S is a *semilinear set* if it is a finite union of linear sets. It is known that S is a semilinear set iff S is Presburger-definable [14].

Let Σ be an alphabet consisting of k symbols $a_1, \ldots, a_k$. For each string (word) w in Σ^*, we define the *Parikh map* $p(w)$ of w as follows: $p(w) = (i_1, \ldots, i_k)$, where i_j is the number of occurrences of a_j in w. If L is a subset of Σ^*, the *Parikh map* of L is defined by $p(L) = \{p(w) \mid w \in L\}$. L is a *semilinear language* if its Parikh map $p(L)$ is a semilinear set. Obviously, if the languages accepted by machines in a class C are effectively semilinear (that is, for each M in C, the semilinear set $p(L(M))$ can be effectively computed), then the emptiness problem for C is decidable. We will need the following theorem from [17]:

Theorem 3. *Let M be a finite-crossing 2NCM(c). Then $p(L(M))$ is a semilinear set effectively computable from M.*

Note the result above is not true for machines that are not finite-crossing. For example, a 2DCM(1,1) can recognize the language $\{0^i 1^j \mid i \text{ divides } j\}$, which is not semilinear.

3 Machines with Monotonic Counters over Bounded Languages

3.1 One-Way Input

Let M be a nondeterministic one-way finite automaton augmented with a number of reversal-bounded counters. An input to M is of the form $a_1^{i_1}...a_n^{i_n}$ for some fixed n and distinct symbols $a_1, ..., a_n$, where $i_1, ..., i_n$ are nonnegative integers. It is known that the Presburger safety verification problem for these machines is decidable [17], even when *one* of the counters is unrestricted in that it can change mode from nondecreasing to nonincreasing and vice-versa an unbounded number of times. The machine M can further be generalized by allowing some of the reversal-bounded counters to reset to zero. Thus M now has one unrestricted counter, some "reset counters", and some reversal-bounded counters. There is no fixed bound on the number of times a reset counter can reset (but between two resets, the counter is reversal-bounded). Reset counters are quite useful in modeling clocks in real-time systems [1]. If each counter is monotonic (i.e., 0-reversal-bounded) between two resets, the problem is decidable [7]. (The decidability holds even when the machine is allowed to have discrete clocks which operate and can be tested as in a timed automaton [1].) For the general case, the aforementioned machines with at least four reset counters is undecidable for the problem. This can be shown by a reduction to two-counter machines, noticing that an unrestricted counter can be simulated by two reset counters, each of which is 1-reversal-bounded between two resets.

Other generalizations of the model which allows linear constraints as counter tests have also been shown decidable for the problem [19].

3.2 Two-Way Input

We now study the more complicated model when the input to the machine is two-way. So now M is a nondeterministic two-way finite automaton augmented with monotonic counters $C_1, ..., C_k$ operating on an input of the form $a_1^{i_1}...a_n^{i_n}$, with left and right endmarkers. There is an interesting model equivalent to this model.

Let M' be a nondeterministic machine with 1 unrestricted counter U and k monotonic counters C_1, ..., C_k. M' has *no* input tape. There are n parameterized constants $A_1, ..., A_n$. Initially, all the counters are zero. During the computation, M' can check if the unrestricted counter U is equal to 0 or equal to A_i ($i = 1, .., n$). Moreover, U is not allowed to exceed $max(A_1, ..., A_n)$.

M' is equivalent to M. An input $a_1^{i_1}...a_n^{i_n}$ (with left and right endmarkers) to M corresponds to the parameterized constants, by interpreting $i_1, (i_1 + i_2), ..., (i_1 + i_2 + ... + i_n)$ to be the values of $A_1, A_2, ..., A_n$, respectively. This correspondence assumes that $A_1 \leq A_2 ... \leq A_n$. However, since there are only a finite number of these orderings, this assumption can be made without loss of generality. Clearly, M starting on the left endmarker corresponds to the unrestricted counter U of M' starting at zero. M moving right (left) on the input corresponds to M' incrementing (decrementing) U. M' can determine if M is on the boundary between a_i and a_{i+1} by checking if U is equal to A_i. In view of the equivalence of the two models, will use them interchangeably in the sequel.

Let E be a Presburger relation on the values $c_1, ..,c_k$ of the monotonic counters and q be a state of M. We say that M *satisfies* E if there is $(i_1, ..., i_n)$ such that M, when started in its initial state on the left end of the input $a_1^{i_1}...a_n^{i_n}$ with all counters initially zero, reaches some configuration where the state is q and the counter values satisfy E. Note that we do not assume that M necessarily halts. So, in fact, a configuration satisfying E may be an intermediate configuration in some computation. Also, there may be several configurations satisfying E. In the paper, we usually do not explicitly specify the state q when E is satisfied, since its specification is usually obvious from construction.

An *atomic equality relation* is a relation $c_i = c_j, i \neq j$. A relation E is an *equality relation* if it is a conjunction of atomic equality relations and no c_i is involved in more than one atomic relation in E. We denote such a relation by E_e.

Theorem 4. *Let M be as specified above. Let E and E_e be a Presburger relation and an equality relation, respectively.*

1. *There is a fixed k such that it is undecidable whether M with k monotonic counters satisfies E_e.*
2. *When $n = 1$ (i.e., there is only one parameterized constant A_1), it is decidable whether M satisfies E.*
3. *When M is deterministic and always halts, it is decidable whether M satisfies E.*

Proof. We first prove Part 1. From Theorem 1, there is a fixed r such the emptiness problem for 2DCM(2,r) over bounded languages is undecidable. Let M be such an automaton. Clearly, we can convert M to an equivalent automaton M' which has $2(1 + \frac{r-1}{2})$ 1-reversal counters where each counter starts at zero, and on input w, M' accepts if and only if it halts with all counters zero. To insure this, we can add to M' a dummy counter which is incremented at the beginning and only decremented, i.e., becomes zero when the input is accepted. Suppose M' has k 1-reversal counters, $C_1, ..., C_k$ (one of these is the dummy counter). Note that k is fixed since r is fixed. We modify M' to a nondeterministic automaton M'' with $2k$ monotonic counters: $C_1^+, C_1^-, ..., C_k^+, C_k^-$, where C_i^+ and C_i^- are associated with counter C_i. M'' on input w simulates the computation of M' , first using counter C_i^+ to simulate C_i when the latter is in a nondecreasing mode. When counter C_i reverses (thus entering the non-increasing mode), M'' continues the simulation but using counter C_i^-: incrementing this counter when M' decrements C_i. At some time during the computation (which may be different for each i), M' guesses that C_i has reached the zero value. From that point on, M'' will no longer use counters C_i^+ and C_i^-, but continues the simulation. When M'' has guessed that $C_i^+ = C_i^-$ for all i's (note that for sure the two counters corresponding to the dummy counter are equal), it halts. Clearly, w is accepted by M' if and only if M'' on w can reach a configuration with $C_i^+ = C_i^-$ for $i = 1, ..., k$ (this is relation E_e). As noted earlier, M'' on input $a_1^{i_1}...a_n^{i_n}$ can be interpreted as a machine M''' with an unrestricted counter U and n parameterized constants $A_1, ..., A_n$ whose values are set to $(1+i_1), (1+i_1+i_2), ..., (1+i_1+i_2+...+i_n)$, respectively. The movement of the input head of M'' is simulated by the unrestricted counter U.

For Part 2, given M with monotonic counters $C_1, ..., C_k$, we construct a two-way nondeterministic finite automaton with reversal-bounded counters over a unary input (with endmarkers) M'. M' simulates M faithfully, with the input head of M' simulating the unrestricted counter U of M and k counters simulating the C_i's. At some point, M' guesses that the values of the monotonic counters satisfy the relation E. M' then uses another set of counters to verify that this is the case, and accepts. The result follows from the decidability of the emptiness problem for two-way nondeterministic finite automata augmented with reversal-bounded counters over a unary alphabet[18], and the fact that a Presburger relation on $C_1, ..., C_k$ can be verified by additional reversal-bounded counters.

For Part 3, since M is deterministic and always halts, M' (i.e., the M' in the proof of Part 1) is finite-crossing on any input. The result then follows from the decidability of the emptiness problem for finite-crossing finite automata with reversal-bounded counters (Theorem 3), and the fact that Presburger relations can be verified by reversal-bounded counters. ∎

In Theorem 4, the relations E and E_e are defined over monotonic counters $C_1, ...,$ C_k. In fact, the theorem still holds when the relations are defined over $C_1, \ldots, C_k$, U, and $A_1, \ldots, A_n$. This is because $n+1$ additional monotonic counters can be added to "store" the values for U and $A_1, \ldots, A_n$ at the time when the test for E is performed.

Consider the following restriction on the counters: $C_1, \ldots, C_k$ are *ordered* in that all the increments of C_i are followed by those of C_j whenever $1 \leq i < j \leq k$. Then it can be shown, using Theorem 2(b) directly, that it is decidable whether M satisfies E.

Open Question: In Part 3, if we do not assume that M always halts, we do not know if the problem is still decidable, since the machine is no longer finite-crossing and, in fact, the set of tuples of reachable values of the monotonic counters is not semilinear in general. For example, consider a deterministic machine M with a two-way unary input and three monotonic counters C_1, C_2, C_3. On input x of length n, M initially stores n in C_1. Then M makes (left-to-right and right-to-left) sweeps of the input, adding 1 to C_2 and n to C_3 after every sweep. M iterates this process without halting. Let $Q = \{(i,j,k) \mid$ there is an instance in the computation where the values of the 3 counters are $i, j, k\}$. Thus Q is the set of all "reachable" counter values. Then Q is not semilinear; otherwise, since semilinear sets are closed under intersection, Q intersected with the semilinear set $\{(n,n,m) \mid n, m \text{ in } \mathbf{N}\}$ will yield $\{(n,n,n^2) \mid n \text{ in } \mathbf{N}\}$, which is not semilinear.

4 Generalized Counter Timed Systems

Timed automata [1] are a standard model for studying real-time systems. A timed automaton can be considered as a finite automaton augmented with a finite number of clocks. In this paper, we consider clocks taking values in $\mathbf{N}$ (i.e., clocks are discrete).

In a timed automaton, a clock can progress (i.e., $x := x + 1$) or reset ($x := 0$). However, one clock progresses iff all the clock progresses. In addition, clocks can participate tests in the form of comparisons between the difference of two clocks against a constant (e.g., $x - y > 5$).

The result in Part 2 of Theorem 4 can be further generalized. In [3], pushdown timed systems (these are timed automata equipped with a pushdown stack) with "observation" counters were studied. The purpose of the 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 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. Here we only consider the case when the clocks of the system are discrete. It follows from the results in [3] (see also [9]) that these systems have decidable safety properties, and the tuples of "reachable" values of observation counters is Presburger.

A special case is when the pushdown stack is replaced by an unrestricted counter U. Call this a counter timed system. Then the results in [3] obviously hold. Now we generalize the counter timed system in the following way:

1. Instead of observation counters, the system has a finite number of reversal-bounded counters, and these counters participate in the dynamic of the system (i.e., they can be tested by the system).
2. The system is parameterized in the sense that there is a parameterized constant A, and during the computation, the system can check if the unrestricted counter U is equal to A (in addition to being able to test if U is zero).
3. U cannot exceed A.

Note that an observation counter can be simulated by two reversal-bounded counters: one counter keeps track of the increases and another counter keeps track of the decreases. When the system guesses that it has reached the target configuration, the difference of the counter values can be computed in one of the counters, and the sign of the difference can be specified in the other counter, which is set to 0 for negative and 1 for positive. Call a system S defined above a *generalized counter timed system*. Such an S can model part of a memory management system as follows:

1. S dynamically allocates/retrieves memory to/from k processes. The total amount of memory available is equal to A (the parameterized constant).
2. S uses the unrestricted counter U to keep track of the available memory, decrementing (resp. incrementing) U when a process requests (releases) memory. Note that S can check if U is zero or equal to A.
3. Among the reversal-bounded counters, we might use $2k$ monotonic counters for the following purpose: Two counters P_i^+ and P_i^- are associated with process i. During the computation, P_i^+ (resp. P_i^-) is incremented if process i requests (resp. returns) memory from (resp. to) the system.
4. Other reversal-bounded counters can be used for other purposes (e.g., each process can use a monotonic counter to keep track of the "cost" of requesting memory; assume there is a unit charge per memory request).

5. As in a timed automaton, the (discrete) clocks are used to enforce timing constraints.

Questions concerning the relationships among the parameterized constant, unrestricted counter, reversal-bounded counters, and clocks can be asked, such as: Is it always the case that $U = A - (P_1^+ - P_1^-) + ... + (P_k^+ - P_k^-)$? Is it always the case that the total cost of process i is less than the total cost of process j? Is it always the case that S is fair (e.g., $3 \cdot |P_i^+ - P_j^+| < A$ for each i, j)? etc.

Clearly, the set of tuples of reachable counter values is not semilinear (even when there are no clocks). However, we can still show that safety is decidable. Let S be a generalized counter timed system. A configuration α of S is a tuple (a, q, X, u, R), where a is the value of the parameterized constant A, q is the state, X is the array of clock values, u is the value of the unrestricted counter with $u \leq a$, and R is the array of values of the reversal-bounded counters.

Theorem 5. *It is decidable to determine, given a generalized timed counter system S and two Presburger sets I and B of configurations of S, whether there is a computation of S that starts from some configuration in I and reaches some configuration in B. Thus, it it decidable to determine whether S can go from an initial configuration in I to an "unsafe" configuration in B.*

Proof. We can view the clocks in S as counters, which we shall refer to as clock-counters. Now reversal-bounded counters can only perform *standard tests* (comparing a counter against 0) and *standard assignments* (increment or decrement a counter by 1, or simply nochange). But clock-counters in S have nonstandard tests and nonstandard assignments. This is because a clock constraint allows comparison between two clocks like $x_2 - x_1 > 5$. Note that using only standard tests, we can compare the difference of two clock-counter values against an integer like 5 by computing $x_2 - x_1$ in another counter. But each time this computation is done, it will cause at least one 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. Denote by 2UNCM a 2NCM(c), for some c, over a unary alphabet. Thus, a 2UNCM is a nondeterministic two-way finite automaton over a unary alphabet augmented with finitely many reversal bounded counters. Define a semi-2UNCM as a 2UNCM which, in addition to reversal-bounded counters, has clock-counters that use nonstandard tests and nonstandard assignments as described in the previous paragraph.

Suppose we are given S and Presburger sets I and B. Let M_I and M_B be one-way nondeterministic finite automata with reversal-bounded counters accepting I and B, respectively. (Since I and B are Presburger, these machines can be constructed [17].) We construct a semi-2UNCM M which operates as follows. The input to M is a (which is a concrete instance of the parameterized constant A) with endmarkers. M starts by guessing and storing in its counters the components of the tuple (q, X, u, R), and checks that (a, q, X, u, R) is in I using M_I. M then simulates S starting in configuration $\alpha = (a, q, X, u, R)$. At some point during the simulation, M guesses that it has reached a configuration $\beta = (a, q', X', u', R')$ in B. M' then simulates M_B and accepts if β is in B.

The semi-2UNCM M can now be converted to an 2UNCM M' by simulating the clocks in M by reversal-bounded counters, using the techniques in [9]: First we construct a semi-2UNCM M_1 from M, where clock-counter comparisons are replaced by "finite table look-up", and therefore, nonstandard tests are not present in M_1. Then we eliminate the nonstandard assignments of the form $x := 0$ (clock resets) in M_1 and obtain the desired 2UNCM M'. The details can be found in [9]. The result now follows since emptiness is decidable for 2UNCM's from Theorem 2 (c). ∎

The safety problem can be generalized as follows (called the multi-safety problem).

– **Given:** A system S and Presburger sets $I, B_1, ..., B_k$ of configurations of S, $k > 0$.
– **Question:** Is there a computation of S that starts from some configuration α in I and goes through some configurations $\alpha_1, ..., \alpha_k$, where α_i is in B_i for $1 \leq i \leq k$? (Note that that $\alpha_1, ..., \alpha_k$ can occur at different times.)

The theorem above easily generalizes to:

Theorem 6. *The multi-safety problem for generalized counter timed system is decidable.*

If S is a generalized counter timed system, define $Reach(S)$ to be the binary reachability set $Reach(S) = \{(\alpha, \beta) \mid$ starting in configuration α, S can reach configuration $\beta\}$. As we have seen, the binary reachability $Reach(S)$ is not Presburger in general. However, we can give a nice characterization for it.

A 2UNCM is a nondeterministic two-way finite automaton over a unary alphabet augmented with reversal bounded counters. Define a k-2UNCM as a 2UNCM which, in addition to the two-way unary input, is provided with k distinguished counters, called input counters. (Note that the machine has other reversal-bounded counters aside from the input counters.) A tuple $(a, i_1, ..., i_k)$ in $\mathbf{N}^{k+1}$ is accepted by a k-2UNCM M if M, when started with a on the two-way input (i.e., a is the length of the unary input) and $i_1, ..., i_k$ in the k input counters, eventually enters an accepting state. Note that the input counters can be tested against zero and decremented during the computation, but cannot be incremented. The set accepted by M is the set of all $(k+1)$-tuples accepted by M. A multi-2UNCM is a k-2UNCM for some k. The proof of the following lemma is straightforward from Theorem 2(c).

Lemma 1. *The emptiness problem for multi-2UNCM is decidable.*

Using the above lemma and the ideas in the proof of Theorem 5, we can show the following characterization for the binary reachability $Reach(S)$:

Theorem 7. *If S is a generalized counter timed system, we can effectively construct a multi-2UNCM M accepting $Reach(S)$.*

5 Synchronized Monotonic Counters

Let $A_1, ..., A_n$ be nonnegative integer parameterized constants, and $\boldsymbol{X} = \{X_1, ..., X_k\}$ be a set of nonnegative integer variables. The variables in $\boldsymbol{X}$ are incremented synchronously. We use $\boldsymbol{X} := \boldsymbol{X} + \delta$ to indicate $X_1 := X_1 + \delta; \ldots X_k := X_k + \delta$. Consider the following instruction set:

$s : \boldsymbol{X} := \boldsymbol{X} + 0$,
$s : \boldsymbol{X} := \boldsymbol{X} + 1$,
$s : \boldsymbol{X} := \boldsymbol{X} + A_i$, for some $1 \le i \le n$,
s : if T then goto p else goto q,

and a nondeterministic choice,

s : goto p or q.

where s is an instruction label and test T is a Presburger formula over $A_1, \ldots, A_n$ and the variables in $\boldsymbol{X}$. A *synchronized counter program* P is a sequence of instructions drawn from the instruction set. A configuration of a program is a tuple consisting of an instruction label, values of $A_1, \ldots, A_n$, and values of variables in $\boldsymbol{X}$. Notice that, in P, there could be many different tests T's and in general P is nondeterministic.

Theorem 8. *It is decidable to determine, given a synchronized counter program P and two Presburger sets I and B of configurations of P, whether there is a computation of P that starts from some configuration in I and reaches some configuration in B. Thus, it it decidable to determine whether P can go from an initial configuration in I to an "unsafe" configuration in B.*

The proof of Theorem 8 uses Theorem 2 (b). Synchronized counter programs have a close relationship with timed automata (with discrete clocks). Though the standard model of timed automata is quite useful, more powerful clock tests are needed in order to model more complicated systems. In [10], we studied a class of generalized timed automata that allows Presburger clock tests. More precisely, a generalized timed automaton $\mathcal{A}$ is a synchronized counter program augmented with *reset instructions*:

$s : Reset(i)$,

where the monotonic variable (i.e., a clock) X_i is reset to 0 and all the other monotonic variables do not change their values. $\mathcal{A}$ is *deterministic* if $\mathcal{A}$ does not contain nondeterministic choice instructions. Since generalized timed automata can simulate two-counter machines [1], though there are practical needs to use generalized timed automata to specify complex real-time systems, the "Turing computing" power of the model prevents automatic verification of simple properties such as reachability. Therefore, decidable approximation techniques are of great interest, since these techniques would provide a user some degree of confidence in analyzing and debugging complex real-time specifications. In contrast to the most direct approximation techniques [2, 12, 11, 13] that bound the number of transitions to a fixed number, the approximation techniques presented in [10] restrict the clock behaviors but do not necessarily bound the number of transition iterations to be finite.

There are three approximation techniques in [10]. The r-reset-bounded approximation bounds the number of clock resets by a given positive integer r for each clock. The n-bounded approximation requires that, for each clock, the clock value is less than a given positive integer n every time the clock resets (but after the last reset, the clock can go unbounded.). Combining these two, the $\langle n, r\rangle$-crossing-bounded approximation requires that, for each clock, there are at most r times that the clock resets after its value is greater or equal to n. It was shown in [10] that, when $\mathcal{A}$ is deterministic, Theorem 8 holds for $\mathcal{A}$ under any one of the three approximations. The case when $\mathcal{A}$ is nondeterministic was left open in [10]. The following result settles the open case.

Theorem 9. *It is decidable to determine, given a generalized timed automaton $\mathcal{A}$ under any one of the three approximations, and two Presburger sets I and B of configurations of $\mathcal{A}$, whether there is a computation of $\mathcal{A}$ that starts from some configuration in I and reaches some configuration in B.*

Proof. We only consider the r-reset-bounded approximation; the other two approximations are similar (see [10]). In an r-reset-bounded execution of $\mathcal{A}$, each clock is reset for at most r times. Therefore, the execution can be partitioned into a concatenation of at most $r \times |\boldsymbol{X}|$ phases such that, in each phase, there is no clock reset, and any two consecutive phases are connected with a clock reset instruction. Notice that a phase can be modeled as an execution of a synchronized counter program (which does not contain resets). The theorem follows by generalizing Theorem 8 into a "concatenation" of the program for each phase. ∎

6 Conclusion

We introduced a model of a two-way nondeterministic finite automaton augmented with monotonic counters operating on inputs over a bounded language for investigating the verification properties of infinite-state systems. We then proved positive and negative results on the safety verification properties for different variations and generalizations of the model (e.g., augmenting the model with reversal-bounded counters, discrete clocks, etc.). In particular, we settled an open problem in [10].

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
2. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 592–601, San Diego, California, 16–18 May 1993.
3. A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
4. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model-checking. In *Concurrency (CONCUR 1997)*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer-Verlag, 1997.
5. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop of Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*. Springer, 1981.

6. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
7. Z. Dang. Phd. dissertation. *Department of Computer Science, University of California at Santa Barbara*, 2000.
8. Z. Dang, O. Ibarra, and Z. Sun. On the emptiness problems for two-way nondeterministic finite automata with one reversal-bounded counter. *Submitted*, 2002.
9. Zhe Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *Proceedings of the International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2000.
10. Zhe Dang, O. H. Ibarra, and R. A. Kemmerer. Decidable Approximations on Generalized and Parameterized Discrete Timed Automata. In *Proceedings of the 7th Annual International Computing and Combinatorics Conference (COCOON'01)*, volume 2108 of *Lecture Notes in Computer Science*, pages 529–539. Springer, 2001.
11. Zhe Dang and R. A. Kemmerer. A symbolic model-checker for testing ASTRAL real-time specifications. In *Proceedings of the Sixth International Conference on Real-time Computing Systems and Applications*, pages 131–142. IEEE Computer Society Press, 1999.
12. Zhe Dang and R. A. Kemmerer. Using the ASTRAL Model Checker to Analyze Mobile IP. In *Proceedings of the 1999 International Conference on Software Engineering (ICSE'99)*, pages 132–141. IEEE Computer Society Press / ACM Press, 1999.
13. Zhe Dang and R. A. Kemmerer. Three approximation techniques for ASTRAL symbolic model checking of infinite state real-time systems. In *Proceedings of the 2000 International Conference on Software Engineering (ICSE'00)*, pages 345–354. IEEE Computer Society Press, 2000.
14. S. Ginsburg and E. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. of Mathematics*, 16:285–296, 1966.
15. E. M. Gurari and O. H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. *Journal of Computer and System Sciences*, 22:220–229, 1981.
16. G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997. Special Issue: Formal Methods in Software Practice.
17. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
18. O. H. Ibarra, T. Jiang, N. Tran, and H. Wang. New decidability results concerning two-way counter machines. *SIAM J. Comput.*, 24:123–137, 1995.
19. O. H. Ibarra, J. Su, Zhe Dang, T. Bultan, and R. A. Kemmerer. Counter machines: decidable properties and applications to verification problems. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000)*, volume 1893 of *Lecture Notes in Computer Science*, pages 426–435. Springer-Verlag, 2000.
20. K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.
21. M. Minsky. Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines. *Ann. of Math.*, 74:437–455, 1961.
22. A. P. Sistla and E. M. Clarke. Complexity of propositional temporal logics. *Journal of ACM*, 32(3):733–749, 1983.
23. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings 1st Annual IEEE Symp. on Logic in Computer Science, LICS'86, Cambridge, MA, USA, 16–18 June 1986*, pages 332–344, Washington, DC, 1986. IEEE Computer Society Press.