International Journal of Foundations of Computer Science
© World Scientific Publishing Company

ON REACHABILITY AND SAFETY IN INFINITE-STATE SYSTEMS

OSCAR H. IBARRA, TEVFİK BULTAN, and JIANWEN SU
Department of Computer Science, University of California
Santa Barbara, CA 93106, USA

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

We introduce some new models of infinite-state transition systems. The basic model, called a (reversal-bounded) counter machine (CM), is a nondeterministic finite automaton augmented with finitely many reversal-bounded counters (i.e. each counter can be incremented or decremented by 1 and tested for zero, but the number of times it can change mode from nondecreasing to nonincreasing and vice-versa is bounded by a constant, independent of the computation). We extend a CM by augmenting it with some familiar data structures: (*i*) A pushdown counter machine (PCM) is a CM augmented with an unrestricted pushdown stack. (*ii*) A tape counter machine (TCM) is a CM augmented with a two-way read/write worktape that is restricted in that the number of times the head crosses the boundary between any two adjacent cells of the worktape is bounded by a constant, independent of the computation (thus, the worktape is finite-crossing). There is no bound on how long the head can remain on a cell. (*iii*) A queue counter machine (QCM) is a CM augmented with a queue that is restricted in that the number of alternations between non-deletion phase and non-insertion phase on the queue is bounded by a constant. A non-deletion (non-insertion) phase is a period consisting of insertions (deletions) and no-changes, i.e., the queue is idle. We show that emptiness, (binary, forward, and backward) reachability, nonsafety, and invariance for these machines are solvable. We also look at extensions of the models that allow the use of linear-relation tests among the counters and parameterized constants as "primitive" predicates. We investigate the conditions under which these problems are still solvable.

Keywords: automated verification, infinite-state systems, reachability, invariants

1. Introduction

Since the introduction of efficient automated verification techniques such as symbolic model-checking [1], finite-state machines have been widely used for modeling reactive systems. However, due to their limited expressiveness, finite-state models are not suitable for specifying most infinite-state systems. To overcome this limitation researchers have used

1. abstraction techniques to generate finite state abstractions of infinite-state systems [2, 3, 4],

2. semi-decision procedures which prove or disprove a property if they converge, but are not guaranteed to converge [5, 6], and

3. conservative approximation techniques which are guaranteed to converge but may not always return a definite answer [7, 8].

Another promising approach for verification of infinite-state systems is to find and identify models which can represent such systems and have decidable verification queries such as reachability, invariance, etc. It is well known that, in general, verification problems for infinite-state systems are undecidable [9]. Indeed, even for systems with only two variables (or counters) that can be incremented or decremented by 1 and tested for 0, the halting problem is already undecidable (and hence the emptiness, reachability, and other problems are also undecidable) [10]. In spite of these negative results, certain restrictions can be placed on the workings of these systems that make them amenable to analysis. Some models that have been shown to have decidable properties are: various approximations on multicounter machines [11, 12, 13], timed automata [14] (and real-time logics [15, 16, 17]), pushdown automata [18, 19, 20]. [21] studied models of hybrid systems of finite automata supplied with (unbounded) discrete data structures and continuous variables and obtains decidability results for several classes of systems with control variables and observation variables. [22] investigated discrete timed automata (i.e., timed automata with integer-valued clocks) augmented with a pushdown stack.

In this paper, we introduce some new models of infinite-state systems. The basic model, called a (reversal-bounded) counter machine (CM), is a nondeterministic finite automaton augmented with finitely many reversal-bounded counters (i.e., each counter can be incremented or decremented by 1 and tested for zero, but the number of times it can change mode from nondecreasing to nonincreasing and vice-versa is bounded by a constant, independent of the computation). This model was first introduced and studied in [23]. We extend a CM by augmenting it with one of the following data structures:

i. an unrestricted pushdown stack;

ii. a finite-crossing read/write worktape;

iii. a restricted queue.

We show that emptiness, (binary, forward, and backward) reachability, nonsafety, and invariance for these machines are solvable. We also look at extensions of the models that allow the use of linear-relation tests among the counters and parameterized constants as "primitive" predicates. We investigate the conditions under which these problems are still solvable.

Our results can be used in automated verification of infinite-state systems in the following ways:

(1) By reducing a given infinite-state system to one of the models we present in this paper, one can prove that certain verification queries for the given system

are decidable and can be verified without any abstractions or approximations (for example, this approach was used in [22] to show the decidability of binary reachability problem for discrete pushdown timed automata by reducing it to the emptiness problem for pushdown counter machines);

(2) By restricting the behaviors of a given infinite-state system using properties such as reversal-boundedness one can obtain a conservative approximation of the given system (in the sense that when an error is found in the restricted system this implies that the error exists in the original system); and

(3) The proofs of decidability of properties such as emptiness and reachability can be used as a basis for algorithms for verification of such properties.

The remainder of the paper is organized as follows. Section 2 defines the new models and summarizes the main results. Section 3 gives proof sketches. Section 4 looks at some generalizations of the models. Section 5 is a brief conclusion.

2. The Models and Main Results

Many verification problems for systems that can be modeled by (finite- or infinite-state) automata can often be reduced to the emptiness problem: Given a machine M, does it accept at least one input? Decidability (existence of an algorithm) of emptiness can lead to decidability of questions such as reachability, nonsafety, invariance, containment, and equivalence, which are at the heart of verification procedures. While these problems are decidable for finite-state systems, they are, in general, undecidable for infinite-state systems. However, with appropriate restrictions, some infinite-state models have been shown to have decidable properties.

In our discussion of computational models below, we consider two types of machines: one with a one-way read-only input tape and the other without an input tape (or, equivalently, the input is always null, i.e., ϵ). When we are interested in "language recognition/acceptance", we consider machines with input. In verification problems (reachability, safety, etc.), we use the machines mostly as systems specifications rather than language recognizers, since the interest is more on the "behaviors" they generate; so these machines have no input. However, machines with input tape are also of interest for "parametric" systems, where the parameters can be specified on the input tape (we discuss this in Section 4). Also, many of the proofs of the results concerning verification problems on machines without an input tape are reductions to the decidability of the emptiness for machines with an input tape. It will be clear from the context, which type of machine we are dealing with.

The basic model is a nondeterministic finite-state machine augmented with k "reversal-bounded" counters (for some k). Thus, each counter can be incremented or decremented by 1 and tested for zero and is reversal-bounded in that the number of alternations between nondecreasing mode and nonincreasing mode is bounded by a constant, independent of the computation. Without loss of generality, we assume that the counters can only store nonnegative integers, since the finite-state control

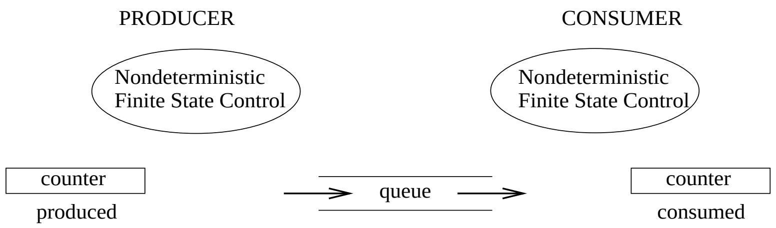

Figure 1: A producer-consumer system with an unbounded buffer

can remember the signs of the numbers. We call this basic model a CM. We can generalize the CM by augmenting it with some well-known data structures:

1. A pushdown counter machine (PCM) is a CM augmented with an unrestricted pushdown stack.
2. A tape counter machine (TCM) is a CM augmented with a two-way read/write worktape that is restricted by the following property: the number of times the head crosses the boundary between any two adjacent cells of the worktape is bounded by a constant, independent of the computation (i.e. the worktape is finite-crossing). There is no bound on how long the head can remain ("sit") on a cell.
3. A queue counter machine (QCM) is a CM augmented with a queue that is restricted in that the number of alternations between non-deletion phase and non-insertion phase on the queue is bounded by a constant. A non-deletion (non-insertion) phase is a period consisting of insertions (deletions) and no-changes, i.e., the queue is idle.

Convention: By convention, throughout, CM, PCM, ... will refer to a machine *without* an input tape. When they have a one-way read-only input tape, they will be called CM acceptor, PCM acceptor, ... In this case the computation starts with all counters zero and the pushdown stack (worktape/queue) empty (blank). Unless otherwise specified, all machines are nondeterministic.

Example 1: We give an example of a system that can be modeled by a QCM. Consider the producer-consumer system given in Fig. 1. The finite state control of the producer has a *produce* state, and the finite state control of the consumer has a *consume* state. When the producer is in *produce* state it can take the *write* transition which increments the counter *produced* by one and writes an item from a queue alphabet $\{a, b, c, d\}$ to the FIFO *queue*. When the consumer is in *consume* state and the queue is not empty, it can take the *read* transition which increments the counter *consumed* by one, removes an item from the FIFO *queue*, and stores the symbol read from the FIFO *queue* in its finite state control. Note that counters *produced* and *consumed* are reversal bounded since they are nondecreasing. If we limit the behavior of the *queue*, so that the alternations between non-write and non-read phases are bounded by a constant, this system can be represented by a

QCM. Note that this limitation does not bound the size of the *queue*, or counters. We can effectively verify properties such as invariance for this restricted system (see Example 2). If we find out that the restricted system does not satisfy an invariant, this implies that the unrestricted system does not satisfy it either.

Clearly, QCMs can be effectively simulated by TCMs. Thus, decidability results for TCMs apply to QCMs. PCMs and TCMs are more powerful than CMs. A PCM is incomparable with a TCM (note that the pushdown in a PCM is unrestricted, i.e., there is no restriction on the number of alternations between non-popping and non-pushing). The restriction that the worktape of a TCM is finite-crossing is necessary; otherwise, the tape would be equivalent to a Turing machine (TM). Similarly, the restriction on the QCM is necessary; otherwise, the queue would be equivalent to a TM. It is also easy to see that a QCM with two restricted queues can simulate a TM.

Define a configuration of a PCM to be a string of the form:

$$0^q\#0^{i_1}\#0^{i_2}\#...\#0^{i_k}\#w$$

where q in $\{1,2,...,n\}$ represents the state, 0^{i_j} represents the value of counter j (in unary), and w represents the contents of the stack, with the rightmost symbol of w the top of the stack. For certain problems, we can have w given in reversed order, i.e., the leftmost character of w is the top of the stack.

For a TCM, since the worktape is two-way read/write, the tape contents w in the configuration has to indicate the position of the read/write head within the tape. For a QCM, the left (right) end w corresponds to the *rear* (*front*) of the queue. For a CM, the configuration has no tape and is of the form $0^q\#0^{i_1}\#0^{i_2}\#...\#0^{i_k}$.

TCM (PCM, QCM) acceptors are quite powerful. For example, a TCM acceptor (even a much restricted version) M can accept the language over the alphabet $\{a,b,c,d\}$ consisting of all strings $x\#x$ such that x has the property that the sum of the lengths of all runs of c's occurring between pairs of symbols a and b (in this order) equals the number of d's. For example, $x = dacbacaccbdd$ satisfies the property, but $x = ddacbacaccbdd$ does not. M has one counter and operates in the following manner. Given input $x\#y$, M copies x onto the worktape and checks that $x = y$ and resets the worktape head to the left end of x. It computes the *sum* in its counter by looking at the worktape and whenever it sees an a, it first checks that there is a matching b to the right and that all symbols in-between are c's. It then moves left (to a), adding the length of the run of c's to the counter. The process is repeated until the whole string has been examined. (So far, M crosses any boundary between two adjacent cells on the worktape at most 7 times.) M then resets the worktape head to the left end of the tape and checks that the number of d's is equal to the *sum* in the counter. Thus, M is 9-crossing, although the worktape head makes an unbounded number of (left-to-right and right-to-left) turns, i.e., it is not finite-turn. Note also that M does not re-write the worktape and is deterministic.

Even a CM is quite powerful. For example, let L be the language consisting of all strings $x\#y\#z$, such that x, y, z are pair-wise distinct binary strings. A CM acceptor M with 3-counters, each making exactly one reversal, can accept L. M

uses one counter to check that x is different from y, a second counter to compare x and z, and a third counter to check that y is different from z. To verify that x is different from y, M "guesses" a position of discrepancy (within the string x). It does this by incrementing the first counter by 1 for every symbol it encounters while moving right on x, and nondeterministically terminating the counting at some point, guessing that a position of discrepancy has been reached. M records in its finite-control the symbol in that position. M uses the value in the counter to arrive at the same location within y where a discrepancy was guessed to occur. The second and third counters are used in a similar way to compare x with z and y with z.

Decidability/complexity results concerning CMs have been obtained in [23, 24]. These results were used recently to prove the decidability (and derive the complexity) of some decision problems (containment, equivalence, disjointness, etc.) for database queries with linear constraints [25, 26].

CMs, PCMs, QCMs, and TCMs (these have no inputs!) can generate rather complex behaviors. For example, one can show that a PCM with one counter can start in its initial state with an empty pushdown stack and reach a configuration where the pushdown contains a string x with the property described in the first example above.

The main results of the paper are the following.

1. (*Emptiness*) The emptiness problem for a class of machine acceptors is the problem of deciding, given a machine M in the class, whether the language $L(M)$ accepted by M is empty. We can show that the emptiness problems for PCM and TCM acceptors are decidable.

2. (*Binary-Reachability*) If M is a PCM (TCM), define its binary reachability set, $R(M)$, to be set of all pairs (α, β) of configurations such that α can reach β in 0 or more transitions. (Note that a pair of configurations can be represented as a string. For a PCM, the stack word component of β is written in reverse.) We can effectively construct, given a PCM (TCM) M, a PCM(TCM) acceptor accepting $R(M)$.

3. (*Forward-Reachability*) Let M be a PCM (TCM) and S be a set of configurations accepted by a CM acceptor. We can effectively construct a PCM (TCM) acceptor accepting $F_M(S)$ = the set of all configurations of M that are reachable from configurations in S in 0 or more transitions.

4. (*Backward-Reachability*) Let M be a PCM (TCM) and S be a set of configurations accepted by a CM acceptor. We can effectively construct a PCM (TCM) acceptor accepting $B_M(S)$ = the set of all configurations of M that can reach configurations in S in 0 or more transitions.

5. (*Nonsafety*) We can effectively construct, given a PCM (TCM) M and two sets of configurations I (*initial set*) and B (*bad set*) accepted by CM acceptors, a PCM (TCM) M' that accepts a configuration α if and only if (i) α is in I, and (ii) M when started in α can reach a configuration in B. Thus nonsafety is decidable.

6. (*Invariance*) We can effectively construct, given a PCM (TCM) M and two sets of configurations I (*initial set*) and G (*good set*) accepted by a CM acceptor and a deterministic CM acceptor respectively, a PCM (TCM) M' that accepts a configuration α if and only if (i) α is in I, and (ii) M when started in α can reach a configuration not in G. Thus invariance is decidable.

Obviously, the above results hold for CMs and, as previously observed, QCMs can effectively be simulated by TCMs; so the results hold for these machines as well. In contrast, emptiness is undecidable for PCMs and TCMs with multiple pushdown stacks (finite-crossing worktapes). In fact, it follows from the undecidability of the Post Correspondence Problem [27] that emptiness is undecidable for machines with only two pushdown stacks, even if the stacks are restricted to making only one alternation from non-popping to non-pushing.

Example 2: Consider again the producer-consumer system given in Figure 1 and the QCM M that is constructed from it by restricting the behavior of the *queue*. An invariance property for this system can be defined as follows: I is defined as the set of configurations where both *produced* and *consumed* are 0 and the *queue* is empty, and G is defined as the set of configurations where *produced* $-$ *consumed* is equal to the number of items in the *queue*, and there are at most as many a's as b's and at least as many c's as d's. We want to verify that for all the configurations in I (which represent the initial configurations of the system) the set of reachable configurations is contained in G, i.e., G is an invariant. We can construct deterministic CM acceptors for both these sets; hence, we can construct a QCM M' which will recognize all the configurations in I which can reach a configuration that is not in G. If the language accepted by M' is not empty, this means that M (and the corresponding unrestricted system) has an error. If the language accepted by M' is empty, however, this only proves that the restricted system is correct. It does not prove the correctness of the unrestricted system. Hence, the restricted system is a conservative approximation, in the sense that there are no false negatives but there could be false positives. If a given input system can be reduced to a QCM both negative and positive results would be exact.

Example 3: Consider a CM M, and suppose we are interested in the set T of pairs of configurations (α, β) of M such that there is a computation path (i.e., sequence of configurations) from α to β that satisfies a property that can be verified by a PCM (QCM, TCM) acceptor. Then T is computable and can be accepted by a PCM (QCM, TCM) acceptor. For example, suppose that the property is for the path to contain two non-overlapping subpaths (i.e., segments of computation) which go through the same sequence of states, and the length of the subpath is no less than a third of the length of the entire path. Clearly, T can be accepted by a QCM acceptor or by a TCM acceptor.

We also study extensions of PCM and TCM acceptors in Section 4. In particular, we look at acceptors that allow as "primitive" predicates the use of linear-relation tests among the counters and parameterized constants, e.g., tests like "Is $3x - 5y + 11z - 2D_1 + 9D_2 < 12$?", where x, y, z are counters and D_1 and D_2 represent parameterized constants whose domain is the set of all integers $(+,-,0)$. Note that directly implementing such tests using the standard "testing for zero" would result in a machine that is not reversal-bounded. We investigate the conditions under which the emptiness problem and other problems are decidable for these extended models.

3. Proof Sketches

We now give proof sketches of the results. We start with the emptiness problem.

Theorem 1 *The emptiness problem for PCM acceptors is decidable.*

Proof. This result was already shown in [23]. For completeness, we describe the idea of the proof. Let A be an alphabet consisting of k symbols $a_1, ..., a_k$, $\mathbb{N}$ the set of nonnegative integers. For each string (word) w in A^*, we define

$f(w) = (i_1, ..., i_k)$, where i_j is the number of occurrences of a_j in w.

If L is a subset of A^*, we define $f(L) = \{f(w) \mid w \in L\}$.

In [23], it was shown that if M is a PCM acceptor with the input alphabet A, then $f(L(M))$ is an effectively computable semilinear set (or, equivalently, definable by a Presburger formula), where $L(M)$ is the language accepted by M. Hence, $L(M)$ is empty iff $f(L(M))$ is empty, which is decidable since it is Presburger. $\square$

Obviously, the above theorem holds for machines with no pushdown stack:

Corollary 1 *The emptiness problem for CM acceptors is decidable.*

It has been shown in [24] that the emptiness problem for CM acceptors 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 PCM acceptors.

To prove the decidability of the emptiness problem for TCM acceptors, we need some lemmas.

Lemma 1 *Let M be a TCM acceptor. We can effectively construct a TCM acceptor M' such that $L(M) = L(M')$ and M' is non-sitting meaning that in any computation, its read/write head does not sit on any tape cell (i.e., it always moves left or right of a cell in every step).*

Proof. Note that M' cannot just simulate a sitting step by a left (or right) move followed by a right (or left) move. This is because the read/write head can sit on a cell an unbounded number of steps, and this would make M' not finite-crossing.

What M' can do is to use a new "dummy" symbol, say #. M' begins the simulation of M by writing a finite-length sequence of #'s on the worktape, the length being chosen nondeterministically. M' simulates M, but whenever M writes a symbol on a *new* tape cell, M' also writes to the right of this cell a finite-length sequence of #'s (again the length is chosen nondeterministically). Thus, at any time, the worktape contains a string where every pair of non-dummy symbols (i.e. symbols in the worktape alphabet of M) is separated by a string of #'s. During the simulation, M' uses moves on the #'s to simulate the sitting moves of M (this is possible if there are enough #'s between any pair of non-dummy symbols). To simulate a nonsitting move of M, M' may need to "skip over" the #'s to get to the correct non-dummy symbol. Clearly, M' is non-sitting and accepts $L(M)$. $\square$

Lemma 2 *Let M be a TCM acceptor. We can effectively construct a TCM M' (with no input tape!) such that $L(M)$ is nonempty if and only if M' when started with a blank worktape and zero counters has a halting sequence of moves. (Note that since the machine is nondeterministic, not all sequences of moves may halt.)*

Proof. The construction of M' is straightforward. In the simulation of M, M' nondeterministically guesses the symbols comprising the input tape. $\square$

Theorem 2 *The emptiness problem for TCM acceptors is decidable.*

Proof. From Lemma 2, we need only show that the halting problem for TCMs is decidable. Let M be a TCM. By Lemma 1, we assume that M is non-sitting (note that the lemma obviously holds for TCMs without an input). We may also assume, without loss of generality, that each counter of M makes exactly one reversal (since a counter making k reversals can be simulated by $(k+1)/2$ counters, each making exactly one reversal), M halts with all the counters zero and the worktape head at the right end of the tape, and that in any computation, every counter becomes positive.

Consider a halting sequence of moves of M and look at position (cell) p of the worktape, $p = 1, 2, ...n$, for some n. In the computation, position p will be visited many times. Let $t_1, ..., t_m$ be the times M visits p.

Corresponding to the time sequence $(t_1, ..., t_m)$ associated with position p, we define a *crossing vector* $R = (I_1, ..., I_m)$, where for each i, $I_i = (d_1, q_1, r_1, r_2, d_2)$,

1. d_1 is the direction from which the head entered p at time t_i;
2. q_1 is the state when it entered p;
3. r_1 is the instruction that was used in the move above;
4. r_2 is the instruction that was used at time $t_i + 1$ when it left p;
5. d_2 is the direction from which it left p at time $t_i + 1$.

We construct a TCM M' which simulates a halting computation of M by nondeterministically guessing the sequence of crossing vectors $R_1, ..., R_n$ as it processes the worktape from left to right, making sure that R_i and R_{i+1} are compatible for $1 \leqslant i \leqslant n$. Corresponding to each counter C of M, the machine M' uses two counters C_1 and C_2. C_1 is used to record the increases in C, while C_2 is used to record the decreases in C. When M' completes the simulation of M, C_1 and C_2 must contain the same value, and this can easily be checked by M'.

The theorem follows from Corollary 1 since deciding whether a TCM (which has no input tape) has a halting sequence of moves is easily reducible to deciding the emptiness problem for a CM. $\square$

The restriction that the worktape in a TCM is finite-crossing is necessary; otherwise (i.e., if it is unrestricted), the machine becomes a Turing machine. In fact, even for a special case, emptiness is undecidable. Restrict the worktape to be a pushdown stack which can only push (i.e., write) but *cannot* pop (i.e., erase), but can enter the stack in a read-only mode. Moreover, once it enters the stack in a read-only mode, it can no longer push. There is no restriction on the number of times the stack head can cross the boundary between any two stack cells. This restricted worktape is called a "checking" tape. Call this machine CCM. (CCM acceptors without counters have been studied in [28].)

Theorem 3 *The emptiness problem for CCM acceptors is undecidable.*

Proof. The proof uses the undecidability of Hilbert's Tenth Problem (HTP) [29], which is to decide for a given polynomial $p(x_1, ..., x_n)$ with integer coefficients whether it has a nonnegative integral root. We omit the construction, but the idea is to show that we can effectively construct, given a polynomial $p(x_1, ..., x_n)$, a CCM acceptor M_p such that p has no integral solution if and only if $L(M_p)$ is empty. $\square$

We consider next the binary reachability.

Theorem 4 *We can effectively construct, for a given PCM M, a PCM acceptor M' accepting $R(M) =$ the set of all pairs (α, β) of configurations such that α can reach β in 0 or more transitions.*

Proof. Given a PCM M, we construct a PCM acceptor M' that accepts $R(M)$. M' when given (α, β), reads the configuration α and sets its counters and pushdown stack to α. Then M' simulates the computation of M starting in this configuration. At some point M' guesses that it has reached the configuration β, which it can verify by reading the input (note that since the pushdown is last-in-first-out, β must have the stack word given in reverse). $\square$

The same construction works for a TCM acceptor, but the stack word in configuration β does not have to be given in reversed order, since the read/write head is two-way.

Theorem 5 *We can effectively construct, for a given TCM M, a TCM acceptor M' accepting $R(M)$ ($R(M)$ is defined in Theorem 4).*

We now look at nonsafety.

Theorem 6 *We can effectively construct, given a PCM (TCM) M and two sets of configurations I (*initial set*) and B (*bad set*) accepted by CM acceptors, a PCM (TMC) M' that accepts a configuration α iff (i) α is in I, and (ii) M when started in α can reach a configuration in B. Thus nonsafety is decidable.*

Proof. We only prove the PCM case; the proof for TCM is similar. Let M_I and M_B be CM acceptors accepting I and B, respectively. We construct a PCM acceptor M' which, when given an input α, sets its counters and pushdown stack to this configuration while also checking that the configuration is accepted by M_I . (Note that this can be done with additional counters *without* popping the stack). Then M' simulates the computation of M starting in this configuration. At some point M' guesses that it has reached a configuration β in B, which it can verify by simulating M_B on β. In the simulation of M_B, M' uses the pushdown stack which contains w to simulate the action of M_B on this input. M' reads w from the right by "popping". There is a slight problem in that M' will be working on the reverse of w, not w. However, it can be shown that if a language is accepted by a CM acceptor, then its reverse can also be accepted by a CM acceptor [23]. Hence, we can use the CM acceptor accepting the reverse of the language accepted by M_B in the computation of M' to decide if β is in B. □

Corollary 2 *We can effectively construct, given a PCM (TCM) M and two sets of configurations I (*initial set*) and G (*good set*) accepted by a CM acceptor and a deterministic CM acceptor, respectively, a PCM (TMC) M' that accepts a configuration α iff (i) α is in I, and (ii) M when started in α can reach a configuration not in G. Thus invariance is decidable.*

Proof. It can be shown that if G is accepted by a deterministic CM acceptor M_G, then we can effectively construct a deterministic CM acceptor accepting the set of bad configurations $B =$ the complement of G. (Note that this is not true if M_G is not deterministic.) The result follows from the above theorem. □

Next, we show that forward reachability is computable.

Theorem 7 *We can effectively construct, given a PCM (TCM) M and a set of configurations S accepted by a CM acceptor, a PCM (TCM) acceptor accepting $F_M(S) =$ the set of all configurations that can be reached from configurations in S in 0 or more transitions.*

Proof. Let M be a PCM and S be a set of configurations accepted by a CM acceptor M_S. We construct a PCM acceptor M', which when given a configuration β (with the stack word given in reverse), nondeterministically guesses a configuration α by simultaneously setting its counters and pushdown stack to this configuration and checking that α is in S. Then M' simulates the computation of M starting in this configuration. At some point, M' guesses that it has reached β, which it can check by reading the input. The proof for the case of TCM M is similar. □

Theorem 8 *We can effectively construct, given a PCM (TCM) M and a set of configurations S accepted by a CM acceptor, a PCM (TCM) acceptor accepting $B_M(S) =$ the set of all configurations that can reach configurations in S in 0 or more transitions.*

Proof. Let M be a PCM and S be a set of configurations accepted by a CM acceptor M_S. We construct a PCM acceptor M', which when given a configuration α, sets its counters and pushdown stack to this configuration. Then M' simulates the computation of M starting in this configuration. At some point M' guesses that it has reached a configuration in S (which is accepted by M_S) and verifies this as in the proof of Theorem 6. The proof for the case of TCM M is similar. □

Clearly, all the results above hold for CMs and QCMs. In fact, some of the results can be strengthened for CMs, for example:

Theorem 9 *Let M be a CM and S be a set of configurations. Then $B_M(S)$ is accepted by a CM acceptor iff S is accepted by a CM acceptor ($B_M(S)$ is defined in Theorem 8).*

Proof. The "if part" follows from the construction in Theorem 8. Conversely, given M and S, suppose $B_M(S)$ is accepted by a CM acceptor M'. We show that S can be accepted by a CM acceptor M''.

Given β on its input tape, M'' guesses and stores in $k+1$ counters a configuration α. By using additional counters and employing M', M'' checks that α is in $B_M(S)$. Then M'' simulates M to check that α can reach β. $\square$

We say that a set of configurations S of a CM, which are strings of the form $0^q\#0^{i_1}\#0^{i_2}\#...\#0^{i_k}$, is Presburger if the set of $(k+1)$-tuples $(q, i_1, ..., i_k)$ corresponding to the configurations in S is definable by a Presburger formula.

It known that a set of configurations is accepted by a CM acceptor if and only if it is Presburger [23]. Hence, we have:

Corollary 3 *Let M be a CM and S be a set of configurations. Then $B_M(S)$ is Presburger if and only if S is Presburger.*

The last two results above also hold for $F_M(S)$.

4. Extensions of the Models

In this section we look at some extensions of PCM (TCM) acceptors. We will only deal with the emptiness problem since the other problems (reachability, safety, etc.) are reducible to emptiness, as we have seen in the previous section. The proofs of the theorems below are generalizations of the proofs of similar results for the corresponding extensions of CMs in [30]. Related results can be found in [13], where decidability of reachability problems for some classes of two counter machines which allow resetting a counter to zero and transferring the value of one counter to another one were shown.

The first extension is to allow the counters of a PCM (TCM) to store negative numbers, and to allow the counters to increment/decrement by c, and also to allow tests of the form: "Is $x\theta c$?", where x is a counter, c is any integer constant (there are many such constants in the specification of the machine), positive, negative, or zero, and θ is one of $<, >, =$.

One can easily show that any PCM (TCM) M that uses the generalized instructions above can be converted to an equivalent machine M' using standard instructions such that $L(M) = L(M')$. The construction of M' is straightforward. M' "remembers" the signs of the counters in the states, so the counters do not have to store negative values. To handle predicates like $x < c$, M' uses fixed-size "buffers" in the states to translate the origin, etc.

Next, consider a PCM (TCM) that allows tests like, "Is $5x - 3y + 2z < 7$?", where x, y are counters. To be precise, let V be a finite set of variables over integers. An *atomic linear relation* on V is defined as

$$\sum_{v \in V} a_v v < b,$$

where a_v for all $v \in V$ and b are integers. A *linear relation* on V is constructed from a finite number of atomic linear relations using $\neg$ and $\wedge$. Note that standard symbols like $>, =, \rightarrow$ (implication), $\vee$ can also be expressed using the above constructions.

Suppose we allow a PCM (TCM) M to use tests of the form: " Is L ", where L is a linear relation on the counters. The emptiness problem becomes undecidable, even for deterministic CMs. In fact, the following can be shown using the undecidability of the halting problem for unrestricted two counter machines [10]:

> Consider only deterministic CMs (no input) with 3 counters (initially zero) which can only be incremented by 0 or 1 (thus decrementing is *not* allowed), and the only tests are of the form "Is $x = y$?", where x, y are counters. The halting problem for such machines is undecidable.

Note that in the above undecidability, the 3-counter CM is 0-reversal bounded because the mode of each of the counters is always nondecreasing. Interestingly, the emptiness problem is decidable for CM acceptors with only two reversal-bounded counters using the standard instructions and the test "Is $x = y$?". This follows from Theorem 1 and the observation that the pushdown stack can be used to keep track of the difference $x - y$.

In defining reversal-boundedness, we only had two modes: nondecreasing and nonincreasing. Suppose we refine these to three modes: increasing, decreasing, no-change. Say that a counter is *mode-bounded* if the number of changes in its mode during any computation is bounded by a constant. Clearly, a counter that is mode-bounded is reversal-bounded, but the converse is not true. For example, a counter that displays the pattern "$122334455\cdots$" correspond to 0-reversal, but is not mode-bounded (since although it is nondecreasing, the number of changes from no-change to increasing and vice-versa is unbounded). Call a PCM (TCM) mode-bounded if the counters are mode-bounded. We can show the following using the techniques in [30] where the results were shown for (mode-bounded) CM acceptors:

Theorem 10 *The emptiness problem is decidable for mode-bounded PCM (TCM) acceptors that can use linear-relation tests on the counters.*

We can further generalize the machines by allowing parameterized constants in the linear relations. So for example, we can allow tests like "Is $3x-5y-2D_1+9D_2 < 12$?", where D_1 and D_2 represent parameterized constants whose domain is the set of all integers $(+, -, 0)$. We can specify the values of these parameters by including them in the input tape. Thus, the input of the machine with t parameterized constants be of the form: "$\#d_1\%\cdots\%d_k\%w\#$", where $d_1, ..., d_k$ are integers $(+, -, 0)$ that the parameterized constants $D_1, ..., D_k$ assume for this run, and % is a separator. We assume that the d_i's are represented in unary along with their signs. We can prove the following:

Theorem 11 *The emptiness problem is decidable for mode-bounded PCM (TCM) acceptors that can use linear-relation tests on the counters and parameterized constants.*

As we have seen, Theorem 10 is not true for machines whose counters are reversal-bounded but not mode-bounded. However, suppose we require that in every linear- relation L, every atomic linear-relation in L involves only the parameterized constants and at most one counter so, e.g., $4D_1 + 9D_2 < 7$ and $5x - 4D_1 + 9D_2 < 7$ are allowable, but $5x + 2y - 4D_1 + 9D_2 < 7$ is not (where x and y are counters, and D_1 and D_2 are parameterized constants). Call such a relation L a *restricted linear-relation.* Then we can prove:

Theorem 12 *The emptiness problem is decidable for PCM (TCM) acceptors that can use restricted linear-relation tests on the counters and parameterized constants.*

5. Conclusions

We introduced some new models of infinite-state transition systems by augmenting the finite-state machine with reversal-bounded counters and suitably restricted data structures (such as pushdown stack, queue, read/write tape) and showed that emptiness, (binary, forward, backward) reachability, nonsafety, and invariance for these models are solvable. We also studied generalizations of the models.

As we have seen in Examples 2 and 3, the results can be used to verify properties that are not verifiable using previous techniques. We give some more examples below.

For a configuration α, α_{x_i} and α_w denote the value of counter x_i and the stack word (or worktape contents) of α, respectively. $\#_a(w)$ denotes the number of occurences of symbol a in a stack word (or worktape contents) w.

Consider the following property concerning a PCM M:

> *For any pair of configurations* (α, β) *if* α *reaches* β, *then* $(\beta_{x_2} = \alpha_{x_1} + 2\alpha_{x_2} \wedge 2\#_a(\beta_w) = 3\#_b(\alpha_w))$.

This property can be verified, by showing that its negation can be verified. From Theorem 4, $R(M)$ can be accepted by a PCM acceptor M'. We construct a PCM acceptor M'' which simulates M' and at the same time (using additional counters) checks that $(\beta_{x_2} = \alpha_{x_1} + 2\alpha_{x_2} \wedge 2\#_a(\beta_w) = 3\#_b(\alpha_w))$ is false. Then $L(M'')$ is empty if and only if the property is false. Hence, the property can be verified. Note that:

- Even without counters, $2\#_a(\beta_w) = 3\#_b(\alpha_w)$ defines a nonregular set of stack word pairs. Hence, this property cannot be verified by the model checking procedures for pushdown systems [18, 19, 20].
- Even without the pushdown stack, $\beta_{x_2} = \alpha_{x_1} + 2\alpha_{x_2}$ is not a "clock region" [14]. Hence, the classical region technique cannot verify this property. This is also pointed out in [31].

As another example, since invariance is decidable from Corollary 2, we can verify the following property concerning a TCM M:

> *Starting from a configuration* α *satisfying:* $\#_a(\alpha_w) = \#_b(\alpha_w)$, M *can only reach a configuration* β *satisfying:* $(\beta_{x_1} = 2\beta_{x_1} + \beta_{x_2} \wedge \#_a(\beta_w) = 3\#_b(\beta_w))$.

The reason is that the set of α's and β's satisfying the stated properties can be accepted by deterministic CM acceptors.

In the future we would like to investigate the decidability of liveness properties for the computational models we presented in this paper. More generally we would like to consider decidability of various temporal logic properties such as CTL, LTL, and μ-calculus. We would also like to investigate the complexity of verification procedures for these infinite-state models.

Acknowledgements

The work by Oscar H. Ibarra and Jianwen Su have been supported in part by NSF grants IRI-9700370 and IIS-9817432. The work by Tevfik Bultan has been supported in part by NSF grant CCR-9970976 and NSF CAREER award CCR-9984822.

References

1. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. H. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information & Computation*, **98(2)** (1992) 142–170.
2. J. Dingel and T. Filkorn, "Model checking for infinite state systems using data abstraction," *Proc. of 7th Int. Conf. on Computer Aided Verification*, Liege, Belgium, July 1995, pp. 54–69.
3. D. Dams, R. Gerth, and O. Grumberg, "Abstract interpretation of reactive systems," *ACM Trans. on Programming Languages and Systems*, **19(2)** (1997) 253–291.
4. S. Bensalem, Y. Lakhnech, and S. Owre, "Computing abstractions of infinite state systems compositionally and automatically," *Proc. of 10th Int. Conf. on Computer Aided Verification*, Vancouver, BC, Canada, June 1998, pp. 319–331.
5. B. Boigelot and P. Godefroid, "Symbolic verification of communication protocols with infinite state spaces using QDDs," *Proc. of 8th Int. Conf. on Computer Aided Verification*, New Brunswick, NJ, USA, July 1996, pp. 1–12.
6. P. Wolper and B. Boigelot, "Verifying systems with infinite but regular state spaces," *Proc. of 10th Int. Conf. on Computer Aided Verification*, Vancouver, BC, Canada, June 1998, pp. 88–97.
7. N. Halbwachs, P. Raymond, and Y. Proy, "Verification of linear hybrid systems by means of convex approximations," *Proc. of First Int. Static Analysis Symposium*, Namur, Belgium, September 1994, pp. 223–237.
8. T. Bultan, R. Gerber, and W. Pugh, "Model Checking Concurrent Systems with Unbounded Integer Variables: Symbolic Representations, Approximations, and Experimental Results," *ACM Trans. on Programming Languages and Systems*, **21(4)** (1999) 747–789.
9. J. Esparza, "Decidability of Model Checking for Infinite-State Concurrent Systems," *Acta Informatica*, **34(2)** (1997) 85–107.
10. M. Minsky, "Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines," *Ann. of Math.*, **74** (1961) 437–455.
11. H. Comon and Y. Jurski, "Multiple counters automata, safety analysis and Presburger arithmetic," *Proc. of 10th Int. Conf. on Computer Aided Verification*, Vancouver, BC, Canada, June 1998, pp. 268–279.
12. B. Boigelot and P. Wolper, "Symbolic verification with periodic sets," *Proc. of 6th Int. Conf. on Computer Aided Verification*, Stanford, CA, USA, June 1994, pp. 55–67.
13. A. Finkel and G. Sutre, "Decidability of reachability problems for classes of two counters automata," *Proc. of 17th Int. Symposium on Theoretical Aspects of Com-*

puter Science, Bratislava, Slovakia, August 2000, pp. 346–357.

14. R. Alur and D. Dill, "Automata for modeling real-time systems," *Theoretical Computer Science*, **126(2)** (1994) 183–236.
15. R. Alur and T. A. Henzinger, "A really temporal logic," *J. ACM*, **41(1)** (1994) 181–204.
16. R. Alur, C. Courcoibetis, and D. Dill, "Model-checking in dense real time," *Information & Computation*, **104(1)** (1993) 2–34.
17. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic Model Checking for Real-time Systems," *Information & Computation*, **111(2)** (1994) 193–244.
18. A. Bouajjani, J. Esparza, and O. Maler, "Reachability analysis of pushdown automata: Application to model-checking," *Proc. of 8th Int. Conf. on Concurrency Theory*, Warsaw, Poland, July 1997, pp. 135–150.
19. A. Finkel, B. Willems, and P. Wolper, "A direct symbolic approach to model checking pushdown systems," *Proc. 2nd Int. Workshop on Verification of Infinite State Systems*, Bologna, Italy, July 1997, pp. 30–45.
20. I. Walukiewicz, "Pushdown processes: games and model checking," *Proc. of 8th Int. Conf. on Computer Aided Verification*, New Brunswick, NJ, USA, July 1996, pp. 62–74.
21. 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, LNCS 999*, eds. P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Springer, 1995) pp.64–85.
22. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su, "Binary reachability analysis of discrete pushdown timed automata," *Proc. of 12th Int. Conf. on Computer Aided Verification*, Chicago, IL, USA, July 2000, pp. 69–84.
23. O. H. Ibarra, "Reversal-bounded multicounter machines and their decision problems," *J. ACM*, **25** (1978) 116–133.
24. E. M. Gurari and O. H. Ibarra, "The complexity of decision problems for finite-turn multicounter machines," *J. Comput. System Sci.*, **22** (1981) 220–229.
25. O. H. Ibarra and J. Su, "A technique for the containment and equivalence of linear constraint queries," *J. Comput. System Sci.*, **59(1)** (1999) 1–28.
26. O. H. Ibarra, J. Su, and C. Bartzis, "Counter machines and the safety and disjointness problems for database queries with linear constraints," in *Words, Sequences, Languages: Where Computer Science, Biology and Linguistics Meet*, eds. C. Martin-Vide and V. Mitrana, (Kluwer, 2000) pp. 127–137.
27. E. Post, "A variant of a recursively unsolvable problem," *Bull. Am. Math. Soc.*, **52** (1946) 264–268.
28. S. A. Greibach, "Checking automata and one-way stack languages," *SDC Document TM 738/045/00*, 1968.
29. Y. Matijasevic, "Enumerable sets are Diophantine," *Soviet Math. Dokl*, **11** (1970) 354–357.
30. O. H. Ibarra, J. Su, Z. Dang, T. Bultan, and R. Kemmerer, "Counter machines: Decidable properties and applications to verification problems," *Proc. of 25th Int. Symposium on Mathematical Foundations of Computer Science*, Bratislava, Slovak Republic, August 2000, pp. 426–435.
31. H. Comon and Y. Jurski, "Timed automata and the theory of real numbers," *Proc. of 10th Int. Conf. on Concurrency Theory*, Netherlands, August 1999, pp. 242–257.