# Counter Machines and Verification Problems

Oscar H. Ibarra [1], Jianwen Su [1], Zhe Dang [2], Tevfik Bultan [3], Richard A. Kemmerer [2]

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

**Abstract**

We study various generalizations of reversal-bounded multicounter machines and show that they have decidable emptiness, infiniteness, disjointness, containment, and equivalence problems. The extensions include allowing the machines to perform linear-relation tests among the counters and parameterized constants (e.g., "Is $3x-5y-2D_1+9D_2 < 12$?", where $x, y$ are counters, and $D_1, D_2$ are parameterized constants). We believe that these machines are the most powerful machines known to date for which these decision problems are decidable. Decidability results for such machines are useful in the analysis of reachability problems and the verification/debugging of safety properties in infinite-state transition systems. For example, we show that (binary, forward, and backward) reachability and safety are solvable for these machines.

*Key words:* Counter machines, automated verification, infinite-state systems, reachability

## 1 Introduction

The simplest language recognizers are the finite automata. It is well known that all varieties of finite automata (one-way, two-way, nondeterministic, etc.)

[1] The work by Oscar H. Ibarra and Jianwen Su have been supported in part by NSF grants IRI-9700370 and IIS-9817432.
[2] The work by Zhe Dang and Richard A. Kemmerer have been supported in part by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Material Command, USAF, under agreement number F30602-97-1-0207.
[3] The work by Tevfik Bultan has been supported in part by NSF grant CCR-9970976 and NSF CAREER award CCR-9984822.

are effectively equivalent, and the class has decidable emptiness, infiniteness, disjointness, containment, and equivalence problems. These problems, referred to as *F-problems*, are defined as follows, for arbitrary finite automata $M_1, M_2$:

- *Emptiness:* Is $L(M_1)$ (the language accepted by $M_1$) empty?
- *Infiniteness:* Is $L(M_1)$ infinite?
- *Disjointness:* Is $L(M_1) \cap L(M_2)$ empty?
- *Containment:* Is $L(M_1) \subseteq L(M_2)$?
- *Equivalence:* Is $L(M_1) = L(M_2)$?

When a two-way finite automaton is augmented with a storage device, such as a counter, a pushdown stack or a Turing machine tape, the F-problems become undecidable (no algorithms exist). In fact, it follows from a result in [Min61] that the emptiness problem is undecidable for two-way counter machines even over a unary input alphabet. On binary inputs, if one restricts the counter machines to make only a finite number of turns on the input tape, the emptiness problem is also undecidable, even for the case when the input head makes only one turn (i.e., change in direction) [Iba78]. However, for one-way counter machines, it is known that the equivalence (hence also the emptiness) problem is decidable, but the containment and disjointness problems are undecidable [VP75].

In this paper, we study two-way finite automata augmented with several counters. A restricted version of these machines was studied in [Iba78]. Since a counter can be incremented/decremented by 1 (and tested if it is 0), we count each alternation from non-increasing mode to nondecreasing mode or vice-versa as a *reversal*. For $k, m, r \in \mathbb{N}$ (the natural numbers), we define an $m$-crossing $r$-reversal $k$-counter machine $M$ as a two-way finite automaton with input delimiters (end-markers), augmented with $k$ counters such that on any input:

(1) No boundary between input symbols (including the delimiters) is *crossed* by the input head more than $m$ times (note that the number of turns, i.e., changes in directions, the input head makes on the input may be unbounded).
(2) Each counter makes no more than $r$ reversals.

We consider various generalizations of finite-crossing reversal-bounded multicounter machines and investigate their decision problems. The extensions include allowing the machines to perform linear-relation tests among the counters and parameterized constants (e.g., "Is $3x-5y-2D_1+9D_2 < 12$?", where $x, y$ are counters and $D_1, D_2$ are parameterized constants). We show that many classes have decidable F-problems. We believe that these machines are the most powerful machines known to date for which the decision problems are decidable. Decidability results for such machines are useful in the analysis of

reachability problems and the verification/debugging of safety properties in infinite-state transition systems. For example, we show that (binary, forward, and backward) reachability and safety are solvable for these machines.

The paper is organized as follows. Section 2 recalls the formal definition of a reversal-bounded multicounter machine. Section 3 presents the fundamental decidable problems for these machines. Section 4 looks at several generalizations of the basic model and investigates their decidable properties. Section 5 uses the results of the previous sections to show that (binary, forward, and backward) reachability and safety are solvable for these machines. Section 6 concludes with an example of how the results can be used to check a safety property in an infinite-state transition system.

## 2 Reversal-Bounded Multicounter Machines

An *input* to a two-way $k$-counter machine $M$ is a string of the form $\#w\#$, where $\#$ is the input delimiter and $w$ is in $A^*$, $A$ being the input alphabet (which does not contain $\#$). We can think of the input as being written on a tape that is divided into tape cells. The machine has an input head that can read symbols from the tape. A move or step of $M$ consists of the following.

Starting in state $p$:

(1) Read the symbol under the input head.
(2) Check the status (zero or non-zero) of the $k$ counters.

Based on the state $p$, input symbol, and counter status:

(3) Move the input head left, right, or remain on the same symbol.
(4) Increment each counter by $+1, -1$, or 0. (Counters can only store non-negative integers; decrementing a zero counter is undefined.)
(5) Enter state $q$.

If the machine is nondeterministic, there may be several choices for 3, 4, 5.

The machine starts a computation in the initial state with the input head on the left delimiter and all the counters set to zero. We assume without loss of generality that the machine does not fall off the left end of the input tape during the computation. There are two special halting states: ***accept*** and ***reject***. $M$ accepts (rejects) an input $\#w\#$ if $M$ on this input halts in state ***accept*** (respectively ***reject***). Note that the machine may not always halt (i.e., it can go into an infinite loop). The set of all inputs accepted by $M$ is denoted by $L(M)$.

$M$ is *reversal-bounded* if there is a nonnegative integer $r$ such that for any computation on any input, every counter of $M$ makes no more than $r$ reversals (alternations between nonincreasing and nondecreasing modes or vice-versa). So, for example, a counter with the following computation pattern:

000001111112222223444 has 0 reversals.

On the other hand,

00000111111222222344444333222123344 has 2 reversals.

$M$ is *finite-crossing* if there is a positive integer $m$ such that on every computation on any input, $M$'s input head crosses the boundary between any two adjacent tape cells at most $m$ times. Note that there is no bound on the number of turns the input head makes on the tape. There is also no bound on how long the head can remain (*sit*) on a symbol.

Actually, we do not need to require that $M$ be finite-crossing and the counters reversal-bounded for inputs that are not accepted. However, we can make this assumption without loss of generality since if $M$ is $m$-crossing and $r$ reversal-bounded, the finite-state control can always keep track of the number of reversals each counter makes, and $M$ rejects an input that causes a counter to make more than $r$ reversals. Moreover, we can add another counter to $M$ and initialize it (using the input) to the value $m \times n$, where $n$ is the length of the input. This counter is then decremented each time $M$ crosses a boundary during the computation. $M$ rejects if this counter becomes zero. The resulting machine will then be finite-crossing and reversal-bounded on any input (accepted or not). $M$ is *one-way* if the input head crosses the boundary between any two adjacent cells exactly once (i.e., $M$ is 1-crossing). Finite-crossing reversal-bounded multicounter machines are quite powerful as the following example shows.

Example 2.1 A deterministic 5-crossing 1-reversal 1-counter machine $M$ can accept the language over the alphabet $\{a, b, c, d\}$ consisting of all strings such 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, $M$ accepts the string "*dacbacaccbdd*" but not the string "*ddacbacaccbdd*." $M$ operates in the following manner. It computes the *sum* in its counter by looking at the input 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 input symbols at most 3 times.) $M$ then moves the input head from the right delimiter to the left delimiter and checks that the number of $d$'s is equal to

the *sum* in the counter. Finally, the input head is moved to right delimiter and the machine accepts if and only if the string is in the language. Thus, $M$ is 5-crossing, although its input head makes an unbounded number of (left-to-right and right-to-left) turns, i.e., it is not finite-turn.

## 3 Fundamental Decidable Problems

Let $\mathbb{N}$ be the set of nonnegative integers and $k$ be a positive integer. A subset $S$ of $\mathbb{N}^k$ is a *linear set* if there exist vectors $v_0, v_1, ..., v_t$ in $\mathbb{N}^k$ such that

$$S = \{v \mid v = v_0 + a_1 v_1 + \cdots + a_t v_t, \forall 1 \leqslant i \leqslant t, a_i \in \mathbb{N}\}.$$

The vectors $v_0$ (the constant vector) and $v_1, ..., v_t$ (the periods) are called *generators*. $S$ is *semilinear* if it is a finite union of linear sets. Semilinear sets are precisely the sets definable by *Presburger formulas* [GS64].

An empty set is a trivial (semi)linear set, where the set of generators is empty. Any finite subset of $\mathbb{N}^k$ is semilinear—it is a finite union of linear sets whose generators are constant vectors.

Let $A$ be an alphabet consisting of $k$ symbols $a_1, ..., a_k$. For each string (word) $w$ in $A^*$, we define the *Parikh map* of $w$, denoted by $f(w)$, as follows:

$$f(w) = (i_1, ..., i_k), \quad \text{where } i_j \text{ is the number of occurrences of } a_j \text{ in } w.$$

If $L$ is a subset of $A^*$, the *Parikh map* of $L$ is defined $f(L) = \{f(w) \mid w \in L\}$.

The following theorem is from [Par66].

**Theorem 3.1** *Let $M$ be a one-way nondeterministic finite automaton (1NFA) or a one-way nondeterministic pushdown automaton (1NPDA). Then $f(L(M))$ is a semilinear set effectively computable from $M$.*

The next result, which generalizes Theorem 3.1, was proved in [Iba78].

**Theorem 3.2** *Let $M$ be a nondeterministic one-way reversal-bounded $k$-counter machine. Then $f(L(M))$ is a semilinear set effectively computable from $M$.*

We now show (using the standard crossing-sequence technique) that nondeterministic finite-crossing machines can be converted to one-way and, therefore, have semilinear property:

**Theorem 3.3** *Let $M$ be a nondeterministic finite-crossing reversal-bounded multicounter machine. We can effectively construct a nondeterministic one-way reversal-bounded multicounter machine $M'$ such that $L(M) = L(M')$.*

**Proof:** We assume without loss of generality that each counter of $M$ makes exactly one reversal, $M$ accepts with all the counters zero and the input head falling off the right end of the tape, and that in any computation, every counter becomes positive.

Let $a_1 ... \ a_n$ be an input (here $a_1$ and $a_n$ are #). Consider an accepting computation of $M$ on this input. Now consider symbol $a_p$ at position $p$. In the computation, $a_p$ will be visited many times. Let $t_1$ be the first time $M$ visits $a_p$. In general, $M$ can *sit* (i.e., stay) on $a_p$ for a while until some time $t_2$ when it moves left or right of $a_p$. In fact, we can assume without loss of generality (by adding states, if necessary) that every time a cell is visited, $M$ sits on the cell at least one step before moving; thus $t_2 > t_1$. $M$ may later revisit $a_p$ at time $t_3$, sit on it, and then move left or right of $a_p$ at time $t_4$. Thus, in the accepting computation, we can associate with $a_p$ a time sequence $(t_1, ..., t_m)$, where for each $i \geqslant 1$, $t_{2i-1}$ is the $i$-th time $M$ visits $a_p$, and $t_{2i}$ is the $i$-th time it leaves $a_p$. $M$ sits on $a_p$ during the time period from $t_{2i-1}$ to $t_{2i}$. Note that even though $M$ is finite-crossing, this time period when $M$ sits on $a_p$ can be unbounded. Clearly, $m$ is no more than some fixed number since $M$ is finite-crossing. Corresponding to the time sequence $(t_1, ..., t_m)$ associated with symbol $a_p$, we define a *crossing vector* $R = (I_1, ..., I_m)$, where for odd $i$, $I_i = (d_1, q_1, r_1, q_2, r_2, d_2, r_3)$, where

(1) $d_1$ is the direction from which the head entered symbol $a_p$ at time $t_i$;
(2) $q_1$ is the state when it entered $a_p$;
(3) $r_1$ is the instruction that was used in the move above;
(4) $q_2$ is the state at time $t_{i+1}$;
(5) $r_2$ is the instruction that was used at time $t_{i+1} - 1$;
(6) $d_2$ is the direction from which it left symbol $a_p$ at time $t_{i+1}$; and
(7) $r_3$ is the instruction used when it left $a_p$.

We construct a nondeterministic one-way machine $M'$ which simulates the accepting computation of $M$ by nondeterministically guessing the sequence of crossing vectors $R_1, ..., R_n$ as it processes the input 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 $M$, machine $M'$ uses two counters $C_1$ and $C_2$. $C_1$ is used to record the increases in $C$, while $C_2$ 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 $M'$. ∎

Since the emptiness problem for semilinear sets is decidable (it is empty if there are no generators), we have:

**Theorem 3.4** *The emptiness problem is decidable for nondeterministic finite-crossing reversal-bounded multicounter machines.*

Theorem 3.2 can be generalized to allow one of the counters to be unrestricted as shown in [Iba78]:

**Theorem 3.5** *Let $M$ be a nondeterministic one-way machine with one unrestricted counter and several reversal-bounded counters. Then $f(L(M))$ is a semilinear set effectively computable from $M$. Thus, the emptiness problem for these machines is decidable.*

We now turn to other decision problems.

**Theorem 3.6** *The infiniteness problem is decidable for the class of nondeterministic finite-crossing reversal-bounded multicounter machines as well as for the class of nondeterministic one-way machines with one unrestricted counter and several reversal-bounded counters. The disjointness problem is also decidable for machines in the first class.*

**Proof:** The first part follows from Theorems 3.4 and 3.5 and the fact that it decidable to determine whether a semilinear set is infinite. It is also clear that for the model that has a finite-crossing input with no unrestricted counter, the disjointness problem is decidable (since given two such machines, one can construct another machine of the same type that simulates them in parallel). ∎

Containment and equivalence are undecidable for nondeterministic machines. In fact, it is undecidable to determine, given a nondeterministic one-way machine with one 1-reversal counter, whether it accepts all strings [BB74]. It is easy to show that the class of languages accepted by deterministic finite-crossing reversal-bounded multicounter machines is effectively closed under union, intersection, and complementation. Hence from Theorem 3.4:

**Theorem 3.7** *The containment and equivalence problems are decidable for deterministic finite-crossing reversal-bounded multicounter machines.*

## 4 Generalizations

Now we study various generalizations of reversal-bounded multicounter machines. For the proofs in this section, it is convenient to represent a machine

| (a) Program structure | (b) Instruction set |
|---|---|
| **begin** | $s$ : **read** ($INPUT$) |
| **input**(#$w$#) | $s$ : **if** $INPUT = a$ **then goto** $p$ **else goto** $q$ |
| $P$ | $s$ : **left** |
| **end** | $s$ : **right** |
| | $s : x := x + 1$ |
| | $s : x := x - 1$ |
| | $s$ : **if** $x = 0$ **then goto** $p$ **else goto** $q$ |
| | $s$ : **accept** |
| | $s$ : **reject** |
| | $s$ : **goto** $p$ |

Fig. 1. General program structure and instruction set

as a program. The specification "refines" (splits) a move into several atomic steps. The standard model of a deterministic multicounter machine can be specified by a program $M$ of the form shown in Figure 1(a). Here $P$ is a sequence of labeled instructions, where each instruction is of the form shown in Figure 1(b), where

(1) $s, p, q$ denote labels or states (we will use the latter terminology in the paper);
(2) **read** ($INPUT$) means read the symbol currently under the input head and store it in $INPUT$;
(3) $a$ is #, $, or a symbol in the input alphabet of the machine;
(4) The instruction **left** means move the input head one cell to the left, and **right** means move the input head one cell to the right;
(5) $x$ represents a counter. Thus a machine with $k$ counters will have $k$ such $x$'s.

The machine starts its computation with the first instruction in $P$ with the input head on the left delimiter and all the counters set to zero. As before, an input #$w$# is accepted (rejected) if $M$ on this input halts in ***accept*** (***reject***).

We can make the machine nondeterministic by allowing a nondeterministic instruction of the form:

$s$ : **goto** $p$ **or goto** $q$

Clearly this is the only nondeterministic instruction we need. Other forms of nondeterminism (e.g., allowing nondeterministic assignments like "$x := x + 1$ **or** $y := y - 1$" or allowing instructions like "**left or right**" do not add any more power to the machine. Hence, we may assume (without loss of generality) that a program for a nondeterministic multicounter machine has only one type of nondeterministic instruction, and it is of the form "$s$ : **goto** $p$ **or goto** $q$". All other instructions in the program are deterministic.

The notions of finite-crossing and reversal-bounded are as before. However, there is a stronger notion of reversal-boundedness. A counter with the following computation pattern:

000001111111222222344444 corresponds to 0-reversal.

In this example there are segments of the computation when the counter value does not change. We define a stronger notion of reversal-boundedness. We say that $M$ is *strongly reversal-bounded* if there is a nonnegative integer $r$ such that for any computation on any input, every counter of $M$ makes no more than $r$ alternations between increasing, no-change, and decreasing modes. In the above example, the pattern corresponds to 6 strong reversals.

Obviously a strongly reversal-bounded multicounter machine is also reversal-bounded. However, a reversal-bounded machine need not be strongly reversal-bounded. For example, the patterns of the form "$122334455\cdots$" correspond to 0-reversal, but are not strongly reversal-bounded.

### *4.1 Constant increments and comparisons*

The first generalization of a multicounter machine is to allow the counters to store negative numbers, and allow the program to use assignments of the form:

$s$ : $x := x + c$,

and conditionals (if statements) of the form:

$s$ : **if** $x\theta c$ **then goto** $p$ **else goto** $q$

where $c$ is an integer constant (there are a finite number of such constants in the program), and $\theta$ is one of $<, >, =$.

One can easily show that any multicounter machine $M$ that uses the generalized instructions above can be converted to an equivalent standard model

$M'$ such that $L(M) = L(M')$. Moreover, $M'$ is (strongly) reversal-bounded if and only if $M$ is (strongly) reversal-bounded. 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. Thus, we have

**Theorem 4.1** *The emptiness, infiniteness, disjointness problems are decidable for nondeterministic finite-crossing reversal bounded multicounter machines with constant increments and comparisons. Furthermore, containment and equivalence problems are decidable for deterministic versions of such machines.*

In view of Theorem 4.1, we will now assume (unless otherwise specified) that a "standard" machine model can use the generalized instructions above.

### *4.2 Linear conditions*

We can further allow conditionals (tests) like

$s:$ **if** $5x - 3y + 2z < 7$ **then goto** $p$ **else goto** $q$

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 $a_v$ 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 multicounter machine $M$ to use conditionals of the form:

$s:$ **if** $L$ **then goto** $p$ **else goto** $q$

where $L$ is a linear relation on the counters. Note that any nondeterministic multicounter machine $M$ that uses linear relation conditionals can be converted to an equivalent machine $M'$ that uses only atomic linear-relation conditionals. Moreover, $M'$ is (strongly) reversal-bounded iff $M$ is (strongly) reversal-bounded. We consider two cases: reversal bounded and strongly reversal bounded cases.

*4.2.1 Reversal-bounded case*

The halting (and, hence, the emptiness) problem is undecidable for reversal-bounded multicounter machines that allow conditionals of the form: "$s$ : **if** $x = y$ **then goto** $p$ **else goto** $q$". (Note that this can be simulated by conditionals of the form "$s$ : **if** $x - y < 0$ **then goto** $p$ **else goto** $q$".) In fact, the undecidability holds for 0-reversal machines. This follows from Minsky's result [Min61] that the halting problem is undecidable for machines with two unrestricted counters. Suppose $M$ is a two-counter machine. We construct a machine $M'$ with four counters. For each counter $x$ of $M$, $M'$ uses two counters $x_+$ and $x_-$. An instruction $x := x+1$ in $M$ becomes an instruction $x_+ := x_+ + 1$ in $M'$; an instruction $x := x - 1$ in $M$ becomes an instruction $x_- := x_- + 1$ in $M'$. The conditional "**if** $x = 0$ **then** $\cdots$" in $M$ becomes "**if** $x_+ = x_-$ **then** $\cdots$" in $M'$. Clearly, $M'$ simulates $M$, and $M'$ halts if and only if $M$ halts. Moreover, each counter in $M'$ is 0-reversal.

In fact, the undecidability for halting holds even when there are only three 0-reversal counters:

**Theorem 4.2** *Consider only deterministic machines with 3 counters, $C_1, C_2$, and $T$, with no input tape. The counters which are initially 0 can only use instructions of the form $x := x + 1$ (where $x$ is a counter), and linear test $T = C_1$? or $T = C_2$? (Note that $C_1 = C_2$? is not allowed). The halting problem for such machines is undecidable.*

**Proof:** A close look at the proof of the undecidability of the halting problem for two-counter machines (with no input tape) in [Min61] reveals that the counters behave in a regular pattern. The two counter machine operates in phases in the following way. Let $C_1$ and $C_2$ be its counters. Then $M$'s operation can be divided into phases $P_1, P_2, P_3, ...$, where each $P_i$ starts with one of the counters equal to zero and the other counter equal to some positive integer $d_i$. During the phase, the first counter is increasing, while the second counter is decreasing. The phase ends with the first counter having value $d_{i+1}$ and the second counter having value 0. Then in the next phase the modes of the counters are interchanged. Thus, a sequence of configurations corresponding to the phases above will be of the form:

$$(q_1, 0, d_1), (q_2, d_2, 0), (q_3, 0, d_3), (q_4, d_4, 0), ...$$

where the $q_i$ are states and $d_1 = 1, d_2, d_3, ...$ are positive integers. Note that the second component of the configuration refers to the value of $C_1$, while the third component refers to the value of $C_2$.

We construct a 3-counter machine $M'$ with counters $C_1', C_2'$ and $T$ which simulates $M$. The sequence of configurations of $M'$ corresponding to the above

phases would have the form (the second, third, and fourth components correspond to the values of $C_1', C_2'$, and $T$, respectively):

$(q_1, 0, d_1, 0)$,

$(q_2, d_1 + d_2, d_1, d_1)$,

$(q_3, d_1 + d_2, d_1 + d_2 + d_3, d_1 + d_2)$,

$(q_4, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3, d_1 + d_2 + d_3)$,

$(q_5, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3 + d_4 + d_5, d_1 + d_2 + d_3 + d_4)$,

$(q_6, d_1 + d_2 + d_3 + d_4 + d_5 + d_6, d_1 + d_2 + d_3 + d_4 + d_5, d_1 + d_2 + d_3 + d_4 + d_5)$,

...

To go from, for example, $(q_1, 0, d_1, 0)$ to $(q_2, d_1{+}d_2, d_1, d_1)$, $C_1'$ and $T$ are incremented until $T{=}C_2'$. During the phase, $C_1'$ also simulates $C_1$, adding $d_2$ to the counter. Thus $C_1'$ will have value $d_1{+}d_2$ at the end of the phase. ∎

The 3 counters in the result above are necessary in view of the following theorem.

**Theorem 4.3** *The emptiness problem is decidable for one-way nondeterministic machines with two reversal-bounded counters, where in addition to standard instructions, the machines can use tests of the form: $x\theta c$ and $x - y\theta c$ where $x, y$ represent the two counters, $c$ represents a constant, and $\theta$ is $>, <$, or $=$.*

**Proof:** Given a reversal-bounded two-counter machine $M$, we construct a machine $M'$ with two reversal-bounded counters and one unrestricted counter that uses only standard instructions, with the help of Theorem 3.5. ∎

### *4.2.2 Strongly reversal-bounded case*

Note that while the machine $M'$ in the construction in Theorem 4.2 is reversal-bounded, it is not strongly reversal-bounded. However, we can prove the following:

**Theorem 4.4** *The emptiness problem is decidable for nondeterministic finite-crossing strongly reversal-bounded multicounter machines using linear-relation conditionals on the counters.*

Before we give the proof we need some definitions and notations.

Suppose $M$ is a nondeterministic finite-crossing strongly reversal-bounded multicounter machine. During a computation, each counter of $M$ can be in any of the following three modes: *increasing, no-change, decreasing.* A counter makes a mode-change if it goes from mode $X$ to mode $Y$, with $Y$ different from $X$. Thus, e.g., a counter can go from no-change to increasing, or from increasing to decreasing, etc. We note that since the machine executes its program sequentially (one instruction at a time), no two counters can make a mode-change at the same time. Assume there are $k$ counters. At any time during the computation, the modes of the counters can be represented by a mode-vector $Q = \langle m_1, ..., m_k \rangle$, where $m_i$ is the mode of the $i$-th counter, for $1 \leqslant i \leqslant k$. There are only a finite number ($3^k$) of such vectors. The behavior of the counters during an accepting computation (which, by definition, is a halting computation) can be represented by a sequence: $Q_1 N_1 Q_2 N_2 \cdots Q_t N_t$ where:

(1) The $Q_i$'s are mode-vectors,
(2) Each $N_i$ represents the (possibly empty) period when no counter changes mode,
(3) For each $1 \leqslant i \leqslant k-1$, $Q_{i+1}$ differs from $Q_i$ in exactly one component (i.e., exactly one counter changes mode), and
(4) The starting mode-vector $Q_1 = \langle \text{no-change}, ..., \text{no-change} \rangle$.

Thus, we can divide the computation into phases, where in each phase, no counter changes mode. Now since the machine is strongly reversal-bounded, $t$ is upper-bounded by some fixed number.

Call the sequence $\langle Q_1, ..., Q_t \rangle$ a $Q$-vector. (Note that since each $Q_i$ is a k-tuple, the $Q$-vector has $k \times t$ components.) Since $t$ is upper-bounded by some fixed number, there are only a finite number of such $Q$-vectors.

We now prove Theorem 4.4. Let $M$ be a nondeterministic finite-crossing strongly reversal-bounded multicounter machine that uses atomic linear-relation predicates. We describe the construction of an equivalent nondeterministic finite-crossing strongly reversal-bounded multicounter machine $M'$ (which may have more counters than $M$) that uses only the standard instructions.

The construction of $M'$ is an induction on the number of atomic linear relations occurring in the program of $M$. Consider a specific instruction, say labeled $s$ (i.e. state $s$) of the form:

$s$ : **if** $L$ **then goto** $p$ **else goto** $q$

in the program of $M$, where $L$ is an atomic linear relation. We will construct an equivalent strongly reversal-bounded machine $M'$ without this instruction (i.e., $M'$ has one less atomic linear relation). Note that $M'$ cannot simply

implement this conditional using the standard instructions since the conditional will require a finite number of reversals on the counters of $M'$. If this conditional is executed by $M$ an unbounded number of times during the computation, the counters of $M'$ will not be reversal-bounded.

The basic idea in the construction of $M'$ is as follows:

(1) $M'$ stores in its states the atomic linear relation $L$.
(2) $M'$ first guesses and stores in its states a $Q$-vector $\langle Q_1, ..., Q_t \rangle$.
(3) $M'$ simulates $M$ by phases, where each phase starts with mode-vector $Q_i$ and ends with mode-vector $Q_{i+1}$. We assume that $M$ keeps track of the values of the counters of $M$ during the computation and, in particular, has available in its counters the values of the counters of $M$ at the start and end of each phase. We also assume that $M'$ keeps track of the state changes of $M$. In the simulation, $M'$ does not use instruction

$$s: \textbf{ if } L \textbf{ then goto } p \textbf{ else goto } q$$

We give the details of simulating a phase starting at $Q_i$ and ending at $Q_{i+1}$:

(1) $M'$ first checks, using the values of the counters involved in the conditional

$$s: \textbf{ if } L \textbf{ then goto } p \textbf{ else goto } q$$

whether $L$ is true or whether it is false at the beginning of the phase.
(2) Consider the case when $L$ is true (the case when $L$ is false is symmetric). There are two subcases:
- *Subcase 1:* Throughout the phase, $L$ remains true. Since $L$ is an atomic linear relation, it is convex. It follows that $L$ is true throughout the phase if and only if it is true at the start and at the end of the phase.
- *Subcase 2:* During the computation, $L$ became false. Again since $L$ is convex, when it turns false it will remain false until the end of the phase. Moreover, the time when $L$ becomes false is unique (i.e., it only occurs once in the entire phase).

So, to simulate a phase, $M'$ guesses one of the two subcases above. Suppose $M'$ guesses Subcase 1. Then it simulates $M'$ faithfully using the instruction "**goto** $p$" in place of "$s:$ **if** $L$ **then goto** $p$ **else goto** $q$" until the end of the phase. At the end of the phase it verifies that $L$ is still true.

Suppose $M'$ guesses Subcase 2. Then it simulates $M'$ faithfully. But, in addition, $M'$ guesses the last time, $u$, the conditional instruction will be executed by $M$ with value true (meaning the conditional instruction becomes false at the $(u+1)$-st time it is executed by $M$).

Up to time $u$, $M'$ uses the instruction "**goto** $p$".

After time $u$, $M'$ uses the instruction "**goto** $q$".

$M'$ also verifies that at time $u$, $L$ is indeed true, and it is false at time $u+1$.

It follows from the description above that we can remove the instruction

$s$ : **if** $L$ **then goto** $p$ **else goto** $q$

and $M'$ is still strongly-reversal bounded. We can iterate the process to remove all atomic linear-relation conditionals.

### *4.3 Allowing parameterized constants*

We can further generalize our model by allowing parameterized constants in the linear relations. So for example, we can allow instructions like

$s$ : **if** $3x - 5y - 2D_1 + 9D_2 < 12$ **then goto** $p$ **else goto** $q$

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 at the start of the computation by including them on the input tape. Thus, the input to the machine with $k$ parameterized constants will have 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.

**Theorem 4.5** *The emptiness problem is decidable for nondeterministic finite-crossing strongly reversal-bounded multicounter machines using linear-relation conditionals on the counters and parameterized constants.*

**Proof:** ¿From the construction in the proof of Theorem 4.4, we see that when the parameterized constants are included in the linear relation, $M'$ will only need to access these constants finitely many times. Thus, when there are parameterized constants, $M'$ first reads the input and stores $d_1, ..., d_k$ in some counters, and the construction of $M'$ proceeds as before. ∎

### *4.4 Allowing one unrestricted counter*

We can allow one of the counters to be unrestricted (i.e., not reversal-bounded) provided the input is one-way. Like Theorem 4.5, we can prove:

**Theorem 4.6** *The emptiness problem is decidable for nondeterministic one-way machines with one unrestricted counter and several strongly reversal-bounded counters using linear-relation conditionals on the reversal-bounded counters and parameterized constants.*

### *4.5 Restricted linear relations*

Because of Theorem 4.2, none of Theorems 4.4–4.6 holds when the machines are reversal-bounded but not strongly reversal-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 fine, 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 one check that the results of Theorems 4.4–4.6 hold for reversal-bounded machines (which are not necessarily strongly reversal-bounded):

**Theorem 4.7** *The emptiness problem is decidable for:*

*(1) Nondeterministic finite-crossing reversal-bounded multicounter machines with restricted linear-relation conditionals on the counters and parameterized constants.*
*(2) Nondeterministic one-way machines with one unrestricted counter and several reversal-bounded counters with restricted linear relation conditionals on the reversal-bounded counters and parameterized constants.*

### *4.6 Generalizing the assignment statement*

So far we have only considered conditionals. Suppose we allow assignments of the form: "$s : x := y$", where $x, y$ are counters. Then we can prove the following.

**Theorem 4.8** *The emptiness problem is decidable for nondeterministic one-way machines with one unrestricted counter and several strongly reversal-bounded counters with linear-relation conditionals on the reversal-bounded counters and parameterized constants, provided the only assignments used are of the form: $x := x + 1$, $x := x - 1$, $x := 0$, and $x := y$.*

The requirement that the machine can only use assignments of the form: $x := x + 1$, $x := x - 1$, $x := 0$ in addition to instructions of the form $x := y$ is

necessary, since we have:

**Theorem 4.9** *The emptiness problem is undecidable for nondeterministic one-way machines with strongly reversal-bounded counters which use only conditionals of the form "$s$ : **if** $x = y$ **then goto** $p$ **else goto** $q$" and only assignment statements of the form: $x := x + a$, even if there are only two distinct constants $a$ used in the program.*

**Proof:** The proof uses a reduction from two-counter machines $M$ that is similar to the one described in the proof of Theorem 4.2, i.e., each counter $x$ is simulated with two counters: $x_+$ tracking all additions and $x_-$ tracking all subtractions.

In particular, let $a, b$ be two distinct constants used in the assignment statements. Without loss of generality, let $a < b$. Now, for each transition, if a counter $x$ does not change in $M$, we add $a$ to both $x_+$ and $x_-$. If $x$ gets an increment (or decrement), we add $b$ to $x_+$ (or, respectively, $x_-$). Note that since $b > a$, the difference $b - a > 0$ is used to record the addition "+1" or the subtraction "−1". Undecidability follows immediately. ∎

We have looked at various generalizations of reversal-bounded multicounter machines in this section. Although Theorems 4.3 - 4.8 show only the decidability of the emptiness problem for these generalizations, all the proofs involve converting the machine being considered to an equivalent nondeterministic finite-crossing reversal-bounded multicounter machine or to an equivalent nondeterministic one-way machine with one unrestricted counter and several reversal-bounded counters. It follows from Theorems 3.6 that the infiniteness problem is also decidable for these generalized models. It also follows from Theorem 3.6 that for the models that have a finite-crossing input, the disjointness problem is decidable. Finally, it is easy to show that the deterministic versions of the models with finite-crossing input are closed under complementation, so their containment and equivalence problems are also decidable.

## 5 Reachability and Safety

The results of the previous section can be used to analyze verification problems (such as reachability and safety) in infinite-state transition systems that can be modeled by multicounter machines. Decidability of reachability is of importance in the areas of model checking, verification, and testing [Esp97,CJ98,WB98]. In these areas, a machine is used as a system specification rather than a language recognizer, the interest being more in the behaviors that the machine generates. Thus, in this section, unless otherwise specified,

the machines have *no* input tape.

For notational convenience, we restrict our attention to machines whose counters can only store nonnegative integers. The results easily extend to the case when the counters can be negative.

Let $M$ be a nondeterministic reversal-bounded $k$-counter machine with state set $\{1, 2, ..., s\}$ for some $s$. Each counter can be incremented by integer constants $(+, -, 0)$ and can be tested if $<, >, =$ to integer constants. Let $(j, v_1, ..., v_k)$ denote the configuration of $M$ when it is in state $j$, and counter $i$ has value $v_i$ for $i = 1, 2, ..., k$. Thus, the set of all possible configurations is a subset of $\mathbb{N}^{k+1}$. We use the symbols $\alpha, \beta, ...$ to denote configurations.

Given $M$, let $R(M) = \{(\alpha, \beta) \mid \alpha$ can reach $\beta$ in 0 or more moves $\}$. $R(M)$, which is a subset of $\mathbb{N}^{2k+2}$, is called the *binary reachability* set of $M$. For a set $S$ of configurations, define $post^*_M(S)$ to be the set of all successors of configurations in $S$, i.e., $post^*_M(S) = \{\alpha \mid \alpha$ can be reached from some configuration in $S$ in 0 or more moves $\}$. Similarly, define $pre^*_M(S) = \{\alpha \mid \alpha$ can reach some configuration in $S$ in 0 or more moves $\}$. $post^*_M(S)$ and $pre^*_M(S)$ are called the *forward* and *backward reachability* of $M$ with respect to $S$, respectively.

Note that configuration $(j, v_1, ..., v_k)$ in $\mathbb{N}^{k+1}$ can be represented as a string $j\%v_1\% \cdots \%v_k$, where $j, v_1, ..., v_k$ are represented in unary (separated by %). Thus, $R(M)$, $post^*_M(S)$, and $pre^*_M(S)$ can be viewed as languages (e.g., regular, context-free, etc.)

When we say that a subset $S$ of $\mathbb{N}^n$ is accepted by a multicounter machine $M$, we mean that, $M$ when started in its initial state with its first $n$ counters ($M$ can have more than $n$ counters) set to an $n$-tuple accepts (i.e., enters an accepting state) if and only if the $n$-tuple is in $S$. Note that this is equivalent to equipping the machine with an input tape that contains the unary encoding of the $n$-tuple.

We will need the following characterizations that have been shown or follow from the results in [Iba78]:

**Theorem 5.1** *Let $S$ be a subset of $\mathbb{N}^n$. Then the following statements are (effectively) equivalent:*

*(1) $S$ is definable by a Presburger formula.*
*(2) $S$ can be accepted by a nondeterministic machine with one unrestricted counter and several reversal-bounded counters.*
*(3) $S$ can be accepted by a nondeterministic reversal-bounded multicounter machine.*
*(4) $S$ can be accepted by a deterministic reversal-bounded multicounter ma-*

*chine.*

**Theorem 5.2** *Let $M$ be a nondeterministic reversal-bounded $k$-counter machine and $S$ a subset of $\mathbb{N}^{k+1}$. Then*

*(1) $R(M)$ is definable by a Presburger formula.*
*(2) $S$ is definable by a Presburger formula if and only if $post^*_M(S)$ ($pre^*_M(S)$) is definable by a Presburger formula.*
*(3) If $S$ is Presburger, then $post^*_M(S)$ ($pre^*_M(S)$) can be accepted by a deterministic reversal-bounded multicounter machine. Similarly for $R(M)$.*
*(4) 1-3 still hold even if one of the counters is unrestricted.*

**Proof:** For Part 1, we construct a machine $M'$ that accepts $R(M)$. $M'$, when given $(\alpha\,\beta)$ in its counters, simulates the computation of $M$ starting in configuration $\alpha$. At some point, $M'$ guesses that it has reached configuration $\beta$, which it can can easily verify. The result follows from Theorem 5.1 (equivalence of 1 and 3).

For Part 2 we only prove the case of $pre^*_M(S)$. If $S$ is definable by a Presburger formula, then there is a machine $M_S$ accepting $S$, by Theorem 5.1 (equivalence of 1 and 3). We construct a machine $M_{pre}$ from $M$ and $M_S$. $M_{pre}$, when given $\alpha$ in its counters, simulates $M$ starting in this configuration. At some point, $M_{pre}$ guesses that it has reached a configuration $\beta$ in $S$, which it can verify by using $M_S$. Hence $pre^*_M(S)$ is Presburger.

Conversely, suppose $pre^*_M(S)$ is Presburger and accepted by a machine $M_{pre}$. We construct a machine $M_S$ accepting $S$. $M_S$, when given $\beta$ in its counters, "guesses" and stores a configuration $\alpha$ in its counters. $M_S$ than checks that $\alpha$ is accepted by $M_{pre}$ and that $\beta$ is reachable from $\alpha$.

Part 3 follows from parts 1-2 and Theorem 5.1 (equivalence of 1 and 4).

Part 4 follows from parts 1-3 and Theorem 5.1 (equivalence of 1 and 2). ∎

Next, we consider strongly reversal-bounded multicounter machines with linear-relation conditionals on the counters and parameterized constants. For these machines, the configuration is now a tuple $(j, v_1, ..., v_k, d_1, ..., d_m)$, where $d_1, ..., d_m$ represent the values of the parameterized constants. Then the following theorem follows from the results of the previous section:

**Theorem 5.3** *The statements in Theorem 5.2 hold for:*

*(1) $M$ a nondeterministic strongly reversal-bounded $k$-counter machine using linear relation conditions on the counters and parameterized constants.*
*(2) $M$ a nondeterministic reversal-bounded $k$-counter machine using*

*restricted linear relation conditions on the counters and parameterized constants.*

*The results are valid even if one of the counters is unrestricted as long as this counter is not involved in the linear relation conditionals.*

Theorem 5.3 (2) is not true when the machine has $k > 1$ reversal-bounded counters and one unrestricted counter, as shown in the next result:

**Theorem 5.4** *Consider deterministic machines with a single unrestricted counter $U$, $k$ reversal-bounded counters, and a finite number of parameterized constants. In addition to the standard instructions, the unrestricted counter can be tested for "$U = D$?" where $D$ represents a parameterized constant. Then*

*(1) The emptiness problem (i.e., deciding given a machine $M$ whether there exists an assignment of values to the parameterized constants which will cause $M$ to accept) is undecidable, even when restricted to $k = 2$.*
*(2) The emptiness problem is decidable when $k = 1$.*
*(3) The emptiness problem is decidable for any $k$, provided there is only* one *parameterized constant.*

**Proof:** The proof of part 1 uses the undecidability of Hilbert's Tenth Problem (HTP) [Mat70], which is to decide for a given polynomial $P(x_1, ..., x_n)$ with integer coefficients whether it has a nonnegative integral root.

First consider a term $st(x_1, ..., x_n)$=$sx_1^{i_1}...x_n^{i_n}$ of the polynomial $P(x_1, ..., x_n)$, where s = + or −, $i_1, ..., i_n \geqslant 0$. We show how to construct a deterministic machine $M_t$ with one unrestricted counter $U$, two reversal-bounded counters $C_1, C_2$, and parameterized constants $A_1, ..., A_n, B$ such that $M_t$ with the constants assigned nonnegative integer values $\alpha_1, ..., \alpha_n, \beta$ accepts if and only if $\beta = \alpha_1^{i_1}...\alpha_n^{i_n}$.

In what follows, when we say that $M_t$ "adds" a parameterized constant to $C_1$, we mean that $M_t$ resets $U$ to zero and then adds 1's to both $U$ and $C_1$ until $U$ is equal to the parameterized constant. $M_t$ "sets" $C_1$ to a parameterized constant means $M_t$ first resets $C_1$ to zero (if it is not already zero) and then adds the parameterized constant to $C_1$.

The exponents $i_1, ..., i_n$ are stored in the states of $M_t$. Assume that each $i_j \geqslant 1$. (Otherwise, ignore the exponent.) Initially, $U, C_1, C_2$ are zero.

$M_t$ sets the $C_1$ to $\alpha_1$. Then $M_t$ computes $\alpha_1^2$ in $C_2$ by iterating the following process until $C_1$ becomes zero: (1) Add $\alpha_1$ to $C_2$. (2) Decrement $C_1$ by one.

By iterating the procedure above and alternately switching the roles of $C_1$ and

$C_2$, $M_t$ can compute $\alpha_1^{i_1} \cdots \alpha_n^{i_n}$ in one of the counters, say $C_2$. $M_t$ then sets $C_1$ to $\beta$ and checks that $C_1$ is equal to $C_2$. Note that the counters $C_1$ and $C_2$ are reversal-bounded.

Now let $P(x_1, ..., x_n)$=$s_1t_1$+...+$s_rt_r$, where each term $s_jt_j = s_jx_1^{i_{j1}}...x_n^{i_{jn}}$. We construct a deterministic machine $M_P$ with one unrestricted counter, two reversal-bounded counters, and parameterized constants $A_1, ..., A_n, B_1, ..., B_r$ such that $M_P$ with the constants assigned nonnegative integer values $\alpha_1, ..., \alpha_n$, $\beta_1, ..., \beta_r$ accepts if and only if $\beta_j = \alpha_1^{i_{j1}} \cdots \alpha_n^{i_{jn}}$ and $s_1\beta_1 + \cdots + s_r\beta_r = 0$. The integers $i_{11}, ..., i_{1n}, ..., i_{r1}, ..., i_{rn}$ and signs $s_1, \ldots, s_r$ are stored in the states of $M_P$. $M_P$ uses the technique described above to check that $\beta_j = \alpha_1^{i_{j1}} \cdots \alpha_n^{i_{jn}}$ and then verifies that $s_1\beta_1 + \cdots + s_r\beta_r = 0$.

For part 2, let $M$ be a deterministic machine with one unrestricted counter $U$ and one reversal-bounded counter $C$ and parameterized constants $A_1, ..., A_n$. Assume without loss of generality that the values $\alpha_1, ..., \alpha_n$ the parameterized constants can assume in any computation is such that $1 \leqslant \alpha_1 < \cdots < \alpha_n$. (Note that the domain of values can be partitioned into a finite number of orderings.)

We convert $M$ to a different type of machine $M'$. $M'$ has an unrestricted two-way input tape (with delimiters) and one reversal-bounded counter $C$ but no parameterized constants, such that $M'$ accepts empty if and only if $M$ accepts empty. The result follows since the emptiness problem is decidable for deterministic two-way machines with one reversal-bounded counter [IJTW95]. $M'$ has input alphabet {1,%} (the delimiter is #). $M'$ rejects all inputs not of the form $\#1^{i_1}\%1^{i_2}\%\cdots 1^{i_n}\%1^k\#$.

Corresponding to values $\alpha_1, ..., \alpha_n$ assigned to the parameterized constants, $M'$ is given input $w = \#1^{i_1}\%1^{i_2}\%\cdots 1^{i_n}\%1^k\#$, where

$$
\begin{aligned}
i_1 &= \alpha_1 - 1, \\
i_2 &= \alpha_2 - \alpha_1 - 1, \\
&\cdots \\
i_n &= \alpha_n - \alpha_{n-1} - \cdots - \alpha_1 - 1
\end{aligned}
$$

and $k$ is a nonnegative integer. Note that there are exactly $n$ occurrences of the symbol % in $w$. $M'$ simulates $M$ faithfully, with the input head simulating the unrestricted counter $U$. Zero of the counter corresponds to the left delimiter, $+1$ corresponds to moving right one cell and $-1$ corresponds to moving left one cell. The input head on the $i$-th % corresponds to $U$ being equal to parameterized constant $\alpha_i$. The suffix $1^k$ on the input (for some $k$) is a "*padding*" used to simulate $U$ when it's value is greater than $\alpha_n$. It follows that $M'$ accepts $w$

(for some $k$) if and only if $M$ accepts when the parameterized constants are assigned values $\alpha_1, ..., \alpha_n$. We now show that the padding $1^k$ can be removed.

If we can show that $k \leqslant c \cdot \alpha_n$ for some positive integer $c$ (note that the length of the input to $M'$ is $\alpha_n$), then $M'$ can use the input to "simulate" the action of $U$ when the counter has value greater than $\alpha_n$. ($M'$ need only make at most $c$ right-to-left and left-to-right sweeps of the input.)

Let $s$ be the number of states of $M$. Consider the situation when counter $U$ of $M$ has just exceeded the value $\alpha_n$ (i.e., it has value $\alpha_n + 1$). Let the value of the reversal-bounded counter $C$ at that time be $v$. Suppose that $U$ is in increasing mode. Clearly, if $C$ is nondecreasing, $U$ cannot increase its count beyond $\alpha_n$ by more than $s$; otherwise, $M$ will be in an infinite loop. The only way that $U$ can increase its count beyond $\alpha_n$ by more than $s$ without going into an infinite loop is for $C$ to be decreasing, eventually becoming zero, i.e., while $C$ is decreasing, $U$ is increasing. Thus the maximum value of $U$ would be no more than $\alpha_n + s \cdot v$. We now derive an upper bound on the maximum value of $v$ during the entire computation of $M$. Initially $v$ is zero. Suppose we want to maximize the value of $v$ when it's in an increasing mode without the machine going into an infinite loop. Clearly, without counter $U$ exceeding value $\alpha_n + s$, $M$ can make no more than $s \cdot (\alpha_n + s)$ moves. Thus, $v$ can have at most value $s \cdot (\alpha_n + s)$ without $C$ reversing. $C$ can then reverse, i.e., decrease its value to zero while increasing $U$. When $C$ becomes zero, $U$ would have value at most $s \cdot (\alpha_n + s) + s \cdot (s \cdot (\alpha_n + s))$. Since $M$ is reversal-bounded, the biggest number $v$ that can be stored in $C$ without the going into an infinite loop can be obtained by $M'$ alternately decreasing $U$ and incrementing $C$ until $C$ becomes zero, and vice-versa. It follows that the maximum value of $v$ is $c \cdot \alpha_n$ for some integer $c$.

For part 3, let $M$ be a deterministic machine with one unrestricted counter $U$, one reversal-bounded counter $C$, and one parameterized constant $A$. As in part 2, we construct a machine $M'$ which, when given a two-way input tape $\#1^\alpha\#$, accepts if and only if $M$ with parameterized constant $A$ set to $\alpha$ accepts. The result follows, since the emptiness problem is decidable for deterministic two-way reversal-bounded multicounter machines over unary input [GI79]. ∎

**Remark.** Obviously, part 2 of the above theorem holds even if we allow tests of the form: $U\theta D$, where $D$ is a parameterized constant, and $\theta$ is $=, <$ or $>$.

The problem of *safety* is of importance in the area of verification. The following theorem follows from Theorems 5.1–5.3.

**Theorem 5.5** *It is decidable to determine for a given nondeterministic reversal-bounded multicounter machine $M$ and two given sets of configura-*

*tions $S$ and $T$ definable by Presburger formulas, whether every configuration in $S$ can only reach configurations in $T$. Thus, safety is decidable.*

## 6 Conclusions

We have introduced several generalizations of reversal-bounded multicounter machines and investigated their decision problems. We then used the decidable properties to analyze verification problems such as (binary, forward, backward) reachability and safety. We give an example analysis of an infinite-state transition system.

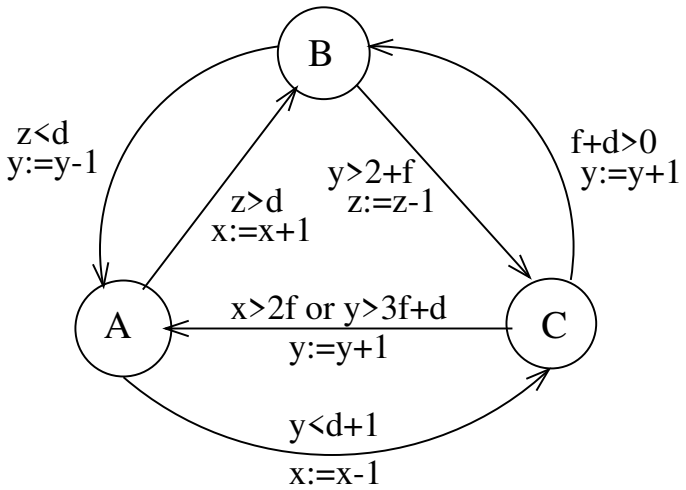


Fig. 2. An infinite-state transition system

In practice, many infinite-state transition systems can be modeled by multicounter machines, like the transition graph shown in Figure 2. Here, $M$ has counters $x, y, z$ and parameterized constants $d, f$. Assuming we are interested in the following safety property

For all $d$ and $f$, for all configurations $\alpha$ and $\beta$ such that $\alpha$ can reach $\beta$, if $d > f$ then counter $x$ in $\beta$ is greater than the sum of counters $y$ and $z$ in $\alpha$.

The negation of the property can be written as $\exists d, f, \alpha, \beta((\alpha, \beta) \in R(M) \wedge \neg(d > f \rightarrow \beta_x > \alpha_y + \alpha_z))$. In order to debug the property, $M$ can be effectively made reversal-bounded as $M'$ by giving a bound for reversals. Since $R(M') \subseteq R(M)$ is a lower approximation of $R(M)$, satisfiability of $(\alpha, \beta) \in R(M') \wedge \neg(d > f \rightarrow \beta_x > \alpha_y + \alpha_z)$ falsifies the property. From Theorem 5.2, we know $R(M')$ is Presburger. Thus, the above satisfiability checking is decidable. This is a new approach for analyzing safety properties for systems where the general reachability problem is known to be undecidable. Other approaches are to use semi-decision algorithms which are not guaranteed to

terminate [WB98] or to look at restricted classes of systems where reachability is decidable [CJ98].

It may seem that strong reversal-boundedness restricts the behavior of a counter too much, since changing from a strictly increasing (or decreasing) mode to a no-change mode counts as a reversal. However, if the counters behave like clocks which either increase with rate 1 or reset to 0 as in timed automata [AD94], strong reversal-boundedness is equivalent to reversal-boundedness. Using this observation and the results in this paper, we are able to show a number of results concerning the binary reachability of discrete timed pushdown automata [DIBKS00], past machines, and clocked systems with bounded resets and parameterized durations. For example, it follows from Theorem 5.3 that the binary reachability of discrete timed automata that use linear-relation tests (on clocks and parameterized constants) whose clocks are bounded-reset (i.e., each clock resets at most a fixed number of times) is Presburger, since these clocks can be viewed as strongly reversal-bounded counters. In fact, this result holds, even if one clock is not bounded-reset. When the clocks are not reset-bounded, it can be shown that binary reachability is not computable. In fact, "node reachability" is not decidable [AD94]. Finally, we note that although our results are for the discrete timed models, the techniques can be applied to the continuous timed versions. For example, a recent paper [Dan01] showed that safety analysis for timed pushdown automata with dense clocks can be reduced to that for timed pushdown automata with discrete clocks. Therefore, in characterizing the binary reachability of real-time systems with dense clocks, we need only look at the discrete time model.

## References


[AD94] R. Alur and D. Dill, A theory of timed automata, *Theoretical Computer Science*, 126(2):183-235, 1994.

[BB74] B. Baker and R. Book, Reversal-bounded multipushdown machines, *J. Comput. System Sci.*, 8:315-332, 1974.

[CJ98] H. Comon and Y. Jurski, Multiple counters automata, safety analysis and Presburger arithmetic, in:*Proc. 10th Int. Conf. on Computer Aided Verification (CAV 1998)*, Lecture Notes in Computer Science, Vol. 1855 (Springer, Berlin, 1998) 268–279.

[Dan01] Z. Dang, Binary reachability analysis of timed pushdown automata with dense clocks. Submitted, 2001.

[DIBKS00] Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su, Binary reachability analysis of discrete pushdown timed automata, in:*Proc. 12th Int.*

*Conf. on Computer Aided Verification (CAV 2000)*, Lecture Notes in Computer Science, Vol. 1427 (Springer, Berlin, 2000) 69–84.

[Esp97] J. Esparza, Decidability of Model Checking for Infinite-State Concurrent Systems, *Acta Informatica* 34(2): 85–107, 1997.

[GI79] E. M. Gurari and O. H. Ibarra, Simple counter machines and number-theoretic problems, *J. Comput. System Sci.*, 19:145–162, 1979.

[GS64] S. Ginsburg and E. Spanier. Bounded Algol-like languages. *Transactions of American Mathematical Society,* 113, 333–368, 1964.

[Iba78] O. H. Ibarra, Reversal-bounded multicounter machines and their decision problems, *J. ACM*, 25:116–133, 1978.

[IJTW95] O. H. Ibarra, T. Jiang, N. Tran, and H. Wang, New decidability results concerning two-way counter machines, *SIAM J. Comput.,* 24(1):123–137, 1995.

[Mat70] Y. Matijasevic, Enumerable sets are Diophantine, *Soviet Math. Dokl*, Vol. 11, 354–357, 1970.

[Min61] 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.

[Par66] R. Parikh, On context-free languages, *J. ACM*, 13:570–581, 1966.

[VP75] L. G. Valiant and M. S. Paterson, Deterministic one-counter automata, *J. Comput. System Sci.*, 10:340–350, 1975.

[WB98] P. Wolper and B. Boigelot, Verifying systems with infinite but regular state spaces, in: *Proc. 10th Int. Conf. on Computer Aided Verification (CAV 1998)*, Lecture Notes in Computer Science, Vol. 1427 (Springer, Berlin, 1998), 88–97.