Theoretical Computer Science 289 (2002) 165–189
Theoretical Computer Science
www.elsevier.com/locate/tcs

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

Received August 2000; received in revised form April 2001; accepted May 2001
Communicated by K. Culik

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.

Keywords: 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.) are

☆ A short version [16] of this paper appeared in the Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Lecture Notes in Computer Science 1893, Springer, Berlin, 2000, pp. 426–435.

* Corresponding author. Tel.: +1-805-893-41; fax: +1-805-893-85.
E-mail address: ibarra@cs.ucsb.edu (O. H. Ibarra).

[1] Supported in part by NSF grants IRI-9700370 and IIS-9817432.
[2] 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] Supported in part by NSF grant CCR-9970976 and NSF CAREER award CCR-9984822.

PII: S0304-3975(01)00268-7

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 [19] that the emptiness problem is undecidable for two-way counter machines even over an unary input alphabet. On binary inputs, if one restricts the input head of the counter machines to make only a finite number of turns (i.e., changes in direction) on the input tape, the emptiness problem is also undecidable, even for the case when the input head makes only one turn [14]. 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 [22].

In this paper, we study two-way finite automata augmented with several counters. A restricted version of these machines was studied in [14]. Since a counter can be incremented/decremented by 1 (and tested if it is 0), we count each alternation from nonincreasing 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., a test condition can be "$3x - 5y - 2D_1 + 9D_2 < 12$", 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.

Besides its own theoretical interests, the work presented in this paper is also motivated by the recent effort in verifying infinite-state systems. Inspired by the successes of efficient model-checking techniques for finite-state systems such as hardware devices and reactive systems [18], researchers are studying various models of infinite-state systems that are amenable to automatic verification. For this purpose, the systems studied include timed automata [2], pushdown automata [4], various versions of counter machines [6, 10], and various queue machines [1, 5, 21]. In particular, recently we have shown that discrete clocks in a timed automaton can be transformed into reversal-bounded counters [8]. A pattern technique is further provided to reduce dense clocks to discrete clocks [7]. These results build a bridge between counter machine theory and real-time verification. We believe the results in this paper can be further extended and used in verification/debugging pushdown/queue systems augmented with counters.

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 is the input alphabet and does not contain the symbol #. We can treat 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 nonzero) 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 nonnegative integers; decrementing a zero counter is undefined.)

(5) Enter state q.

If the machine is nondeterministic, there may be several choices for actions (3)–(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:

0000011111122222234444 has 0 reversals.

On the other hand,

000001111112222222344444333222123344 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 every pair of 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. Also, 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 the input 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) is equal to 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 the 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 $v_0, v_1, \ldots, v_n$ in $\mathbb{N}^k$ such that

$$S = \{v \mid v = v_0 + t_1 v_1 + \cdots + t_n v_n,\ \forall 1 \leqslant i \leqslant n,\ t_i \in \mathbb{N}\}.$$

The vectors v_0 (the constant vector) and $v_1, \ldots, v_n$ (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* [11].

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 $a_1, \ldots, a_k$. For each string (word) w in A^*, we define the *Parikh map* of w, denoted by $f(w)$, as follows:

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

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 [20].

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

The next result, which generalizes Theorem 3.1, was proved in [14].

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 machines 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 in any computation every counter becomes positive.

Let $a_1 \ldots a_n$ 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 may 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 $(t_1, \ldots, t_m)$, where for each $i \geqslant 1$, t_{2i-1} is the ith time M visits a_p, and t_{2i} is the ith 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, the 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 $(t_1, \ldots, t_m)$ associated with symbol a_p, we define a *crossing vector* $R = (I_1, \ldots, I_m)$, 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' that simulates the accepting computation of M by nondeterministically guessing the sequence of crossing vectors $R_1, \ldots, R_n$ 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 records 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 [14]:

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 is decidable to determine whether a semilinear set is infinite. It is also clear that since the model has a finite-crossing input with no unrestricted counter, the disjointness problem is also decidable (this is because 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 [3]. 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 multicounter machine as a program. The specification "refines" (splits) a move into several atomic steps and, in the meantime, employs parallel counter assignments. The standard model of a deterministic multicounter machine can be specified by a program M of the form shown in Fig. 1(a). Here P is a sequence of labeled instructions, where each instruction is of the form shown in Fig. 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, and
(5) X is a vector of k counters, where k is the number of counters in the machine and $\mathbf{v}$ is a vector over $\{-1, 0, 1\}$ called an *increment vector*. The instruction $X := X + \mathbf{v}$ performs *parallel* increments for all counters, where each counter can be incremented or decremented by 1 or stay unchanged.

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***).

begin
  input(#w#)
  P
end

(a) Program structure

s: **read** ($INPUT$)
s: **if** $INPUT = a$
  then goto p
  else goto q
s: **left**
s: **right**
s: $X := X + \mathbf{v}$
s: **if** $x = 0$
  then goto p
  else goto q
s: **accept**
s: **reject**
s: **goto** p

(b) Instruction set

Fig. 1. General program structure and instruction set.

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 notion of finite-crossing is the same as before and the notion of reversal-boundedness is extended in the following sense. During a computation, only parallel assignment instructions can change counter values. For every counter, its values before each parallel assignment instruction in the computation is executed can be recorded as a sequence, e.g.,

0000011111222222344444.

Each alternation between nonincreasing and nondecreasing in such a sequence for a counter is called a reversal for this counter.

More interestingly, there is a stronger version of the reversal-boundedness notion, which is introduced below. Consider the sequence above. Although it corresponds to 0-reversal, in this sequence there are segments of the computation when the counter value does not change (that is, the assignment instructions executed on each of these segments do not increase the value of this counter). Intuitively, the stronger notion of reversal-boundedness is to distinguish the modes of counter increasing, decreasing, and no-change. We say that a counter machine 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, on the sequence of assignments in the computation. In the above example, the pattern corresponds to 6 strong reversals, marked as follows (assuming the initial mode is stay):

$$00000\bar{1}\bar{1}1111\bar{2}\bar{2}2222\bar{3}4\bar{4}44.$$

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..." correspond to 0-reversal, but are not strongly reversal-bounded.

4.1. Constant comparisons

The first generalization of a multicounter machine is to allow the counters to store negative numbers, and allow the program to use conditionals (if statements) of the form

$$s: \textbf{ if } x\theta c \textbf{ then goto } p \textbf{ else goto } q$$

where c is an integer constant (there are a finite number of such constants in the program), and θ 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 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

$$s: \textbf{ if } 5x - 3y + 2z < 7 \textbf{ then goto } p \textbf{ else goto } q$$

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 negation ($\neg$) and conjunction ($\wedge$). Note that standard operations such as greater than ($>$), equality ($=$), logical implication ($\rightarrow$), and disjunction ($\vee$) 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.

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 [19] 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** ..." in M becomes "**if** $x_+ = x_-$ **then** ..." 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 [19] 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 $P_1, P_2, P_3, \ldots,$ 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

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

where the q_i are states and $d_1 = 1, d_2, d_3, \ldots$ 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)

$$\begin{aligned}
&(q_1, 0, d_1, 0),\\
&(q_2, d_1 + d_2, d_1, d_1),\\
&(q_3, d_1 + d_2, d_1 + d_2 + d_3, d_1 + d_2),\\
&(q_4, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3, d_1 + d_2 + d_3),\\
&(q_5, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3 + d_4 + d_5, d_1 + d_2 + d_3 + d_4),\\
&(q_6, d_1 + d_2 + d_3 + d_4 + d_5 + d_6, d_1 + d_2 + d_3 + d_4 + d_5,\\
&\quad d_1 + d_2 + d_3 + d_4 + d_5),\\
&\quad\quad \ldots .
\end{aligned}$$

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* θ *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. Since counter values can only be changed by parallel assignment instructions, there is no counter change between any two assignment instructions. During a computation, right before any parallel assignment instruction is executed, each counter of M can be in any of the following three modes: *increasing*, *no-change*, *decreasing*. These modes correspond to $x := x + 1$; $x := x + 0$; $x := x + (-1)$ in the parallel assignment instruction, respectively. 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. Notice that one parallel assignment instruction may cause mode-change for more than one counter. Assume there are k counters. At any time during the computation, the modes of the counters can be represented by a mode-vector $Q = \langle m_1, \ldots, m_k \rangle$, where m_i is the mode of the ith 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: $N_0 Q_1 N_1 Q_2 N_2 \ldots Q_t N_t$ 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 at least one component.

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 $\langle Q_1, \ldots, Q_t \rangle$ 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 $\langle Q_1, \ldots, Q_t \rangle$.

(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 $d_1, \ldots, d_k$ are integers $(+, -, 0)$ that the parameterized constants $D_1, \ldots, D_k$ 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 $d_1, \ldots, d_k$ 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 strongly reversal-bounded and not reversal-bounded) provided the input is one-way. As long as the unrestricted counter does not participate in any linear conditions, Theorem 4.5 can be extended to the following:

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 can 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

Up to now we have only considered conditionals. Suppose we allow a component of a parallel assignment instruction to be of the form: $x := y$ or $x := 0$, where x, y are counters. When such assignments are allowed, the notion of (strongly) reversal boundedness remains the same except that counter values may be incremented or decremented by some number greater than 1. For example, the following sequence is possible:

001234305677700,

where the third and fourth occurrences of 0 are results of the counter being reset to 0 and the occurrence of 5 is the result of an assignment acquiring the value of another counter. The sequence has 6 strong reversals that are marked: $00\bar{1}234\bar{3}0\bar{5}67\bar{7}7\bar{0}\bar{0}$ (assuming the initial mode is stay). Then we can establish the following result.

Theorem 4.8. *The emptiness problem is decidable for nondeterministic one-way machines with one unrestricted counter and several strongly reversal-bounded counters and with linear-relation conditionals on the reversal-bounded counters and parameterized constants*, *provided that the only assignments* (*components of parallel assignments*) *used are of the form:* $x := x + 1$, $x := x - 1$, $x := 0$, *or* $x := y$, *and the unrestricted counter cannot appear in any assignment of the form* $x := 0$ *or* $x := y$.

Proof. Briefly, the proof is done by direct simulations of the new assignments of the form $x := 0$ and $x := y$. With a careful accounting of the reversals, it is shown that the simulations are also strongly reversal bounded. By Theorem 4.6, the decidability result follows.

Assignments of the form $x := 0$ on counter x will be simulated by the following process. If x is already 0, no operations are needed; otherwise x is reset to 0 by a sequence of assignments $x := x - 1$, while all other counters wait. The process will add at most 1 strong reversal to X and at most 2 reversals to each of the other counters. Clearly, if there are no further assignments of the form $x := 0$ or $x := y$ to be executed, the total number of strong reversals for the entire computation is increased by at most 2. Suppose that this is not the case and consider the immediate next $x := 0$ or $x := y$ assignment. We consider the case of $x := 0$, the other case is similar. Two possibilities could arise: either x remains 0 between the two assignments or x becomes non-0 at some point. In the former case, clearly the second $x := 0$ assignment adds no additional strong reversals, while in the latter case, there must be at least one strong reversal between the two assignments and we can "charge" the two additional strong reversals to the strong reversal in the original computation. It either case, it is easy to see that the total number of strong reversals is increased by a factor of at most 2 and still bounded.

We now consider assignments of the form $x := y$. Again this is done by simulating the assignment. Consider an assignment $x := y$. If $x = y$, no additional operations are needed. Otherwise, either $x < y$ or $x > y$. We will then increment x ($x := x + 1$) or, respectively, decrement x ($x := x - 1$) until $x = y$, while other counters remain unchanged. Note that the tests can be done using linear conditionals. For counters other than x, the simulation adds at most two strong reversals and at most one strong reversal for x. Using a similar reasoning as the above, it suffices to show that if in the original computation x has no more reversals from this point on, in the simulated computation x will have a bounded number of reversals. Without loss of generality, we assume that x is in the increasing mode. We argue that x can only execute at most C number of assignments of the form $x := y$ such that $y > x + 1$ where C is the total number of counters. (However, there is no bound on the number of executions of assignments of the form $x := y$ when y happens to be $x + 1$. In this case, the assignment is equivalent to $x := x + 1$.) Indeed once x gets the largest value among all counters, any additional assignment of the form $x := y$ will introduce a strong reversal. □

The above result can be further generalized to include assignments of the form $x := c$ where c is some constant. The proof technique is the same. The requirement that the machine can only use assignments of the form: $x := x + 1$, $x := x - 1$, $x := c$ 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 Theorem 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 [9, 6, 23]. 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 $\{1, 2, \ldots, s\}$ for some s. Each counter can be incremented by integer constants $(+, -, 0)$ and can be tested if $<, >, =$ to integer constants. Let $(j, v_1, \ldots, v_k)$ denote the configuration of M when it is in state j, and counter i has value v_i for $i = 1, 2, \ldots, k$. Thus, the set of all possible configurations is a subset of $\mathbb{N}^{k+1}$. We use the symbols $\alpha, \beta, \ldots$ to denote configurations.

Given M, let $R(M) = \{(\alpha, \beta) \mid \alpha \text{ can reach } \beta \text{ in 0 or more moves}\}$. $R(M)$, 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 $(j, v_1, \ldots, v_k)$ in $\mathbb{N}^{k+1}$ can be represented as a string $j\%v_1\% \cdots \%v_k$, where $j, v_1, \ldots, v_k$ 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 $\mathbb{N}^k$ is accepted by a multicounter machine M, we mean that M, when started in its initial state with its first k counters (M can have more than k counters) set to an input k-tuple, accepts (i.e., enters an accepting state) if and only if the k-tuple is in S. Note that this is equivalent to equipping the machine with an input tape that contains the unary encoding of the k-tuple.

The proof of the following theorem uses results in [14, 12].

Theorem 5.1. *Let S be a subset of $\mathbb{N}^k$. Then the following statements are* (*effectively*) *equivalent*:
(1) *S can be accepted by a nondeterministic machine with one unrestricted counter and several reversal-bounded counters.*
(2) *S can be accepted by a nondeterministic reversal-bounded multicounter machine.*
(3) *S can be accepted by a deterministic reversal-bounded multicounter machine.*

Proof. S definable by a Presburger formula is equivalent to S being a semilinear set and, clearly, it is straightforward to construct a nondeterministic reversal-bounded multicounter machine accepting a semilinear set. It follows from Theorems 3.2 and 3.5 that (1)–(3) are equivalent. To complete the proof, we need to show that (1) implies (4).

Suppose S is a semilinear set. We describe the construction of a deterministic reversal-bounded multicounter machine accepting S. Since S a finite union of linear sets, it is sufficient to show the construction when S is a linear set. Let

$$S = \left\{ v_0 + \sum_{j=1}^{n} t_j v_j \mid t_j \geqslant 0 \right\},$$

where $v_j \in \mathbb{N}^k$. Given $\bar{x} = (x_1, \ldots, x_k)^{\mathrm{T}}$ in $\mathbb{N}^k$, to determine whether it is in S, we need to find $t_1, \ldots, t_n \in \mathbb{N}$ such that $\bar{x} = v_0 + \sum_{j=1}^{n} t_j v_j$. This process involves solving a diophantine system (in the nonnegative integer variables $t_1, \ldots, t_n$). The efficient solution of such a system depends on the following result [12]:

Let $V\bar{y} = \bar{b}$ be a system of linear equations, where V is a $k \times n$ integral matrix, $\bar{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ is a column vector of variables, and $\bar{b} = (b_1, \ldots, b_k)^{\mathrm{T}}$ is an integral column vector. Let r be the rank of V. Denote by Ψ the maximum of the absolute values of all $r \times r$ subdeterminants of V. If the system has a nonnegative integral solution, then it has a nonnegative integral solution $(\hat{y}_1, \ldots, \hat{y}_n)^{\mathrm{T}}$ such that for some set of indices $L \subseteq \{l_1, \ldots, l_r\} \subseteq \{1, \ldots, n\}$, $\hat{y}_i < \Psi$ for each $i \notin L$. Moreover, the submatrix formed by columns $l_1, \ldots, l_r$ of V is nonsingular (i.e., it has rank r).

Thus with the semilinear graph is associated a finite set of nonsingular systems each of which arises by "predetermining" some of the t_j's. For each $\bar{x} = (x_1, \ldots, x_k)^{\mathrm{T}} \in \mathbb{N}^k$, one need only try each such system to see if the remaining t_j's are solvable in the nonnegative integers.

The solution of each of the nonsingular systems can be effected by applying Cramer's rule to a square nonsingular subsystem. Thus each "nonpredetermined" t_j can be written as

$$t_j = \frac{\sum a_p y_p - \sum b_q z_q + c}{\Delta},$$

where y_p, z_q are components of $\bar{x}, a_p, b_q$ and Δ are positive integers, and c is an integer. Here a_p, b_q and Δ depend only on the vectors defining the underlying linear set, whereas c depends on the predetermined t_j's as well. Clearly, computing the t_j's (from equations of the form given above) and checking that they are all nonnegative integers can be accomplished by a multicounter machine by reading the input x_i's stored initially in the counters, and using a finite number of auxiliary reversal-bounded counters. The machine can then check that these t_j's satisfy any rows that were deleted to obtain the square nonsingular subsystem. Again, this checking requires only a finite number of reversal-bounded counters. □

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 $post^*_M(S)$ (or $pre^*_M S$)) is definable by a Presburger formula.*
(3) *If S is Presburger, then $post^*_M(S)$ (or $pre^*_M(S)$) 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 α. At some point, M' guesses that it has reached configuration β, which it 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 α in its counters, simulates M starting in this configuration. At some point, M_{pre} guesses that it has reached a configuration β 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 β in its counters, "guesses" and stores a configuration α in its counters. M_S than checks that α is accepted by M_{pre} and that β is reachable from α.

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 $(j, v_1, \ldots, v_k, d_1, \ldots, d_m)$, where $d_1, \ldots, d_m$ represent the values of the parameterized constants. Then the next 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 with 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 that 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) [17], which is to decide for a given polynomial $P(x_1, \ldots, x_n)$ with integer coefficients whether it has a nonnegative integral root.

First consider a term $st(x_1, \ldots, x_n) = sx_1^{i_1} \cdot x_n^{i_n}$ of the polynomial $P(x_1, \ldots, x_n)$, where $s = +$ or $-, i_1, \ldots, i_n \geqslant 0$. 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 $A_1, \ldots, A_n, B$ such that M_t with the constants assigned nonnegative integer values $\alpha_1, \ldots, \alpha_n, \beta$ accepts if and only if $\beta = \alpha_1^{i_1} \ldots \alpha_n^{i_n}$.

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 $i_1, \ldots, i_n$ 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 α_1. Then M_t computes α_1^2 in C_2 by iterating the following process until C_1 becomes zero: (1) Add α_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 $\alpha_1^{i_1}\dots\alpha_n^{i_n}$ in one of the counters, say C_2. M_t then sets C_1 to β and checks that C_1 is equal to C_2. Note that the counters C_1 and C_2 are reversal-bounded.

Now let $P(x_1,\dots,x_n)=s_1t_1+\cdots+s_rt_r$, where each term $s_jt_j=s_jx_1^{i_{j1}}\dots x_n^{i_{jn}}$. We construct a deterministic machine M_P with one unrestricted counter, two reversal-bounded counters, and parameterized constants $A_1,\dots,A_n,B_1,\dots,B_r$ such that M_P with the constants assigned nonnegative integer values $\alpha_1,\dots,\alpha_n$, $\beta_1,\dots,\beta_r$ accepts if and only if $\beta_j=\alpha_1^{i_{j1}}\dots\alpha_n^{i_{jn}}$ and $s_1\beta_1+\cdots+s_r\beta_r=0$. The integers $i_{11},\dots,i_{1n},\dots,i_{r1},\dots,i_{rn}$ and signs $s_1,\dots,s_r$ are stored in the states of M_P. M_P uses the technique described above to check that $\beta_j=\alpha_1^{i_{j1}}\dots\alpha_n^{i_{jn}}$ 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 $A_1,\dots,A_n$. Assume without loss of generality that the values $\alpha_1,\dots,\alpha_n$ 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 [15]. M' has input alphabet $\{1,\%\}$ (the delimiter is #). M' rejects all inputs not of the form $\#1^{i_1}\%1^{i_2}\%\cdots1^{i_n}\%1^k\#$.

Corresponding to values $\alpha_1,\dots,\alpha_n$ assigned to the parameterized constants, M' is given input $w=\#1^{i_1}\%1^{i_2}\%\cdots1^{i_n}\%1^k\#$, where

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

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 ith % corresponds to U being equal to parameterized constant α_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 α_n. It follows that M' accepts w (for some k) if and only if M accepts when the parameterized constants are assigned values $\alpha_1,\dots,\alpha_n$. We now show that the padding 1^k can be removed.

If we can show that $k\leqslant c\alpha_n$ for some positive integer c (note that the length of the input to M' is α_n), then M' can use the input to "simulate" the action of U when the counter has value greater than α_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 α_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 α_n by more than s; otherwise, M will be in an infinite loop. The only way that U can increase its count beyond α_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 $\alpha_n + sv$. 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 $s(\alpha_n + s)$ moves. Thus, v can have at most value $s(\alpha_n + s)$ 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 $s(\alpha_n + s) + s(s(\alpha_n + s))$. Since M is reversal-bounded, the biggest number v that can be stored in C without 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 $c\alpha_n$ 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' that, when given a two-way input tape $\#1^{\alpha}\#$, accepts if and only if M with parameterized constant A set to α accepts. The result follows, since the emptiness problem is decidable for deterministic two-way reversal-bounded multicounter machines over unary input [13]. □

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 θ 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 configurations 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.

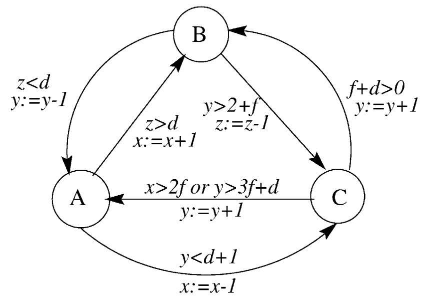

Fig. 2. An infinite-state transition system.

In practice, many infinite-state transition systems can be modeled by multicounter machines, like the transition graph shown in Fig. 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 α and β such that α can reach β, if $d > f$ then counter x in β is greater than the sum of counters y and z in α.

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 that are not guaranteed to terminate [23] or to look at restricted classes of systems where reachability is decidable [6].

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 that either increase with rate 1 or reset to 0, as in timed automata [2], 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 [8], 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 reset bounded (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 reset bounded. When the clocks are not reset bounded, it can be shown that binary reachability is not computable. In fact, "node reachability" is not decidable [2]. 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 [7] 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

[1] P. Abdulla, B. Jonsson, Verifying programs with unreliable channels, Inform. Comput. 127 (2) (1996) 91–101.

[2] R. Alur, D. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.

[3] B. Baker, R. Book, Reversal-bounded multipushdown machines, J. Comput. System Sci. 8 (1974) 315–332.

[4] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model-checking, Proc. Internat. Conf. on Concurrency (CONCUR 1997), Lecture Notes in Computer Science, Vol. 1243, Springer, Berlin, 1997, pp. 135–150.

[5] G. Cece, A. Finkel, Programs with quasi-stable channels are effectively recognizable, Proc. 9th Internat. Conf. on Computer Aided Verification (CAV 1997), Lecture Notes in Computer Science, Vol. 1254, Springer, Berlin, 1997, pp. 304–315.

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

[7] Z. Dang, Binary reachability analysis of timed pushdown automata with dense clocks, Proc. 13th Internat. Conf. on Computer Aided Verification (CAV 2001), Lecture Notes in Computer Science, Vol. 2102, Springer, Berlin, 2001, pp. 506–518.

[8] Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su, Binary reachability analysis of discrete pushdown timed automata, Proc. 12th Internat. Conf. on Computer Aided Verification (CAV 2000), Lecture Notes in Computer Science, Vol. 1427, Springer, Berlin, 2000, pp. 69–84.

[9] J. Esparza, Decidability of model checking for infinite-state concurrent systems, Acta Inform. 34 (2) (1997) 85–107.

[10] A. Finkel, G. Sutre, Decidability of reachability problems for classes of two counter automata, Proc. 17th Internat. Conf. on Theoretical Aspects of Computer Science (STACS 2000), Lecture Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 346–357.

[11] S. Ginsburg, E. Spanier, Bounded Algol-like languages, Trans. Amer. Math. Soc. 113 (1964) 333–368.

[12] E.M. Gurari, O.H. Ibarra, An NP-complete number-theoretic problem, J. ACM 26 (1979) 567–581.

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

[14] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. ACM 25 (1978) 116–133.

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

[16] O.H. Ibarra, J. Su, Z. Dang, T. Bultan, R.A. Kemmerer, Counter machines: decidable properties and applications to verification problems, Proc. 25th Internat. Conf. on Mathematical Foundations of Computer Science (MFCS 2000), Lecture Notes in Computer Science, Vol. 1893, Springer, Berlin, 1998, pp. 426–435.

[17] Y. Matijasevic, Enumerable sets are Diophantine, Soviet Math. Dokl 11 (1970) 354–357.

[18] K.L. McMillan, Symbolic model-checking—an approach to the state explosion problem, Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, 1992.

[19] M. Minsky, Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines, Ann. Math. 74 (1961) 437–455.

[20] R. Parikh, On context-free languages, J. ACM 13 (1966) 570–581.

[21] W. Peng, S. Purushothaman, Analysis of a class of communicating finite state machines, Acta Inform. 29 (6/7) (1992) 499–522.

[22] L.G. Valiant, M.S. Paterson, Deterministic one-counter automata, J. Comput. System Sci. 10 (1975) 340–350.

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