Counter Machines: Decidable Properties and Applications to Verification Problems

Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard Kemmerer

Department of Computer Science, University of California, Santa Barbara, CA 93106, USA.
{ibarra, su, dang, bultan, kemm}@cs.ucsb.edu

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, safety, and invariance are solvable for these machines.

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 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 [12] 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) [9]. 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 [14].

In this paper, we study two-way finite automata augmented with finitely many counters. A restricted version of these machines was studied in [9], where: i) each counter is reversal-bounded in that it can be incremented or decremented by 1 and tested for zero, but the number of times it can change mode from nondecreasing to nonincreasing and vice-versa is bounded by a constant, and ii) the two-way input is finite-crossing in that the number of times the input head crosses the boundary between any two adjacent cells of the input tape is bounded by a constant (there is no bound on how long the head can remain on a cell).

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, safety, and invariance are solvable for these machines.

2 Reversal-Bounded Multicounter Machines

It is convenient to represent a counter machine as a program. The standard model of a deterministic two-way multicounter machine can be specified by a program M of the form

begin input $(\#w\#)$; P **end**.

Here w is the input with delimiters #, and P is a sequence of labeled instructions, where each instruction is of the form shown in Fig. 1, 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.

s : **read** (*INPUT*)	$s : x := x + 1$	s : **left**	s : **accept**
s : **goto** p	$s : x := x - 1$	s : **right**	s : **reject**
s : **if** *INPUT* $= a$ **then goto** p **else goto** q		s : **if** $x = 0$ **then goto** p **else goto** q	

Fig. 1. Instructions

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. An input $\#w\#$ is accepted (rejected) if M on this input halts in **accept** (**reject**). Note that the machine may not always halt. The set of all inputs accepted by M is denoted by $L(M)$.

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.

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 nondecreasing and nonincreasing modes). So, for example, a counter with the computation pattern "00000111111222222344444" has 0 reversals. On the other hand, "000001111112222223444443332221233444" 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. M is *one-way* if it is 1-crossing.

3 Fundamental Decidable Problems

We begin with the following theorem in [9].

Theorem 1. The emptiness problem is decidable for nondeterministic one-way reversal-bounded multicounter machines.

Nondeterministic finite-crossing machines can be converted to one-way (this is *not* true for the deterministic case). Hence:

Theorem 2. The emptiness problem is decidable for nondeterministic finite-crossing reversal-bounded multicounter machines.

We can generalize Theorem 1 to allow one of the counters to be unrestricted:

Theorem 3. The emptiness problem is decidable for nondeterministic one-way machines with one unrestricted counter and several reversal-bounded counters.

Theorem 4. The infiniteness and disjointness problems are decidable for nondeterministic finite-crossing reversal-bounded multicounter machines.

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 [2]. However for deterministic machines, we can prove:

Theorem 5. The containment and equivalence problems are decidable for deterministic finite-crossing reversal-bounded multicounter machines.

4 Generalizations

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 θ is one of $<, >, =$. One can easily show that any (reversal-bounded) multicounter machine M that uses these generalized instructions can be converted to an equivalent (reversal-bounded) standard model M' such that $L(M) = L(M')$.

4.2 Linear conditions

We can further allow 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.

We can allow a multicounter machine M to use conditionals (tests) of the form "s : **if** L **then goto** p **else goto** q," where L is a linear relation on the counters. Unfortunately, the halting (and, hence, the emptiness) problem is undecidable for reversal-bounded multicounter machines that allow linear-relation conditionals. In fact, the undecidability holds even in the case of only 3 counters:

Theorem 6. 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 [12] 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):

$$(q_1, 0, d_1, 0),$$
$$(q_2, d_1 + d_2, d_1, d_1),$$
$$(q_3, d_1 + d_2, d_1 + d_2 + d_3, d_1 + d_2),$$
$$(q_4, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3, d_1 + d_2 + d_3),$$
$$(q_5, d_1 + d_2 + d_3 + d_4, d_1 + d_2 + d_3 + d_4 + d_5, d_1 + d_2 + d_3 + d_4), \ldots$$

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 since we can show:

Theorem 7. 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 $=$.

There is a stronger notion of reversal-boundedness. A counter with the computation pattern "00000111111222222344444" 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 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.

Note that while the machine M' in the construction in Theorem 6 is reversal-bounded, it is not strongly reversal-bounded. However, we can prove the following:

Theorem 8. 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 $1 \leqslant i \leqslant t-1$, Q_{i+1} differs from Q_i in exactly one component (i.e., exactly one counter changes mode), and (4) The starting mode-vector $Q_1 = \langle$no-change, ..., 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 8. 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 "**if** L **then goto** p **else goto** q" in the program of M, where L is a linear relation. W.l.o.g, we assume that 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 : **if** L **then goto** p **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 : **if** L **then goto** p **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 C 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 9. The emptiness problem is decidable for nondeterministic finite-crossing strongly reversal-bounded multicounter machines using linear-relation conditionals on the counters and parameterized constants.

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:

Theorem 10. 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 6, none of Theorems 8–10 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 the results of Theorems 8–10 hold for reversal-bounded machines (which are not necessarily strongly reversal-bounded) that use such restricted linear relations.

Finally, we state that Theorems 4 and 5 can be generalized:

Theorem 11. The infiniteness, disjointness, containment, and equivalence problems are decidable for various generalizations introduced in this section (with containment and equivalence holding only for deterministic machines).

5 Reachability, Safety, and Invariance

The results of the previous section can be used to analyze verification problems (such as reachability, safety, and invariance) 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 [7, 3, 15]. 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 $\mathbb{N}^{k+1}$.

Given M, let $R(M) = \{(j, v_1, ..., v_k, j', v'_1, ..., v'_k) \mid$ configuration $(j, v_1, ..., v_k)$ can reach configuration $(j', v'_1, ..., v'_k)$ 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, *pre*$*_M(S)$ and *pre*$*_M(S)$ can be viewed as languages (e.g., regular, context-free, etc.). Similarly, $R(M)$ can be viewed as a language.

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 state the following characterization theorem without proof.

Theorem 12. Let M be a nondeterministic reversal-bounded k-counter machine and S a subset of $\mathbb{N}^{k+1}$.

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. *post*$*_M(S)$ (*pre*$*_M(S)$) can be accepted by a deterministic reversal-bounded multicounter machine.
4. 1–3 still hold even if one of the counters is unrestricted.

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 13. The statements in Theorem 12 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 14. Consider deterministic machines with one 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.

The problems of *safety* and *invariance* are of importance in the area of verification. The following theorem follows from Theorems 12 and 13.

Theorem 15. In the following, M is a nondeterministic reversal-bounded multicounter machine and S and T are two sets of configurations accepted by nondeterministic reversal-bounded multicounter machines:

1. It is decidable to determine whether there is no configuration in S that can reach a configuration in T. Thus, safety is decidable with respect to a bad set of configurations T.
2. It is decidable to determine whether every configuration in S can only reach configurations in T. Thus, invariance is decidable with respect to a good set T.

6 Conclusions

We 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. In practice, many infinite-state transition systems can be modeled by multicounter machines. In order to debug a safety property, a multicounter machine can be effectively made reversal-bounded by giving a bound on reversals. This is a new approach for analyzing safety properties where the general reachability problem is known to be undecidable. Other approaches use semi-decision algorithms that are not guaranteed to terminate [15] or to look at restricted classes of systems where reachability is decidable [3].

It may seem that strong reversal-boundness 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 [1], strong reversal-boundness is equivalent to reversal-boundness. Using this observation and the results in this paper, we are able to show a number of results on clocked systems with parameterized durations [4], binary reachability characterization of discrete timed pushdown automata [5], and reachability of past machines [6].

Acknowledgment The work by Ibarra and Su was supported in part by NSF grant IRI-9700370; the work by Bultan was supported in part by NSF grant CCR-9970976; the work by Dang and Kemmerer was 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.

References

1. R. Alur and D. Dill. "A theory of timed automata," *Theo. Comp. Sci.*, 126(2):183-235, 1994.
2. B. Baker and R. Book. "Reversal-bounded multipushdown machines," *J.C.S.S.*, 8:315-332, 1974.
3. H. Comon and Y. Jurski. "Multiple counters automata, safety analysis and Presburger arithmetic," *Proc. Int. Conf. on Computer Aided Verification*, pp. 268-279, 1998.
4. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. "Decidable approximations on discrete clock machines with parameterized durations," in preparation.
5. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. "Binary reachability analysis of discrete pushdown timed automata," to appear in CAV 2000.
6. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. "Past machines," in preparation.
7. J. Esparza. "Decidability of model checking for infinite-state concurrent systems," *Acta Informatica,* 34(2):85-107, 1997.
8. E. M. Gurari and O. H. Ibarra. "Simple counter machines and number-theoretic problems," *J.C.S.S.*, 19:145-162, 1979.
9. O. H. Ibarra. "Reversal-bounded multicounter machines and their decision problems," *J. ACM*, 25:116-133, 1978.
10. 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.
11. Y. Matijasevic. "Enumerable sets are Diophantine," *Soviet Math. Dokl*, 11:354-357, 1970.
12. 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.
13. R. Parikh. "On context-free languages," *J. ACM*, 13:570-581, 1966.
14. L. G. Valiant and M. S. Paterson. "Deterministic one-counter automata," *J.C.S.S.,* 10:340-350, 1975.
15. P. Wolper and B. Boigelot. "Verifying systems with infinite but regular state spaces," *Proc. 10th Int. Conf. on Computer Aided Verification*, pp. 88–97, 1998.