A Solvable Class of Quadratic Diophantine Equations with Applications to Verification of Infinite-State Systems

Gaoyan Xie[1], Zhe Dang[1] *, and Oscar H. Ibarra[2]**

[1] School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164, USA
[2] Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

Abstract. A *k-system* consists of k quadratic Diophantine equations over nonnegative integer variables $s_1, ..., s_m, t_1, ..., t_n$ of the form:

$$\sum_{1 \le j \le l} B_{1j}(t_1, ..., t_n) A_{1j}(s_1, ..., s_m) = C_1(s_1, ..., s_m)$$
$$\vdots$$
$$\sum_{1 \le j \le l} B_{kj}(t_1, ..., t_n) A_{kj}(s_1, ..., s_m) = C_k(s_1, ..., s_m)$$

where l, n, m are positive integers, the B's are nonnegative linear polynomials over $t_1, ..., t_n$ (i.e., they are of the form $b_0 + b_1 t_1 + ... + b_n t_n$, where each b_i is a nonnegative integer), and the A's and C's are nonnegative linear polynomials over $s_1, ..., s_m$. We show that it is decidable to determine, given any 2-system, whether it has a solution in $s_1, ..., s_m, t_1, ..., t_n$, and give applications of this result to some interesting problems in verification of infinite-state systems. The general problem is undecidable; in fact, there is a fixed $k > 2$ for which the k-system problem is undecidable. However, certain special cases are decidable and these, too, have applications to verification.

1 Introduction

During the past decade, there has been significant progress in automated verification techniques for finite-state systems. One such technique is model-checking [5, 19] that explores the state space of a finite-state system and checks that a desired temporal property is satisfied. Model-checkers like SMV [13] and SPIN [10] have been successful in many industrial-level applications. The successes have greatly inspired researchers to develop automatic techniques for analyzing infinite-state systems (such as systems that contain integer variables and parameters). However, in general, it is not possible to develop such techniques, e.g., it is not possible to (automatically) verify whether an arithmetic program with two integer variables is going to halt [17]. Therefore, an important

* Corresponding author (`zdang@eecs.wsu.edu`).
** The research of Oscar H. Ibarra has been supported in part by NSF Grants IIS-0101134 and CCR02-08595.

aspect of the research on infinite-state system verification is to identify what kinds of practically useful infinite-state models are decidable with respect to a particular form of properties (e.g., reachability).

In this paper, we look at a class of infinite-state systems that contain parameterized or unspecified constants. For instance, consider a nondeterministic finite state machine M. Each transition in M is assigned a label. On firing the transition $s \rightarrow^a s'$ from state s to state s' with label a, an *activity* a is performed. There are finitely many labels $a_1, \ldots, a_l$ in M. M can be used to model, among others, a finite state process where an execution of the process corresponds to an execution path (e.g., $s_0 \rightarrow^{a^0} s_1 \rightarrow^{a^1} \ldots \rightarrow^{a^r} s_{r+1}$, for some r) in M. On the path, a sequence of activities $a^0 \ldots a^r$ are performed. Let $\Sigma_1, \ldots, \Sigma_k$ be any k sets (not necessarily disjoint) of labels. An activity a is of type i if $a \in \Sigma_i$. An activity could have multiple types. Additionally, activities $a_1, \ldots, a_l$ are associated with *weights* $w_1, \ldots, w_l$ that are unspecified (or parameterized) constants in $\mathbf{N}$, respectively. Depending on the various application domains, the weight of an activity can be interpreted as, e.g., the time in seconds, the bytes of memory, or the budget in dollars, etc., needed to complete the activity. A type of activities is useful to model a "cluster" of activities. When executing M, we use nonnegative integer variables W_i to denote the accumulated weight of all the activities of type i performed so far, $1 \leq i \leq k$. One verification question concerns reachability:

> (*) whether, for some values of the parameterized constants $w_1, \ldots, w_l$, there is an execution path from a given state to another on which $w_1, \ldots, w_l, W_1, \ldots, W_k$ satisfy a given Presburger formula P (a Boolean combination of linear constraints and congruences).

One can easily find applications for the verification problem. For instance, consider a packet-based network switch that uses a scheduling discipline to decide the order in which the packets from different incoming connections $c_1, ..., c_l$ are serviced (visited). Suppose that each connection c_i is assigned a weight w_i, $1 \leq i \leq l$, and each time when a connection is serviced (visited), the number of packets serviced from that connection is in proportion to its weight. But in this switch we have two outgoing connections (two servers in the "queue theory" jargon) o_1 and o_2 each of which serves a set of incoming connections C_1 and C_2 respectively ($C_1 \cup C_2 = \{c_1, ..., c_l\}$). The scheduling discipline for this switch can be modeled as a finite state system. If we take the event that an incoming connection is serviced by a specific server as an activity, then the weight of the activity could be the number of packets served that is in proportion to the weight of the incoming connection. Thus W_1 and W_2 could be used to denote the total amount (accumulated weights) of packets served by the two servers respectively. Later in the paper, we shall see how to model a fairness property using (*).

In this paper, we study the verification problem in (*) and its variants. First, we show that the problem is undecidable, in general. Then, we consider various restricted as well as modified cases in which the problem becomes decidable. For instance, if P in (*) has only one linear constraint that contains some of $W_1, \ldots, W_k$, then the problem is decidable. Also, rather surprisingly, if in the problem in (*) we assume that the weight of each activity a_i can be nondeterministically chosen as any value between a concrete constant (such as 5) and a parameterized constant w_i, then it becomes decidable. We also consider cases when the transition system is augmented with other unbounded data structures, such as a pushdown stack, dense clocks, and other restricted counters.

In the heart of our decidability proofs, we first show that some special classes of systems of quadratic Diophantine equations/inequalities are decidable (though in general, these systems are undecidable [16]). This nonlinear Diophantine approach towards verification problems is significantly different from many existing techniques for analyzing infinite-state systems (e.g., automata-theoretic techniques in [14, 3, 7] , computing closures for Presburger transition systems [6, 4], etc.). Then, we study a more general version of the verification problem by considering weighted semilinear languages in which a symbol is associated with a weight. Using the decidability results on the restricted classes of quadratic Diophantine systems, we show that various verification problems concerning weighted semilinear languages are decidable. Finally, as applications, we "re-interpret" the decidability results for weighted semilinear languages into the results for some classes of machine models, whose behaviors (e.g., languages accepted, reachability sets, etc) are known to be semilinear, augmented with weighted activities.

Adding weighted activities to a transition system can be found, for instance, in [15]. In that paper, a "price" is associated with a control state in a timed automaton [2]. The price may be very complex; e.g., linear in other clock values etc. In general, the reachability problem for priced timed automata is undecidable [15]. Here, we are mainly interested in the decidable cases of the problem: what kind of "prices" (i.e., weights) can be placed such that some verification queries are still decidable, for transition systems like pushdown automata, restricted counter machines, etc., in addition to timed automata.

The paper is organized as follows. In the next section, we present the decidability results for the satisfiability problem of two special classes of quadratic Diophantine systems (Lemma 2 and Theorem 1). Then in Section 3, we generalize the verification problem in (*) in terms of weighted semilinear languages, and reduce the problem and its restricted versions to the classes of quadratic Diophantine systems studied in Section 2. In Section 4, we discuss the application aspects and extensions of the decidability results to other machine models. Due to space limitation, some of the proofs are omitted in the paper. The full version of the paper is accessible at `www.eecs.wsu.edu/~zdang`.

2 Preliminaries

Let $\mathbf{N}$ be the set of nonnegative integers and let $x_1, \ldots, x_n$ be n variables over $\mathbf{N}$. A *linear constraint* is defined as $a_1x_1 + \ldots + a_nx_n > b$, where $a_1, \ldots, a_n$ and b are integers. A *congruence* is $x_i \equiv_b c$, where $1 \leq i \leq n$, and $b \neq 0, 0 \leq c < b$. A Presburger formula is a Boolean combination of linear constraints and congruences using $\vee$ and $\neg$. Notice that, here, Presburger formulas are defined over nonnegative integer variables (instead of integer variables). It is well known that Presburger formulas are closed under quantifications ($\forall$ and $\exists$).

A subset S of $\mathbf{N}^n$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^n$ such that $S = \{v | v = v_0 + b_1v_1 + \ldots + b_tv_t, b_i \in \mathbf{N}\}$. The set S is a *semilinear set* if it is a finite union of linear sets. It is well known that S is semilinear iff S is Presburger definable (i.e., there is a Presburger formula P such that $P(v)$ iff $v \in S$).

A *linear polynomial* is a polynomial of the form $a_0 + a_1x_1 + ... + a_nx_n$ where each coefficient a_i, $0 \le i \le n$, is an integer. The polynomial is *constant* if each $a_i = 0$, $1 \le i \le n$. The polynomial is *nonnegative* if each a_i, $0 \le i \le n$, is in $\mathbf{N}$. The polynomial is *positive* if it is nonnegative and $a_0 > 0$. A variable *appears* in a linear polynomial iff its coefficient in that polynomial is nonzero. The following result is needed in the paper.

Lemma 1. *It is decidable whether an equation of the following form has a solution in nonnegative integer variables $s_1, \ldots, s_m, t_1, \ldots, t_n$:*

$$L_0 + L_1t_1 + \ldots + L_nt_n = 0 \tag{1}$$

where $L_0, L_1, \ldots, L_n$ are linear polynomials over $s_1, \ldots, s_m$. The decidability remains even when the solution is restricted to satisfy a given Presburger formula P over $s_1, \ldots, s_m$.

Proof. The first part of the lemma has already been proved in [8], while the second part is shown below using a "semilinear transform". As we mentioned earlier, the set of all $(s_1, \ldots, s_m) \in \mathbf{N}^m$ satisfying P is a semilinear set (i.e., a finite union of linear sets). For each linear set of P, one can find nonnegative integer variables $u_1, \ldots, u_k$ for some k and a nonnegative linear polynomial $p_i(u_1, \ldots, u_k)$ for each $1 \le i \le m$ such that $(s_1, \ldots, s_m)$ is in the linear set iff each $s_i = p_i(u_1, \ldots, u_k)$, for some $u_1, \ldots, u_k$. The second part follows from the first part by substituting $p_i(u_1, \ldots, u_k)$ for s_i in $L_0, L_1, \ldots, L_n$. ∎

Let I, J and K be three pairwise disjoint subsets of $\{1, \ldots, n\}$. An *n-inequality* is an inequality over n nonnegative integer variables $t_1, \ldots, t_n$ and m (for some m) nonnegative integer variables $s_1, \ldots, s_m$ of the following form:

$$D_1 + a(\sum_{i \in I} L_{1i}t_i + \sum_{j \in J} L_{1j}t_j) \le D_2 + \sum_{i \in I} L_{2i}t_i + \sum_{k \in K} L_{2k}t_k$$

$$\le D_1' + a'(\sum_{i \in I} L_{1i}t_i + \sum_{j \in J} L_{1j}t_j), \tag{2}$$

where $a < a' \in \mathbf{N}$, the D's (resp. the L's) are nonnegative (resp. positive) linear polynomials over $s_1, \ldots, s_m$, and $D_1 \le D_1'$ is always true (i.e., true for all $s_1, \ldots, s_m \in \mathbf{N}$).

Lemma 2. *For any n, it is decidable whether an n-inequality in (2) has a solution in nonnegative integer variables $s_1, \ldots, s_m, t_1, \ldots, t_n$. The decidability remains even when the solution is restricted to satisfy a given Presburger formula P over $s_1, \ldots, s_m$.*

Theorem 1. *It is decidable whether a system in the following form has a solution in nonnegative integer variables $s_1, \ldots, s_m$, $t_1, \ldots, t_n$: $P(D_1 + \sum_{1 \le i \le n} L_{1i}t_i, D_2 + \sum_{1 \le i \le n} L_{2i}t_i)$, where P is a Presburger formula over two nonnegative integer variables and the D's and the L's are nonnegative linear polynomials over $s_1, \ldots, s_m$.*

3 Semilinear Languages with Weights

We first recall the definition of semilinear languages. Let $\Sigma = \{a_1, \ldots, a_l\}$ be an alphabet. For each word α in Σ^*, the *Parikh* map of α is defined to be $\phi(\alpha) = (\phi_{a_1}(\alpha), \ldots, \phi_{a_l}(\alpha))$ where $\phi_{a_i}(\alpha)$ denotes the number of occurrences of symbol a_i in word α, $1 \leq i \leq l$. For a language $L \in \Sigma^*$, the Parikh map of L is $\phi(L) = \{\phi(\alpha) | \alpha \in L\}$. The language L is semilinear iff $\phi(L)$ is a semilinear set. L is effectively semilinear if the semilinear set $\phi(L)$ can be computed from the description of L.

Now, we add "weights" to a language L. A *weight measure* is a mapping that maps a symbol in Σ to a weight in $\mathbf{N}$. We shall use $w_1, \ldots, w_l$ to denote the weights for $a_1, \ldots, a_l$, respectively, under the measure. Let $\Sigma_1, \ldots, \Sigma_k$ be any k fixed subsets of Σ. For each $1 \leq i \leq k$, we use $W_i(\alpha)$ to denote the total weight of all the occurrences for symbols $a \in \Sigma_i$ in word α; i.e.,

$$W_i(\alpha) = \sum_{a_j \in \Sigma_i} w_j \cdot \phi_{a_j}(\alpha). \tag{3}$$

$W_i(\alpha)$ is called the *accumulated weight* of α wrt Σ_i. We are interested in the following *k-accumulated weight problem*:

- **Given:** An effectively semilinear language L, k subsets $\Sigma_1, \ldots, \Sigma_k$ of Σ, and a Presburger formula P over $l + k$ variables.
- **Question:** Is there a word α in L such that, for some $w_1, \ldots, w_l \in \mathbf{N}$,

  $$P(w_1, \ldots, w_l, W_1(\alpha), \ldots, W_k(\alpha)) \tag{4}$$

  holds?

In a later section, we shall look at the application side of the problem. The rest of this section investigates the decidability issues of the problem by transforming the problem and its restricted versions to a class of Diophantine equations.

A *k-system* is a quadratic Diophantine equation system that consists of k equations over nonnegative integer variables $s_1, ..., s_m, t_1, ..., t_n$ (for some m, n) in the following form:

$$\begin{cases} \sum_{1 \leq j \leq l} B_{1j}(t_1, ..., t_n) A_{1j}(s_1, ..., s_m) = C_1(s_1, ..., s_m) \\ \vdots \\ \sum_{1 \leq j \leq l} B_{kj}(t_1, ..., t_n) A_{kj}(s_1, ..., s_m) = C_k(s_1, ..., s_m) \end{cases} \tag{5}$$

where the A's, B's and C's are nonnegative linear polynomials, and l, n, m are positive integers.

Theorem 2. *For each k, the k-accumulated weight problem is decidable iff it is decidable whether a k-system has a solution.*

It is known [12] that there is a fixed k such that there is no algorithm to solve Diophantine systems in the following form: $t_1 F_1 = G_1, t_1 H_1 = I_1, \ldots, t_k F_k = G_k, t_k H_k = I_k$, where the F's, G's, H's, I's are nonnegative linear polynomials over nonnegative integer variables $s_1, \ldots, s_m$, for some m. Observe that the above systems are $2k$-systems. Therefore, from Theorem 2,

Theorem 3. *There is a fixed k such that the k-accumulated weight problem is undecidable.*

Currently, it is an open problem to find the maximal k such that the k-accumulated weight problem is decidable. Clearly, when $k = 1$, the problem is decidable. This is because 1-systems are decidable (Lemma 1). Below, using Theorem 1, we show that the problem is decidable when $k = 2$. Interestingly, it is still open whether the decidability remains for $k = 3$.

Theorem 4. *The 2-accumulated weight problem is decidable.*

In some restricted cases, the accumulated weight problem is decidable for a general k. We are now going to elaborate these cases. Consider a k-accumulated weight problem such that (4) is a disjunction of formulas in the following special form:

$$Q(w_1, \ldots, w_l) \ \wedge \ a_1 W_1(\alpha) + \ldots + a_k W_k(\alpha) + b_1 w_1 + \ldots + b_l w_l \sim a_0 \qquad (6)$$

where Q is a Presburger formula over l variables, the a's and b's are integers, and $\sim \in \{=, \neq, >, <, \geq, \leq\}$. Under this restriction, the k-accumulated weight problem is decidable.

Theorem 5. *For each k, the k-accumulated weight problem, in which (4) is a disjunction of formulas in the form of (6), is decidable.*

Currently we do not know whether Theorem 5 still holds if (6) is conjuncted with one additional inequality: $a'_1 W_1(\alpha) + \ldots + a'_k W_k(\alpha) + b'_1 w_1 + \ldots + b'_l w_l \sim a'_0$.

As in the statement of the problem at the beginning of this section, a weight measure assigns numbers $w_1, \ldots, w_l$ to symbols $a_1, \ldots, a_l$ respectively. Instead of a fixed one, suppose that the weight of a symbol a_i can take any value between a given number q_i and w_i. That is, the weight measure defines a possible weight range that a symbol can have, with the given number q_i being the lowest possible weight. Thus, in contrast to (3), $W_i(\alpha)$, $1 \leq i \leq l$, will be a set:

$$\{\hat{W}_i : \sum_{a_j \in \Sigma_i} q_j \cdot \phi_{a_j}(\alpha) \leq \hat{W}_i \leq \sum_{a_j \in \Sigma_i} w_j \cdot \phi_{a_j}(\alpha)\}. \qquad (7)$$

For instance, suppose $\Sigma_1 = \{a_1\}$, $q_1 = 2$, $w_1 = 7$, and a word $\alpha = a_1 a_1 a_1$. Clearly, 12 is a weight in $W_1(\alpha)$ according to (7).

With the new definition of $W_i(\alpha)$, the following *loose* k-accumulated weight problem can be formulated:

- **Given:** An effectively semilinear language L, numbers $q_1, \ldots, q_l \in \mathbf{N}$, k subsets $\Sigma_1, \ldots, \Sigma_k$ of Σ, and a Presburger formula P over $l + k$ variables.
- **Question:** Is there a word α in L such that, for some $w_1, \ldots, w_l \in \mathbf{N}$, and for some $\hat{W}_1, \ldots, \hat{W}_k$,

$$\hat{W}_1 \in W_1(\alpha) \wedge \ldots \wedge \hat{W}_k \in W_k(\alpha) \wedge P(w_1, \ldots, w_l, \hat{W}_1, \ldots, \hat{W}_k) \qquad (8)$$

holds?

Notice that the lower weight bounds $q_1, \ldots, q_l$ are in the **Given**-part, hence they are constants; while the upper bounds $w_1, \ldots, w_l$ in the **Question**-part, are essentially unspecified parameters. (Otherwise, if the lower bounds $q_1, \ldots, q_l$ are moved into the **Question**-part; i.e., both the lower and the upper bounds are parameterized constants, then the k-accumulated weight problem is a special case of the loose k-accumulated weight problem under this definition, by letting the lower bound and the upper bound be the same parameterized constant for each activity.)

The following result shows that the loose k-accumulated weight problem is decidable for each k. It is in contrast to Theorem 3 that the k-accumulated weight problem is undecidable for some large k.

Theorem 6. *For each k, the loose k-accumulated weight problem is decidable.*

4 Applications

In this section, we will apply the results presented in the previous section to some verification problems concerning infinite systems containing parameterized constants. We start with a general definition.

A transition system M can be described as a relation $T \subseteq S \times \Gamma^* \times \Sigma \times S \times \Gamma^*$, where S is a finite set of states, Γ is the configuration alphabet, and Σ is the activity alphabet. Obviously, we always assume that M can be effectively described; i.e., T is recursive. A configuration $\langle s, \beta \rangle$ of M is a pair of a state s in S and a word β in Γ^*. In the description of M, an initial configuration is also designated. According to the definition of T, an activity in Σ transforms one configuration to another. More precisely, we write $\langle s, \beta \rangle \xrightarrow{a} \langle s', \beta' \rangle$ if $T(s, \beta, a, s', \beta')$. Let $\alpha \in \Sigma^*$ with $\alpha = a^1 \ldots a^m$ for some m. We say that $\langle s, \beta, \alpha \rangle$ is *reachable* if, for some configurations $\langle s_0, \beta_0 \rangle, \ldots, \langle s_m, \beta_m \rangle$, the following is satisfied

$$\langle s_0, \beta_0 \rangle \xrightarrow{a^1} \ldots \xrightarrow{a^m} \langle s_m, \beta_m \rangle, \tag{9}$$

where $\langle s_0, \beta_0 \rangle$ is the initial configuration, $s_m = s$ and $\beta_m = \beta$. We use L_s to denote the set $\{(\beta, \alpha) : \langle s, \beta, \alpha \rangle \text{ is reachable}\}$. M is a *semilinear system* if L_s is an effectively semilinear language for each $s \in S$ (i.e., the semilinear set of L_s is computable from the description of M). As before, we use $w_1, \ldots, w_l$ to denote a weight measure of $\Sigma = \{a_1, \ldots, a_l\}$, and use $\Sigma_1, \ldots, \Sigma_k$ to denote k subsets of Σ. We may introduce *weight counters* $W_1, \ldots, W_k$ into M to indicate that the accumulated weight on each Σ_i is incremented by w_i whenever an activity $a_j \in \Sigma_i$ is performed. That is, on a transition $\langle s, \beta \rangle \xrightarrow{a_j} \langle s', \beta' \rangle$ in M, the counters are updated as follows, for each $1 \le i \le k$, if $a_j \in \Sigma_i$ then $W_i := W_i + w_j$ else $W_i := W_i$. Similarly, for a loose weight measure $(q_1, w_1), \ldots, (q_l, w_l)$, the counters are updated on the transition as follows: for each $1 \le i \le k$, if $a_j \in \Sigma_i$ then $W_i := W_i + p_j$ else $W_i := W_i$, for some $q_j \le p_j \le w_j$ (i.e., p_j is nondeterministically chosen between q_j and w_j). Starting with 0, the weight counters are updated along an execution path in (9). We say that $\langle s, \beta, \alpha, W_1, \ldots, W_k \rangle$ is reachable (under the weight measure $w_1, \ldots, w_l$) if the weight counters have values $W_1, \ldots, W_k$ at the end of an execution path in (9) witnessing that $\langle s, \beta, \alpha \rangle$ is reachable.

Let $y_1, \ldots, y_u$ and $z_1, \ldots, z_v$ be distinct variables. A (u, v)-formula, denoted by $P([y_1, \ldots, y_u]; [z_1, \ldots, z_v])$, is a Presburger formula that is a Boolean combination (using $\wedge$ and $\neg$) of Presburger formulas over $y_1, \ldots, y_u$ and Presburger formulas over $z_1, \ldots, z_v$. For the M specified in above, we let $u = |\Gamma| + l$ and $v = l + k$. Now, we consider the *k-reachability problem* for M: given a state s and a (u, v)-formula P, are there $w_1, \ldots, w_l \in \mathbf{N}$ such that

$$P([\phi(\alpha), \phi(\beta)]; [w_1, \ldots, w_l, W_1, \ldots, W_k]) \tag{10}$$

holds for some reachable $\langle s, \beta, \alpha, W_1, \ldots, W_k \rangle$ (under the weight measure $w_1, \ldots, w_l$)? The *loose k-reachability problem* for M can be defined similarly where the lower weights $q_1, \ldots, q_l$ are given. Directly from Theorems 4, 5 and 6, one can show the following results.

Theorem 7. *The 2-reachability problem is decidable for semilinear systems.*

Theorem 8. *For each k, the k-reachability problem is decidable for semilinear systems, when P in (10) is a disjunction of formulas in the following form:*

$$Q([\phi(\alpha), \phi(\beta)]; [w_1, \ldots, w_l]) \wedge c_1 W_1 + \ldots + c_k W_k + d_1 w_1 + \ldots + d_l w_l \sim c_0,$$

where Q is a (u, l)-formula, the c's and d's are integers, and $\sim \in \{=, \neq, >, <, \geq, \leq\}$.

Theorem 9. *For each k, the loose k-reachability problem for semilinear systems is decidable.*

Many machine models are semilinear systems. We start with a simple model. Consider a nondeterministic finite state machine M, which is specified in Section 1 with a designated initial state. Notice that, in this case, $\Gamma = \emptyset$. Let s be a state. Clearly, L_s, the set of all the activity sequences when M moves from the initial state to s is a regular (and hence semilinear) language. Therefore, Theorems 7 and 8 hold for such M. Conversely, for any semilinear language L, one can construct, from the semilinear set of L, a regular language whose semilinear set is the same as the semilinear set of L [18]. From the regular language, one can easily construct a M and a state s such that the regular language is exactly L_s. It is routine to establish the fact that the k-reachability problem is decidable (for the M) iff the k-accumulated weight problem is decidable (for the L). From Theorem 3, one can show

Theorem 10. *There is a fixed k such that the k-reachability problem is undecidable for finite state machines M.*

In the definition of the k-reachability problem, the Presburger formula P in (10) is to specify the undesired values for the w's and the W's. When M is understood as a design of some system, a positive answer to the instance of the k-reachability problem indicates a design bug. In software engineering, it is highly desirable that a design bug is found as early as possible, since it is very costly to fix a bug once a system has already been implemented. It is noticed that in a specific implementation of the design, the parameterized constants are concrete, though the values differ from one implementation to another. Of course, one may test the specification by plugging in a particular choice for the concrete values. However, it is important to guarantee that for *any* concrete values for the parameterized constants, the design M is bug-free.

For instance, consider again the packet-based network switch example, where as we mentioned in Section 1, the switch is modeled as a finite state machine. Suppose the scheduling discipline is required to achieve such fairness property that no matter how the weights are assigned, the total packets serviced by o_1 must be greater than that of o_2 only if the summation of weights of connections in C_1 is greater than that of C_2 (we assume that all connections are nonempty at any time); i.e.,

$$\sum_{c_i \in C_1} w_i - \sum_{c_i \in C_2} w_i \geq 0 \rightarrow W_1 - W_2 \geq 0.$$

From Theorem 7, we know this fairness property can be automatically verified. When there are k servers involved in the example switch, a fairness property can be similarly formulated as a conjunction of the fairness between any two servers. In this case, the fairness property over k-servers is hard to be automatically verified, because of Theorem 10.

One may consider other variations on the model of M. For instance, an activity a_i is associated with, instead of one parameterized weight w_i, but two (or any fixed number of) parameterized weights w_i^1 and w_i^2, from which an instance of the activity can nondeterministically choose during execution. But this variation does not increase the expressive power of M, since "performing activity a_i" can be simulated by "performing activity a_i^1" or "performing activity a_i^2" (nondeterministically chosen) where activity a_i^1 (resp. a_i^2) has weight w_i^1 (resp. w_i^2). One may consider another variation on the model of M where an instance of activity a_i has a weight nondeterministically chosen in between some given number (such as 2) in $\mathbf{N}$ and a parameterized constant w_i. Clearly, from Theorem 9, the loose k-reachability problem is decidable for this model of M.

M can be further generalized; e.g., M is augmented with a pushdown stack. Each transition in M now is in the following form: $s \rightarrow^{a,b,\gamma} s'$, indicating that M moves from state s to state s' while performing an activity a and also updating the stack (replacing the top symbol b in the stack by a stack word γ). There are only finitely many transitions in the description of M. Initially, the stack contains a designated initial symbol (i.e., an *initialized stack*) and the machines stays at an initial state. Notice that, for this model of M, L_s is a permutation of a context-free (hence semilinear) language. Therefore, M is still a semilinear system. The results of Theorems 7, 8 and 9 apply for pushdown systems.

M can be further augmented with a finite number of reversal-bounded counters. A nonnegative integer counter is reversal-bounded [11] if it alternates between a nondecreasing mode and a nonincreasing mode (and vice versa) for a given finite number of times, independent of the computations. Hence, a transition in M, in addition to the stack operation, can increment/decrement a counter by one and test a counter against zero. When the counter values are encoded as unary strings, it is not hard to show that the language of L_s is a semilinear language [11]. Hence, this model of M is still a semilinear system, and hence, Theorems 7, 8 and 9 can be applied.

M can be further generalized by adding a number of dense clocks. A clock is a nonnegative real variable. Clock behavior in M includes progresses and resets. A clock progress makes all the clocks advance with the same rate for a nondeterministically chosen amount in positive reals. A clock reset brings a clock value to 0 while keeping all the other clocks unchanged. In M, a transition is either a *stay transition* or a *reset transition*. A stay transition makes M stay in the current state and not perform any stack and counter operations, but the clocks progress. A reset transition makes M move from a state to another while performing an activity, a stack operation, and/or a counter operation. In addition, the transition resets some clocks. A *clock constraint* is a Boolean combination of formulas $x \sim c$ and $x - y \sim c$ where x, y are clocks, and c is an integer, $\sim \in \{>, <, =, \geq, \leq\}$. A (stay/reset) transition in M is also associated with a clock constraint that must be satisfied in order for the transition to fire. The reader may have already noticed that, when M does not have reversal-bounded counters and the pushdown stack, and when each activity is understood as an "input symbol", M is simply equivalent to a timed automaton [2] that has been well studied in recent years for modeling and verifying real-time systems (see [1] for a survey). Here in this paper, an activity on a transition in M is associated with a weight. This weight can be understood as a special form of "prices" in the sense of [15] that tries to model some (e.g., linearly) time-dependent variables in a complex real-time systems. Though, in general, priced timed automata are undecidable for reachability [15], some restricted forms of prices should be decidable, as shown in below, when one understands a weight as a special form of prices.

Consider an execution of M that starts from the initial state and ends with state s. Initially, all the clocks and counters are 0 and the stack is initialized. At the end of the execution, we require that the clock values $(x_1, \ldots, x_t)$, the counter values $(y_1, \ldots, y_u)$, and the stack content (γ) satisfy a given formula $Q(x_1, \ldots, x_t, y_1, \ldots, y_u, z_1, \ldots, z_m)$ where z_i is the number of occurrences of stack symbol b_i in stack word γ. The form of the formula Q is a Boolean combination of $l(x_1, \ldots, x_t, y_1, \ldots, y_u, z_1, \ldots, z_m) \sim 0$ where l is a linear polynomial and $\sim \in \{>, <, =, \geq, \leq\}$. Notice that Q contains both dense variables and discrete variables. Here, we use L to denote the set of all activity sequences on all such executions. If M does not have counters and the stack, L is a regular language and Q is a clock constraint (i.e., as we defined earlier, comparing one clock or the difference of two clocks against a constant). The regularity can be shown using the classic region technique in [2]. In general, however, L is not regular. Using the main theorem in [9], one can show that L can be accepted by a nondeterministic pushdown automaton with reversal-bounded counters. Hence, L is still a semilinear language according to [11]. Associating an activity with a parameterized constant, one can formulate a k-reachability problem for M (similar to (10)): Is there an execution of M from the initial state to state s such that, at the end of the execution, the parameterized constants $w_1, \ldots, w_l$, the accumulated weights $W_1, \ldots, W_k$, the clocks values $x_1, \ldots, x_t$, the counter values $y_1, \ldots, y_u$, and the stack word counts $z_1, \ldots, z_m$, satisfy

$$P(w_1, \ldots, w_l, W_1, \ldots, W_k) \wedge Q(x_1, \ldots, x_t, y_1, \ldots, y_u, z_1, \ldots, z_m)?$$

Following the same proof ideas, one can show that the results of Theorems 7, 8 and 9 still hold for the M augmented with dense clocks, a pushdown stack and reversal-bounded counters.

As a final example, we use the decidability of 2-systems to strengthen recent results in [12]. Consider the model of a two-way deterministic finite automaton augmented with monotonic (i.e., nondecreasing) counters $C_1, ..., C_k$ operating on an input of the form $a_1^{i_1}...a_n^{i_n}$ (for some fixed n), with left and right endmarkers. M starts in its initial state on the left end of the input with all counters initially zero. At each step, a counter can be incremented by 0 or 1, but the counters do not participate in the dynamics of the machine. An *m-equality relation E* over the counter values is a conjunction of m atomic relations of the form $c_i = c_j$. The *m-equality relation problem* is that of deciding, given a machine M, a state q, and an m-equality relation E, whether there is $(i_1, \ldots, i_n)$ such that M, on input $a_1^{i_1} \ldots a_n^{i_n}$, reaches some configuration where the state is q and the counter values satisfy E. Note that in dealing with the m-equality relation problem, we need only consider machines with at most $2m$ monotonic counters. It is open whether the m-equality relation problem is decidable. However, when $m = 1$, it was recently shown in [12] that the 1-equality relation problem is decidable. The proof of the decidability for $m = 1$ in [12] does not generalize to the case when the two counter values must satisfy an arbitrary Presburger formula E. We give a proof of this generalization below.

First we generalize the m-equality relation problem by allowing E to be an arbitrary Presburger relation $E(c_1, ..., c_k)$ over the counter values $c_1, ..., c_k$. Call this the *Presburger relation problem*. Note that the m-equality relation problem is a very special case of the Presburger relation problem. We can use the decidability of 2-systems to show that the Presburger relation problem for machines with only 2 monotonic counters is decidable. The idea is as follows. In [12], it was shown that the values c_1 and c_2 of the two counters at any time can effectively be represented by equations of the form:

$$c_1 = A_1 + yB_1 + C_1, c_2 = A_2 + yB_2 + C_2,$$

where y is a nonnegative integer variable, and $A_1, B_1, C_1, A_2, B_2, C_2$ are nonnegative linear polynomials in some nonnegative integer variables $x_1, ..., x_m$. (Even though C_1 and C_2 can be absorbed by A_1 and A_2, we use the formulation above to be consistent with the formulation in [12].) Since E (subset of $\mathbf{N}^2$) is Presburger, it is semilinear. First assume that E is a linear set. Then the two components of E can be represented by nonnegative linear polynomials $p_1(z_1, ..., z_r)$ and $p_2(z_1, ..., z_r)$ for some nonnegative integer variables $z_1, ..., z_r$. Thus, using the two equations above, we get: $A_1 + yB_1 + C_1 = p_1(z_1, ..., z_r)$, $A_2 + yB_2 + C_2 = p_2(z_2, ..., z_r)$. Rearranging terms, these two equations can be written as: $yB_1 = p_1 - A_1 - C_1$ and $yB_2 = p_2 - A_2 - C_2$. By semilinear transformation, we can reduce these equations to $yB_1 = D_1$ and $yB_2 = D_2$, where B_1, B_2, D_1, D_2 are nonnegative linear polynomials in some nonnegative integer variables $w_1, ..., w_t$. Since the above equations constitute a 2-system, it is solvable in $y, w_1, ..., w_t$. When E is a semilinear set, we just need to check if at least one of a finite number of equations of the form above has a solution.

It is open whether the Presburger relation problem is decidable when there are more than 2 monotonic counters (since the m-equality relation problem, which is a special case, is open). But suppose the Presburger relation E takes the following special form: $p_1(c_1, ..., c_k) \sim d_1 \wedge p_2(c_1, ..., c_k) \sim d_2 \wedge ... \wedge p_m(c_1, ..., c_k) \sim d_m$, where $d_1, ..., d_m$ are integers (positive, negative, zero) and each $p_i(c_1, ..., c_k)$ is a linear polynomial (not necessarily nonnegative), and each $\sim$ in $\{>, <, =, \geq, \leq\}$. It is easy to see that when $m = 2$, i.e., there are only two linear polynomials p_1 and p_2 involved in the conjunction

above, then by adding "slack" variables and doing semilinear transformation, we can again reduce the problem to solving a system of the form: $yB_1 = D_1, yB_2 = D_2$, and, therefore, solvable. However, the case when $m > 2$ is open.
Acknowledgement. The authors would like to thank the anonymous referees for many valuable comments and suggestions.

References

1. R. Alur. Timed automata. In *CAV'99*, volume 1633 of *LNCS*, pages 8–22. Springer, 1999.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
3. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model-checking. In *CONCUR'97*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.
4. T. Bultan, R. Gerber, and W. Pugh. Model-checking concurrent systems with unbounded integer variables: symbolic representations, approximations, and experimental results. *ACM Transactions on Programming Languages and Systems*, 21(4):747–789, July 1999.
5. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
6. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
7. Z. Dang. Verifying and debugging real-time infinite state systems (PhD. Dissertation). *Department of Computer Science, University of California at Santa Barbara*, 2000.
8. Z. Dang, O. Ibarra, and Z. Sun. On the emptiness problems for two-way nondeterministic finite automata with one reversal-bounded counter. In *ISAAC'02*, volume 2518 of *LNCS*, pages 103–114. Springer, 2002.
9. Zhe Dang. Binary reachability analysis of pushdown timed automata with dense clocks. In *CAV'01*, volume 2102 of *LNCS*, pages 506–517. Springer, 2001.
10. G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997. Special Issue: Formal Methods in Software Practice.
11. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
12. O. H. Ibarra and Z. Dang. Deterministic two-way finite automata augmented with monotonic counters. 2002 (submitted).
13. K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.
14. O. Kupferman and M.Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *CAV'00*, volume 1855 of *LNCS*, pages 36–52. Springer, 2000.
15. K. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV'01*, volume 2102 of *LNCS*, pages 493–505. Springer, 2001.
16. Y. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.
17. 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.
18. R. Parikh. On context-free languages. *Journal of the ACM*, 13:570–581, 1966.
19. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344. IEEE Computer Society Press, 1986.