Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 299 (2003) 413–438

Theoretical Computer Science

www.elsevier.com/locate/tcs

Presburger liveness verification of discrete timed automata☆

Zhe Dang[a,*], Pierluigi San Pietro[b], Richard A. Kemmerer[c]

[a] *School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164, USA*

[b] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*

[c] *Department of Computer Science, University of California at Santa Barbara, CA 93106, USA*

Received 4 May 2001; received in revised form 11 February 2002; accepted 10 April 2002
Communicated by O.H. Ibarra

Abstract

Using an automata-theoretic approach, we investigate the decidability of liveness properties (called *Presburger liveness properties*) for timed automata when Presburger formulas on configurations are allowed. While the general problem of checking a temporal logic such as TPTL augmented with Presburger clock constraints is undecidable, we show that there are various classes of Presburger liveness properties which are decidable for *discrete* timed automata. For instance, it is decidable, given a discrete timed automaton $\mathscr{A}$ and a Presburger property P, whether there exists an ω-path of $\mathscr{A}$ where P holds infinitely often. We also show that other classes of Presburger liveness properties are indeed undecidable for discrete timed automata, e.g., whether P holds infinitely often *for each* ω-path of $\mathscr{A}$. These results might give insights into the corresponding problems for timed automata over dense domains, and help in the definition of a fragment of linear temporal logic, augmented with Presburger conditions on configurations, which is decidable for model checking timed automata.

Keywords: Model-checking; Timed automata; Temporal logic; Liveness

☆ A preliminary version [13] of this paper appeared in the *Proceedings* of the 18th International Symposium on Theoretical Aspects of Computer Science (STACS 2001), Lecture Notes in Computer Science 2010, Springer-Verlag, 2001, pp. 132–143.

* Corresponding author.

E-mail addresses: zdang@eecs.wsu.edu (Zhe Dang), sanpietr@elet.polimi.it (P.S. Pietro), kemm@cs.ucsb.edu (R.A. Kemmerer).

0304-3975/03/$ - see front matter
PII: S0304-3975(02)00485-1

1. Introduction

Timed automata [3] are widely regarded as a standard model for real-time systems, because of their ability to express quantitative time requirements. A timed automaton can be considered as a finite automaton augmented with a number of clocks. Clocks can either progress (synchronously) at rate 1, or some of them may be reset to 0 when a transition is fired. Transitions may fire when their enabling conditions hold. In particular, the enabling condition of a transition is in the form of clock *regions*: a clock, or the difference of two clocks, is compared against an integer constant, e.g., $x - y > 5$, where x and y are clocks.

A fundamental result in the theory of timed automata is that region reachability is decidable. This has been proved by using the region technique [3]. This result is very useful since in principle it allows some forms of automatic verification of timed automata. In particular, it helps in developing a number of temporal logics [2–6,17,20,22,24], in investigating the model-checking problem and in building model-checking tools [18,25,21] (see [1,26] for surveys).

In real-world applications [8], clock constraints represented as clock regions are useful but often not powerful enough. For instance, we might want to argue whether a non-region property such as $x_1 - x_2 > x_3 - x_4$ (i.e., the difference of clocks x_1 and x_2 is larger than that of x_3 and x_4) always holds when a timed automaton starts from clock values satisfying another non-region property. Therefore, it would be useful to consider Presburger formulas as clock constraints, such as the example given in Section 5. Even though a temporal logic like TPTL [6] is undecidable when augmented with Presburger clock constraints [6], recent work [10–12] has found decidable characterizations of the binary reachability of timed automata, giving hope that *some* important classes of non-region temporal properties are still decidable for timed automata.

In this paper, we look at *discrete* timed automata (*dta*), i.e., timed automata where clocks are integer-valued. Discrete time makes it possible to apply, as underlying theoretical tools, a good number of automata-theoretic techniques and results. In addition to the fact that discrete clocks are usually easier for practitioners to handle than dense clocks and that *dta*s are useful by themselves as a model of real-time systems [5], results on *dta*s may give insights into corresponding properties of dense timed automata [16].

The study of *safety* properties and *liveness* properties is of course of the utmost importance for real-life applications [7]. In [10,12], it has been shown that the Presburger safety analysis problem is decidable for discrete timed automata. That is, it is decidable whether, given a discrete timed automaton $\mathscr{A}$ and two sets I and P of configurations of $\mathscr{A}$ (tuples of control state and clock values) definable by Presburger formulas, $\mathscr{A}$ always reaches a configuration in P when starting from a configuration in I.

In this paper we concentrate on the Presburger liveness problem, by systematically formulating a number of Presburger liveness properties and investigating their decidability. For instance, we consider the $\exists$-Presburger-i.o. problem: whether, given two Presburger sets P and I as above, there exists an ω-path p for $\mathscr{A}$ such that p starts from I and P is satisfied on p infinitely often. Another example is the $\forall$-Presburger-

eventual problem: whether for all ω-paths p that start from I, P is eventually satisfied on p.

The main results of this paper show that (using an obvious notation, once it is clear that $\exists$ and $\forall$ are path quantifiers):

- The $\exists$-Presburger-i.o. problem and the $\exists$-Presburger-eventual problem are both decidable. So are their duals, the $\forall$-Presburger-almost-always problem and the $\forall$-Presburger-always problem.
- The $\forall$-Presburger-i.o. problem and the $\forall$-Presburger-eventual problem are both undecidable. So are their duals, the $\exists$-Presburger-almost-always problem and the $\exists$-Presburger-always problem.

These results can be helpful in formulating a weak form of a Presburger linear temporal logic and in defining a fragment thereof that is decidable for model-checking *dta*s. The proofs are based on the definition of a version of *dta*s, called *static dta*s, which do not have enabling conditions on transitions. The approach used is to first translate the problem for generalized *dta*s to an equivalent problem for static *dta*s. The results are then proved for the static *dta*s. Static *dta*s are much simpler to deal with than *dta*s; therefore, the proofs are easier.

This paper is organized as follows. Section 2 introduces the main definitions, such as discrete timed automata and the Presburger liveness properties. Section 3 shows the decidability of the $\exists$-Presburger-i.o. problem and the $\exists$-Presburger-eventual problem. Section 4 shows the undecidability of the $\forall$-Presburger-i.o. problem and the $\forall$-Presburger-eventual problem. Section 5 discusses some aspects related to the introduction of Presburger conditions in temporal logic and to the extension of our results to dense time domains.

2. Preliminaries

A timed automaton [3] is a finite state machine augmented with a number of clocks. All the clocks progress synchronously with rate 1, except when a clock is reset to 0 at some transition. In this paper, we consider integer-valued clocks. A *clock constraint* (or a *region*) is a Boolean combination of *atomic clock constraints* in the following form:

$$x\#c, x - y\#c,$$

where # denotes $\leqslant, \geqslant, <, >$, or $=$, c is an integer, and x, y are integer-valued clocks. Let $\mathscr{L}_X$ be the set of all clock constraints on clocks X. Let $\mathbf{N}$ be the set of nonnegative integers.

Definition 1. A *discrete timed automaton* (*dta*) is a tuple

$$\mathscr{A} = \langle S, X, E \rangle$$

where S is a finite set of (*control*) *states*, X is a finite set of *clocks* with values in $\mathbf{N}$, and $E \subseteq S \times 2^X \times \mathscr{L}_X \times S$ is a finite set of *edges* or *transitions*.

Each edge $\langle s, \lambda, l, s' \rangle$ denotes a transition from state s to state s' with *enabling condition* $l \in \mathscr{L}_X$ and a set of clock resets $\lambda \subseteq X$. Note that λ may be empty: in this case, the edge is called a *clock progress transition*. Also note that since a state may be connected to more than one state through multiple edges with the same enabling condition, $\mathscr{A}$ is, in general, nondeterministic.

The semantics of *dta*s is defined as follows. We use $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{X}, \boldsymbol{Y}$ to denote clock vectors (i.e., vectors of clock values) with $\boldsymbol{V}_x$ being the value of clock x in $\boldsymbol{V}$. $\mathbf{1}$ denotes the identity vector in $\mathbf{N}^{|X|}$; i.e., $\mathbf{1}_x = 1$ for each $x \in X$.

Definition 2. A *configuration* $\langle s, \boldsymbol{V} \rangle \in S \times \mathbf{N}^{|X|}$ is a pair of a control state s and a clock vector $\boldsymbol{V}$.

$$\langle s, \boldsymbol{V} \rangle \rightarrow^{\mathscr{A}} \langle s', \boldsymbol{V}' \rangle$$

denotes a *one-step transition* from configuration $\langle s, \boldsymbol{V} \rangle$ to configuration $\langle s', \boldsymbol{V}' \rangle$ satisfying all the following conditions:

- There is an edge $\langle s, \lambda, l, s' \rangle$ in $\mathscr{A}$ connecting state s to state s',
- The enabling condition of the edge is satisfied; i.e., $l(\boldsymbol{V})$ is true,
- Each clock changes according to the edge. If there are no clock resets on the edge, i.e., $\lambda = \emptyset$, then clocks progress by one time unit, i.e., $\boldsymbol{V}' = \boldsymbol{V} + \mathbf{1}$. If $\lambda \neq \emptyset$, then for each $x \in \lambda$, $\boldsymbol{V}'_x = 0$ while for each $x \notin \lambda$, $\boldsymbol{V}'_x = \boldsymbol{V}_x$.

A configuration $\langle s, \boldsymbol{V} \rangle$ is a *deadlock configuration* if there is no configuration $\langle s', \boldsymbol{V}' \rangle$ such that $\langle s, \boldsymbol{V} \rangle \rightarrow^{\mathscr{A}} \langle s', \boldsymbol{V}' \rangle$. $\mathscr{A}$ is *total* if every configuration is not a deadlock configuration. A *path* is a finite sequence

$$\langle s_0, \boldsymbol{V}^0 \rangle \cdots \langle s_k, \boldsymbol{V}^k \rangle$$

for some $k \geqslant 1$ such that $\langle s_i, \boldsymbol{V}^i \rangle \rightarrow^{\mathscr{A}} \langle s_{i+1}, \boldsymbol{V}^{i+1} \rangle$ for each $0 \leqslant i \leqslant k-1$. A path is a *progress* path if there is at least one clock progress transition, i.e., with $\lambda = \emptyset$, on the path. An *ω-path* is an infinite sequence

$$\langle s_0, \boldsymbol{V}^0 \rangle \cdots \langle s_k, \boldsymbol{V}^k \rangle \cdots$$

such that each prefix $\langle s_0, \boldsymbol{V}^0 \rangle \cdots \langle s_k, \boldsymbol{V}^k \rangle$ is a path. An ω-path is *divergent* if there is an infinite number of clock progress transitions on the ω-path. It is noticed that an "invariant" associated with a control state in a standard discrete timed automaton [3] can be modeled with a self-looping transition in our model [12]. Since we focus on the clock behaviors of a *dta*, instead of the ω-language accepted by it, input symbols (or event labels) [3] are not considered in our definition. The input to a timed automaton is always one-way. Thus, input symbols can always be built into control states.

Fig. 1 shows a simple discrete timed automaton with one clock x. It models a system with two states *down* and *up*, in which each down time is bounded by 4 and the system has been up for at least 100 time units before the next down time. The following sequence of configurations is a path: $\langle down, x = 0 \rangle, \langle down, x = 1 \rangle, \langle up, x = 0 \rangle, \langle up, x = 1 \rangle, \langle up, x = 2 \rangle, \ldots, \langle up, x = 150 \rangle, \langle down, x = 0 \rangle$.

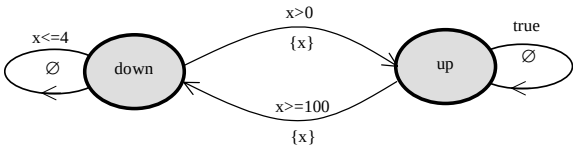

Fig. 1. An example of a discrete timed automaton.

Let Y be a finite set of variables over integers. For all integers a_y, with $y \in Y$, b and c (with $b > 0$),

$$\sum_{y \in Y} a_y y < c$$

is an *atomic linear relation* on Y and

$$\sum_{y \in Y} a_y y \equiv_b c$$

is a *linear congruence* on Y. A *linear relation* on Y is a Boolean combination (using $\neg$ and $\wedge$) of atomic linear relations on Y. A *Presburger formula* on Y is the Boolean combination of atomic linear relations on Y and linear congruences on Y. A set P is *Presburger-definable* if there exists a Presburger formula $\mathscr{F}$ on Y such that P is exactly the set of the solutions for Y that make $\mathscr{F}$ true. Since Presburger formulas are closed under quantification, we will allow quantifiers over integer variables.

Write

$$\langle s, \boldsymbol{V} \rangle \rightsquigarrow^{\mathscr{A}} \langle s', \boldsymbol{V}' \rangle$$

if $\langle s, \boldsymbol{V} \rangle$ reaches $\langle s', \boldsymbol{V}' \rangle$ through a path in $\mathscr{A}$. The binary relation $\rightsquigarrow^{\mathscr{A}}$ can be considered as a subset of configuration pairs and called *binary reachability*. It has recently been shown that,

Theorem 1. *The binary reachability* $\rightsquigarrow^{\mathscr{A}}$ *is Presburger-definable* [10,12].

The *Presburger safety analysis problem* is to decide whether $\mathscr{A}$ can only reach configurations in P starting from any configuration in I, given two Presburger-definable sets I and P of configurations. For example, an instance of this problem is:

"Given a discrete timed automaton $\mathscr{A}$ and a state s_0, starting from any configuration $\langle s, \boldsymbol{V} \rangle$ with $\boldsymbol{V}_{x_1} + \boldsymbol{V}_{x_2} - \boldsymbol{V}_{x_3} < 5 \wedge s = s_0$, $\mathscr{A}$ can only reach configurations $\langle s', \boldsymbol{V}' \rangle$ with $\boldsymbol{V}'_{x_1} - 2\boldsymbol{V}'_{x_2} > 1$."

Because of Theorem 1, the Presburger safety analysis problem is decidable for *dta*s.

In this paper, we consider *Presburger liveness analysis problems* for *dta*s, obtained by combining a path-quantifier with various modalities of satisfaction on an ω-path. Let I and P be two Presburger-definable sets of configurations, and let p be an ω-

path $\langle s_0, \boldsymbol{V}^0\rangle, \langle s_1, \boldsymbol{V}^1\rangle \ldots$. Define the following modalities of satisfactions of P and I over p:

- p is *P-i.o.* if P is satisfied infinitely often on the ω-path, i.e., there are infinitely many k such that $\langle s_k, \boldsymbol{V}^k\rangle \in P$.
- p is *P-always* if for each k, $\langle s_k, \boldsymbol{V}^k\rangle \in P$.
- p is *P-eventual* if there exists k such that $\langle s_k, \boldsymbol{V}^k\rangle \in P$.
- p is *P-almost-always* if there exists k such that for all $k' > k$, $\langle s_{k'}, \boldsymbol{V}^{k'}\rangle \in P$.
- p *starts* from I if $\langle s_0, \boldsymbol{V}^0\rangle \in I$.

Definition 3. Let $\mathscr{A}$ be a *dta* and let I and P be two Presburger-definable sets of configurations of $\mathscr{A}$. The *∃-Presburger-i.o.* (*resp. always*, *eventual and almost-always*) *problem* is to decide whether the following statement holds:

> there is an ω-path p starting from I that is P-i.o. (resp. P-always, P-eventual and P-almost-always).

The *∀-Presburger-i.o.* (*resp. always*, *eventual and almost-always*) *problem* is to decide whether the following statement holds:

> for every ω-path p, if p starts from I, then p is P-i.o. (resp. always, eventual and almost-always).

3. Decidability results

In this section, we show that the ∃-Presburger-i.o. problem is decidable for *dta*s. Proofs of an infinitely-often property usually involve analysis of cycles in the transition system. However, for *dta*s this is difficult because of the enabling conditions on edges. The difficulty is that because the clock values may be changed in a cycle, the enabling condition on the edge that starts the cycle may no longer be satisfied. Therefore, the cycle cannot be repeated. Our approach is to introduce *static* discrete timed automata (i.e., *dta*s with all the enabling conditions being simply *true*). Since the enabling condition for every cycle is now *true*, every cycle can be repeated. We first show that an ∃-Presburger-i.o. problem for a *dta* can be translated into an ∃-Presburger-i.o. problem for a static *dta*. Then, we prove that the ∃-Presburger-i.o. problem is decidable for static *dta*s.

3.1. A translation from dtas to static dtas

In this subsection, we use a technique modified from [12] to show that the tests in a *dta* $\mathscr{A}$ can be eliminated. That is, $\mathscr{A}$ can be effectively transformed into a static *dta* $\mathscr{A}''$ where all the tests are simply *true* and $\mathscr{A}''$ has (almost) the same static transition graph as $\mathscr{A}$. This is based on an encoding of the tests of $\mathscr{A}$ into the finite state control of $\mathscr{A}''$.

Let $X=\{x_1,\ldots,x_k\}$ be the clocks in $\mathscr{A}$ and $\boldsymbol{V}$ be the clock vector that $\mathscr{A}$ starts with. *Tests*, which are the enabling conditions on edges, are Boolean combinations of $x_i\#c$, $x_i-x_j\#c$ with c an integer. Here we demonstrate a technique to eliminate these tests, based on the definition of a finite table lookup $a_{ij}\#c$ and $b_i\#c$ to replace tests $x_i-x_j\#c$ and $x_i\#c$. Let m be one plus the maximal absolute value of all the integer constants that appear in the tests in $\mathscr{A}$. Denote the finite set $[m]=_{def}\{-m,\ldots,0,\ldots,m\}$. We have entries a_{ij} and b_i for $1\leqslant i,\ j\leqslant k$. Each entry can be regarded as a finite state variable with states in $[m]$. Intuitively, a_{ij} is used to record the difference between two clock values of x_i and x_j, and b_i is used to record the clock value of x_i. During the computation of $\mathscr{A}$, when the difference x_i-x_j (or the value x_i) goes beyond m or below $-m$, a_{ij} (or b_i) keeps the value m or $-m$, respectively.

The procedure for updating the entries is given below, in which "$\oplus 1$" means adding one if the result does not exceed m, otherwise it keeps the same value. "$\ominus 1$" means subtracting one if the result is not less than $-m$, otherwise it keeps the same value. We modify $\mathscr{A}$ as follows. Let e be an edge of $\mathscr{A}$. If the set of clock resets on e is $\lambda=\emptyset$, then the entries are updated by adding the following instructions to e, for each $1\leqslant i\leqslant k$:

- $a_{ij}:=a_{ij}$ for each $1\leqslant j\leqslant k$. Recall that all the clocks progress after this edge, thus the difference between any two clocks is unchanged.
- $b_i:=b_i\oplus 1$. That is, clocks progress by one time unit.

If the set of clock resets is $\lambda\neq\emptyset$, the entries are updated by adding the following instructions to e, for each $1\leqslant i,j\leqslant k$:

- $a_{ij}:=0$ if $i\in\lambda$ and $j\in\lambda$. In this case, both the clocks x_i and x_j reset to 0.
- $a_{ij}:=-b_j$ if $i\in\lambda$ and $j\notin\lambda$. In this case, x_i resets but x_j does not. So the difference should be $-x_j$.
- $a_{ij}:=b_i$ if $i\notin\lambda$ and $j\in\lambda$.
- $a_{ij}:=a_{ij}$ if $i\notin\lambda$ and $j\notin\lambda$

followed by adding the following instructions:

- $b_i:=b_i$ if $x_i\notin\lambda$.
- $b_i:=0$ if $x_i\in\lambda$.

The initial values of a_{ij} and b_i can be constructed directly from the values $\boldsymbol{V}_{x_i}$ of clocks x_i, for each $1\leqslant i,\ j\leqslant k$:

- $a_{ij}:=\boldsymbol{V}_{x_i}-\boldsymbol{V}_{x_j}$ if $|\boldsymbol{V}_{x_i}-\boldsymbol{V}_{x_j}|\leqslant m$,
- $a_{ij}:=m$ if $\boldsymbol{V}_{x_i}-\boldsymbol{V}_{x_j}>m$,
- $a_{ij}:=-m$ if $\boldsymbol{V}_{x_i}-\boldsymbol{V}_{x_j}<-m$,

and, noticing that clocks are nonnegative,

- $b_i:=\boldsymbol{V}_{x_i}$ if $\boldsymbol{V}_{x_i}\leqslant m$,
- $b_i:=m$ if $\boldsymbol{V}_{x_i}>m$.

The correctness of using $a_{ij}\#c$ for a test $x_i-x_j\#c$ and using $b_i\#c$ for $x_i\#c$ is shown by the following statement: by adding the above entry updating instructions to $\mathscr{A}$, after $\mathscr{A}$ executes any transition, the following conditions hold for all $1\leqslant i,\ j\leqslant k$ and for each integer c, $-m<c<m$:

(1) $x_i-x_j\#c$ iff $a_{ij}\#c$,
(2) $x_i\#c$ iff $b_i\#c$.

The proof of this statement (omitted here) is a variant of the one in [12]. Thus, each edge in $\mathscr{A}$ is modified so that:

- Each test $x_i - x_j \# c$ and $x_i \# c$ is replaced by *a replaced test* $a_{ij} \# c$ and $b_i \# c$, respectively,
- the entry updating instructions are added.

The resulting automaton is denoted by $\mathscr{A}'$.

The *replaced tests* and the entry updating instructions in $\mathscr{A}'$ can be further eliminated by expanding the states S. The resulting automaton is called $\mathscr{A}''$ with states $S'' \subseteq S \times [m]^{(k^2+k)}$. In short, each *expanded state* in S'' is a tuple of the *original state* $s \in S$ and values (totally, $k^2 + k$ many) of all entries a_{ij} and b_i, $1 \leqslant i,\ j \leqslant k$. Each edge e (connecting a pair of states s_1, s_2 in S) in $\mathscr{A}'$ is thus split into a finite number of edges in $\mathscr{A}''$. Each split edge in $\mathscr{A}''$ connects two expanded states s_1'', s_2'' in S'', corresponding to the original states s_1 and s_2, respectively, such that: (a) the values of the entries in s_1'' satisfy the replaced test on e (thus, the replaced tests are eliminated in $\mathscr{A}''$); (b) the values of the entries in s_1'' and s_2'' are consistent with the entry updating instructions on e (thus the entry updating instructions are eliminated in $\mathscr{A}''$). Each enabling condition in $\mathscr{A}''$ is simply *true*. Thus, $\mathscr{A}''$ is a static *dta*.

It should be clear that $\mathscr{A}''$ simulates $\mathscr{A}$. That is, an ω-path in $\mathscr{A}$ corresponds to an ω-path in $\mathscr{A}''$ (and vice versa) while preserving the exact sequence of clock values. This can be precisely stated in the following theorem:

Theorem 2. *Given a dta $\mathscr{A}$ and a static dta $\mathscr{A}''$ constructed as above,*

$$\langle s_0, \boldsymbol{V}^0 \rangle \cdots \langle s_k, \boldsymbol{V}^k \rangle \cdots$$

is a ω-path of $\mathscr{A}$ iff there exists an ω-path

$$\langle s_0'', \boldsymbol{V}^0 \rangle \cdots \langle s_k'', \boldsymbol{V}^k \rangle \cdots$$

of $\mathscr{A}''$ such that:

- *For each $i \geqslant 0$, the original state of the extended state s_i'' is equal to s_i.*
- *The entry values a_{ij} and b_i in the extended state s_0'' are the initial entry values constructed from $\boldsymbol{V}^0$.*

Now we look at the $\exists$-Presburger-i.o. problem for $\mathscr{A}$. Recall that the problem is to determine, given two Presburger-definable sets I and P of configurations of $\mathscr{A}$, whether there exists a P-i.o. ω-path p starting from I (p is called a *witness*). We relate the instance of the $\exists$-Presburger-i.o. problem for $\mathscr{A}$ to an instance of the $\exists$-Presburger-i.o. problem for the static *dta* $\mathscr{A}''$:

Lemma 1. *Given a dta $\mathscr{A}$, and two Presburger-definable sets I and P of configurations of $\mathscr{A}$, there exist a static dta $\mathscr{A}''$ and two Presburger definable sets I'' and P'' of configurations of $\mathscr{A}''$ such that: the existence of a witness to the $\exists$-Presburger-i.o. for $\mathscr{A}$, given I and P, is equivalent to the existence of a witness to the $\exists$-Presburger-i.o. for $\mathscr{A}''$, given I'' and P''.*

Proof. Define I'' and P'' to be two sets of configurations of $\mathscr{A}''$ as follows. A configuration of $\mathscr{A}''$ is in I'' if and only if the configuration is in I (when the entry values

a_{ij} and b_i are ignored) and the entry values are the initial entry values constructed from clock values in the configuration. That is,

$$\langle s, a_{11}, \ldots, a_{1k}, \ldots, a_{k1}, \ldots, a_{kk}, b_1, \ldots, b_k, \boldsymbol{V}\rangle \in I''$$

iff

- $\langle s, \boldsymbol{V}\rangle \in I$,
- For each $1 \leqslant i, j \leqslant k$, a_{ij} and b_i are the initial entry values constructed from clock values $\boldsymbol{V}$ as we mentioned before. That is, for each $1 \leqslant i,\ j \leqslant k$:
  - $a_{ij} = \boldsymbol{V}_i - \boldsymbol{V}_j$ if $|\boldsymbol{V}_i - \boldsymbol{V}_j| \leqslant m$,
  - $a_{ij} = m$ if $\boldsymbol{V}_i - \boldsymbol{V}_j > m$,
  - $a_{ij} = -m$ if $\boldsymbol{V}_i - \boldsymbol{V}_j < -m$,
  - $b_i = \boldsymbol{V}_i$ if $\boldsymbol{V}_i \leqslant m$,
  - $b_i = m$ if $\boldsymbol{V}_i > m$.

Obviously, I'' is Presburger-definable. A configuration of $\mathscr{A}''$ is in P'' if and only if the configuration is in P when the entry values a_{ij} and b_i are ignored. That is,

$$\langle s, a_{11}, \ldots, a_{1k}, \ldots, a_{k1}, \ldots, a_{kk}, b_1, \ldots, b_k, \mathbf{X}\rangle \in P''$$

iff $\langle s, \mathbf{X}\rangle \in P$. It is also obvious that P'' is Presburger-definable.

Notice that, in Theorem 2, $\langle s_0'', \boldsymbol{V}^0\rangle \in I''$ if $\langle s_0, \boldsymbol{V}^0\rangle \in I$. Also notice that $\langle s_0, \boldsymbol{V}^0\rangle \cdots \langle s_k, \boldsymbol{V}^k\rangle \cdots$ and $\langle s_0'', \boldsymbol{V}^0\rangle \cdots \langle s_k'', \boldsymbol{V}^k\rangle \cdots$ have exactly the same sequence of clock values. Therefore, in Theorem 2, $\langle s_0, \boldsymbol{V}^0\rangle \cdots \langle s_k, \boldsymbol{V}^k\rangle \cdots$, with $\langle s_0, \boldsymbol{V}^0\rangle \in I$, is a P-i.o. ω-path of $\mathscr{A}$ iff $\langle s_0'', \boldsymbol{V}^0\rangle \cdots \langle s_k'', \boldsymbol{V}^k\rangle \cdots$ is a P''-i.o. ω-path of $\mathscr{A}''$ with $\langle s_0'', \boldsymbol{V}^0\rangle \in I''$. Thus, the existence of a witness to the $\exists$-Presburger-i.o. for $\mathscr{A}$, given I and P, is equivalent to the existence of a witness to the $\exists$-Presburger-i.o. for $\mathscr{A}''$, given I'' and P''. □

Due to Lemma 1, it suffices for us to consider static *dta*s in showing the decidability of the $\exists$-Presburger-i.o. problems.

3.2. The $\exists$-Presburger-i.o. problem for static dtas

In this subsection we show that the $\exists$-Presburger-i.o. problem for static *dta*s is decidable. Let $\mathscr{A}$ be a static *dta* throughout this subsection. Recall that, given two sets I and P of configurations of $\mathscr{A}$ definable by Presburger formulas, an ω-path $p = \langle s_0, \boldsymbol{V}^0\rangle \cdots \langle s_k, \boldsymbol{V}^k\rangle \cdots$ is a witness to the $\exists$-Presburger-i.o. problem if p is P-i.o. and p starts from I (i.e., $\langle s_0, \boldsymbol{V}^0\rangle \in I$). There are two cases to consider: (1) p is not divergent; (2) p is divergent. For the first case, the following lemma can be easily established from the binary reachability of $\mathscr{A}$.

Lemma 2. *The existence of a non-divergent witness is decidable.*

Proof. Let $\mathscr{A}'$ be the automaton obtained by dropping all the clock progress transitions from $\mathscr{A}$ (thus, each path included in an ω-path of $\mathscr{A}'$ is a non-progress path). Observe that there exists a nondivergent witness iff there are s and $\boldsymbol{V}$ such that

- there exists $\langle s_0, \boldsymbol{V}^0\rangle \in I$ such that $\langle s_0, \boldsymbol{V}^0\rangle \rightsquigarrow^{\mathscr{A}} \langle s, \boldsymbol{V}\rangle$. That is, $\langle s, \boldsymbol{V}\rangle$ is reachable from a configuration in I,

- $\langle s, \boldsymbol{V}\rangle \in P$, and
- $\langle s, \boldsymbol{V}\rangle$ can reach itself (i.e., a cycle) in $\mathscr{A}'$.

The statement of the lemma follows immediately from the fact that each item above is Presburger-definable (Theorem 1). □

The remainder of this subsection is devoted to the proof that the existence of a divergent witness is decidable. In order to do this, we need the following definition of a *progress cycle* (in contrast to the cycle defined in the previous proof).

Definition 4. Let s be a control state, $X_r \subseteq X$ be a set of clocks,[1] and $\boldsymbol{V}$ and $\boldsymbol{V}'$ be clock vectors. We write $\langle s, \boldsymbol{V}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}'\rangle$ if

1. $\langle s, \boldsymbol{V}\rangle$ is reachable from a configuration in I,
2. $\langle s, \boldsymbol{V}'\rangle \in P$,
3. $\langle s, \boldsymbol{V}\rangle \leadsto^{\mathscr{A}} \langle s, \boldsymbol{V}'\rangle$ through a *progress* path where all the clocks in X_r are reset at least once and all the clocks not in X_r are not reset.

The proof proceeds as follows. Since $\mathscr{A}$ has finitely many control states, there exists a P-i.o. ω-path p iff there is a state s such that P holds infinitely often on p at state s. This is equivalent to saying (Lemma 3) that there exist clock vectors $\boldsymbol{V}^1, \boldsymbol{V}^2, \ldots$ such that $\langle s, \boldsymbol{V}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}^{i+1}\rangle$ for each $i > 0$. We then show (Lemma 4) that the relation $\leadsto^{\mathscr{A}}_{X_r}$ is Presburger-definable. Since the actual values of the clocks in X_r may be abstracted away (Lemma 5 and Definition 5) and the clocks in $X - X_r$ progress synchronously, this is equivalent to saying that there exist $\boldsymbol{V}, d_1 > 0, d_2 > 0, \ldots$ such that $\boldsymbol{V}^i_x = \boldsymbol{V}_x + d_i$ for all $x \in X - X_r$ (Lemma 6). The set $\{d_i\}$ may be defined with a Presburger formula, as shown in Lemma 7, since each d_i may always be selected to be of the form $c_i + f(c_i)$, where the set $\{c_i\}$ is a periodic set (hence, Presburger-definable) and f is a Presburger-definable function. This is based on the fact that static automata have no edge conditions, allowing us to increase the length d of a progress cycle to a length nd (Lemma 8), for every $n > 0$. The decidability result on the existence of a divergent witness follows directly from Lemma 8.

Lemma 3. *There is a divergent witness p iff there are s, X_r and clock vectors $\boldsymbol{V}^1, \boldsymbol{V}^2, \ldots$ such that*

$$\langle s, \boldsymbol{V}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}^{i+1}\rangle$$

for each $i > 0$.

Proof. Immediate from Definition 4. □

Lemma 4. $\leadsto^{\mathscr{A}}_{X_r}$ *is Presburger-definable. That is, given $s \in S$,*

$$\langle s, \boldsymbol{V}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}'\rangle$$

[1] We assume $X - X_r \neq \emptyset$. This can be achieved by adding a dummy clock *now* that never resets and that indicates the current time.

is a Presburger formula, when the clock vectors $\boldsymbol{V}, \boldsymbol{V}'$ are regarded as integer variables.

Proof. Let R denote all tuples $\langle s, \boldsymbol{V}\rangle, \langle s, \boldsymbol{V}'\rangle$ such that condition (3) of Definition 4 holds. A reversal-bounded multicounter machine [19] is a one-way (input) finite automaton augmented with finitely many reversal-bounded counters (i.e., each counter can be incremented or decremented by one and tested for zero, but the number of times it changes mode from nondecreasing to nonincreasing and vice versa is bounded by a constant, independent of the computation). In [12], we have shown that $\rightsquigarrow^{\mathscr{A}}$ can be accepted by a reversal-bounded multicounter machine M (i.e., Theorem 1, since reversal-bounded counter machines accept only Presburger-definable sets when the input is of integer tuples [19].). M can be easily modified to a machine M' accepting R as follows. M', with $\langle s, \boldsymbol{V}\rangle$ and $\langle s, \boldsymbol{V}'\rangle$ on its input tape, simulates $\mathscr{A}$ starting from configuration $\langle s, \boldsymbol{V}\rangle$ exactly as M. But during the simulation, M' makes sure, using its own finite control, that

- each clock in X_r resets at least once,
- each clock not in X_r does not reset,
- there is at least one clock progress cycle executed.

At some moment, M' guesses that the configuration $\langle s, \boldsymbol{V}'\rangle$ has been reached and compares it with the current configuration of $\mathscr{A}$ on the input tape (see [12] for details). Thus, R can be accepted by M'. Therefore, R is Presburger-definable.

Now, $\langle s, \boldsymbol{V}\rangle \rightsquigarrow^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}'\rangle$ can be rewritten as

$$\exists \langle s_0, \boldsymbol{V}^0\rangle \in I(\langle s_0, \boldsymbol{V}^0\rangle \rightsquigarrow^{\mathscr{A}} \langle s, \boldsymbol{V}\rangle \wedge \langle s, \boldsymbol{V}'\rangle \in P \wedge (\langle s, \boldsymbol{V}\rangle, \langle s, \boldsymbol{V}'\rangle) \in R).$$

Clearly, this is a Presburger formula on $\boldsymbol{V}$ and $\boldsymbol{V}'$, since I, P and R are Presburger, and it is known that Presburger formulas are closed under quantification. □

$\langle s, \boldsymbol{V}\rangle \rightsquigarrow^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{W}\rangle$ denotes the following scenario. Starting from some configuration in I, $\mathscr{A}$ can reach $\langle s, \boldsymbol{V}\rangle$ and return to s again with clock values $\boldsymbol{W}$. The cycle at s is a progress such that each clock in X_r resets at least once and all clocks not in X_r do not reset. Since $\mathscr{A}$ is static, the cycle can be represented by a sequence $s_0 s_1 \cdots s_t$ of control states, with $s_0 = s_t = s$, and such that, for each $0 \leqslant i < t$, there is an edge in $\mathscr{A}$ connecting s_i and s_{i+1}. Observe that, since each $x \in X_r$ is reset in the cycle, the starting clock values $\boldsymbol{V}_x$ for $x \in X_r$ at $s_0 = s$ are insensitive to the ending clock values $\boldsymbol{W}_x$ with $x \in X_r$ at $s_t = s$ (these values of $\boldsymbol{W}_x$ only depend on the sequence of control states). We write $\boldsymbol{V} =_{X - X_r} \boldsymbol{U}$ if $\boldsymbol{V}$ and $\boldsymbol{U}$ agree on the values of the clocks not in X_r, i.e., $\boldsymbol{V}_x = \boldsymbol{U}_x$, for each $x \in X - X_r$. The insensitivity property for static *dta*s is stated in the following lemma.

Lemma 5. *For all clock vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$, if*

$$\langle s, \boldsymbol{V}\rangle \rightsquigarrow^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{W}\rangle$$

and $\langle s, \boldsymbol{U}\rangle$ is reachable from some configuration in I with $\boldsymbol{V} =_{X - X_r} \boldsymbol{U}$, then

$$\langle s, \boldsymbol{U}\rangle \rightsquigarrow^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{W}\rangle.$$

Also note that, since all clocks not in X_r do not reset and progress synchronously on the cycle, the differences $\boldsymbol{W}_x - \boldsymbol{V}_x$ for each $x \in X - X_r$ are equal to the *duration* of the cycle (i.e., the number of progress transitions in the cycle). The following technical definition allows us to "abstract" clock values for X_r away in $\langle s, \boldsymbol{V}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{W}\rangle$.

Definition 5. For all clock vectors $\boldsymbol{Y}$ and $\boldsymbol{Y}'$, we write

$$\boldsymbol{Y} \leadsto^{\mathscr{A}}_{\langle s, X_r\rangle} \boldsymbol{Y}'$$

if there exist two clock vectors $\boldsymbol{V}$ and $\boldsymbol{W}$ such that $\langle s, \boldsymbol{V}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{W}\rangle$ with $\boldsymbol{Y} =_{X-X_r} \boldsymbol{V}$ and $\boldsymbol{Y}' =_{X-X_r} \boldsymbol{W}$.

Obviously, the relation $\leadsto^{\mathscr{A}}_{\langle s, X_r\rangle}$ is Presburger-definable.

Lemma 6. *There exists a divergent witness for $\mathscr{A}$ if and only if there are s, X_r, $\boldsymbol{Y}$, d_1, $d_2, \ldots$ such that*

$$0 \leqslant d_1 < d_2 < \cdots$$

and

$$\boldsymbol{Y} + d_i\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s, X_r\rangle} \boldsymbol{Y} + d_{i+1}\mathbf{1},$$

for each $i \geqslant 1$.

Proof (if-part). By Definition 5, for each $i \geqslant 1$,

$$\boldsymbol{Y} + d_i\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s, X_r\rangle} \boldsymbol{Y} + d_{i+1}\mathbf{1}$$

if there exist $\boldsymbol{A}^i, \boldsymbol{B}^i$, such that $\langle s, \boldsymbol{A}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{B}^i\rangle$ with a cycle of duration $d_{i+1} - d_i$, and with $\boldsymbol{A}^i =_{X-X_r} \boldsymbol{Y} + d_i\mathbf{1}$ and $\boldsymbol{B}^i =_{X-X_r} \boldsymbol{Y} + d_{i+1}\mathbf{1}$. By Lemma 5, for each $i \geqslant 1$,

$$\langle s, \boldsymbol{B}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{B}^{i+1}\rangle.$$

From Lemma 3, there is a divergent witness.

(*only-if-part*). Conversely, assume the existence of a divergent witness, i.e., by Lemma 3, there are clock vectors $\boldsymbol{B}^1, \boldsymbol{B}^2, \ldots$ such that for each $i \geqslant 1$,

$$\langle s, \boldsymbol{B}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{B}^{i+1}\rangle.$$

Let $\boldsymbol{Y} = \boldsymbol{B}^1$. For each $i \geqslant 1$, let d_i such that

$$d_i\mathbf{1} =_{X-X_r} \boldsymbol{B}^i - \boldsymbol{Y}.$$

Therefore, for each $i \geqslant 1$, $\boldsymbol{B}^i =_{X-X_r} \boldsymbol{Y} + d_i\mathbf{1}$: by Definition 5,

$$\boldsymbol{Y} + d_i\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s, X_r\rangle} \boldsymbol{Y} + d_{i+1}\mathbf{1}. \qquad \square$$

The following lemma uses the fact that a cycle in a static *dta* can be repeated. It will be used in the proof of Lemma 8.

Lemma 7. *For all* $\boldsymbol{Y}, \boldsymbol{Y}'$, $d>0$, *if* $\boldsymbol{Y} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}+d\mathbf{1}$ *and, for some* $n>1$, $\boldsymbol{Y}+nd\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}'$, *then* $\boldsymbol{Y}+d\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}'$.

Proof. By Definition 5, since $\boldsymbol{Y} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}+d\mathbf{1}$, there are clock vectors $\boldsymbol{A},\boldsymbol{B}$ such that $\boldsymbol{Y} =_{X-X_r} \boldsymbol{A}$, $\boldsymbol{Y}+d\mathbf{1} =_{X-X_r} \boldsymbol{B}$, and $\langle s,\boldsymbol{A}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s,\boldsymbol{B}\rangle$ with a progress cycle of duration d. Moreover, again by Definition 5, since $\boldsymbol{Y}+nd\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}'$, there exist $\boldsymbol{C},\boldsymbol{E}$ such that $\langle s,\boldsymbol{C}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s,\boldsymbol{E}\rangle$ with $\boldsymbol{C} =_{X-X_r} \boldsymbol{Y}+nd\mathbf{1}$ and $\boldsymbol{E} =_{X-X_r} \boldsymbol{Y}'$. Since $\mathscr{A}$ is static, a cycle from state s to state s may be repeated any number of times starting from $\langle s,\boldsymbol{B}\rangle$. That is, there exists $\langle s,\boldsymbol{B}'\rangle$ such that from $\langle s,\boldsymbol{B}\rangle$ we may reach $\langle s,\boldsymbol{B}'\rangle$ with a cycle of duration $(n-1)d$, which verifies condition (3) of Definition 4. But $\boldsymbol{B}' =_{X-X_r} \boldsymbol{C}$, and $\langle s,\boldsymbol{B}'\rangle$ is reachable from $\langle s,\boldsymbol{A}\rangle$ and hence from I. By Lemma 5, $\langle s,\boldsymbol{B}'\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s,\boldsymbol{E}\rangle$. Therefore, $\langle s,\boldsymbol{B}\rangle$ can reach $\langle s,\boldsymbol{E}\rangle$ through a progress cycle on which all and only the clocks in X_r are reset at least once, $\langle s,\boldsymbol{B}\rangle$ is reachable from I and $\langle s,\boldsymbol{E}\rangle \in P$: by Definition 5, $\langle s,\boldsymbol{B}\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s,\boldsymbol{E}\rangle$, i.e., $\boldsymbol{Y}+d\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}'$. □

Lemma 8. *The existence of a divergent witness is decidable for a static dta* $\mathscr{A}$.

Proof. We claim that, there are s, X_r such that the Presburger formula

$$(*)\quad \exists Y \forall m > 0 \exists D_1 \geqslant m \exists D_2 > 0\ (\boldsymbol{Y}+D_1\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}+(D_1+D_2)\mathbf{1})$$

holds if and only if there is a divergent witness for $\mathscr{A}$. The statement of the lemma then follows immediately.

Assume there is a divergent witness. Then, by Lemma 3, there exist $\boldsymbol{V}^1, \boldsymbol{V}^2, \ldots$ and $d_1, d_2, \ldots$ such that, for each $i \geqslant 1$,

$$\langle s, \boldsymbol{V}^i\rangle \leadsto^{\mathscr{A}}_{X_r} \langle s, \boldsymbol{V}^{i+1}\rangle$$

with a progress cycle of duration $d_i>0$. Let $\boldsymbol{Y}=\boldsymbol{V}^1$. By Definition 5,

$$\boldsymbol{Y}+\left(\sum_{j=1}^{i-1} d_j\right)\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}+\left(\sum_{j=1}^{i} d_j\right)\mathbf{1}$$

for each $i \geqslant 1$. For each $m>0$, let

$$D_1 = \sum_{j=1}^{m} d_j$$

and $D_2 = d_{m+1}$. It is immediate that (*) holds.

Conversely, let $\boldsymbol{Y}_0$ be one of the vectors $\boldsymbol{Y}$ such that $(*)$ holds. Since $\boldsymbol{Y}_0$ is fixed, the formula $H(D_1)$, defined as

$$\exists D_2 > 0(\boldsymbol{Y}_0+D_1\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}_0+(D_1+D_2)\mathbf{1}),$$

is Presburger-definable. Apply skolemization to the above formula by introducing a function $f(D_1)$ to replace the variable D_2:

$$\boldsymbol{Y}_0+D_1\mathbf{1} \leadsto^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}_0+(D_1+f(D_1))\mathbf{1}.$$

Since (*) holds, then $H(D_1)$ holds for infinitely many values of D_1. Combining the fact that $H(D_1)$ is Presburger-definable, there is a periodic set included in the infinite domain of H, i.e., there exist $n>1, k\geqslant 0$ such that for all $d\geqslant 0$ if $d\equiv_n k$ then $H(d)$ holds. Let c_0 be any value in the periodic set, and let

$$c_i = c_{i-1} + nf(c_{i-1}),$$

for every $i\geqslant 1$. Obviously, every c_i satisfies the periodic condition: $c_i\equiv_n k$, and therefore $H(c_i)$ holds. Hence, for every $i\geqslant 1$,

$$\boldsymbol{Y}_0 + c_i\mathbf{1} \rightsquigarrow^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}_0 + (c_i + f(c_i))\mathbf{1}.$$

Since

$$\boldsymbol{Y}_0 + c_{i+1}\mathbf{1} = \boldsymbol{Y}_0 + c_i\mathbf{1} + nf(c_i)\mathbf{1} \rightsquigarrow^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}_0 + (c_{i+1} + f(c_{i+1}))\mathbf{1},$$

we may apply Lemma 7, with: $\boldsymbol{Y} = \boldsymbol{Y}_0 + c_i\mathbf{1}$, $d = f(c_i)$, and

$$\boldsymbol{Y}' = \boldsymbol{Y}_0 + (c_{i+1} + f(c_{i+1}))\mathbf{1}.$$

Lemma 7 then gives $\boldsymbol{Y} + d\mathbf{1} \rightsquigarrow^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}'$, i.e.,

$$\boldsymbol{Y}_0 + (c_i + f(c_i))\mathbf{1} \rightsquigarrow^{\mathscr{A}}_{\langle s,X_r\rangle} \boldsymbol{Y}_0 + (c_{i+1} + f(c_{i+1}))\mathbf{1},$$

for every $i\geqslant 1$. By Lemma 6, with $d_i = c_i + f(c_i)$, there is a divergent witness. □

By Lemmas 2 and 8, we have:

Theorem 3. *The ∃-Presburger-i.o. problem is decidable for static dtas.*

Since $\mathscr{A}''$ in Lemma 1 is a static *dta*, the decidability of the ∃-Presburger-i.o. problem for *dta*s follows from Lemma 1 and Theorem 3. Also notice that deciding the ∃-Presburger-i.o. problem is equivalent to deciding the negation of the ∀-Presburger-almost-always problem. Thus,

Theorem 4. *The ∃-Presburger-i.o. problem and the ∀-Presburger-almost-always problem are decidable for dtas.*

Since the length of the Presburger formula for the binary reachability of a *dta* is unknown, we cannot give an accurate complexity bound for the decision procedure of the ∃-Presburger-i.o. problem. However, if we assume that L is the maximum of the length of the binary reachability (written in a quantifier-free linear relation) for the static *dta* translated from the given *dta*, the length of the initial condition I, and the property P, the complexity bound is $exp(L^{(cn)^3})$ for some constant c, where n is the number of clocks in the *dta*. This can be obtained by looking at the complexity for eliminating the quantifiers in formula (*) in the proof of Lemma 8 and using the result in [23]. Hence, the complexity bound for the ∃-Presburger-i.o. problem is at least double exponential in the number of clocks.

3.3. Decidability of the ∃-Presburger-eventual problem

Given a *dta* $\mathscr{A}$, and two Presburger-definable sets I and P of configurations, the ∃-Presburger-eventual problem is to decide whether there exists a P-eventual ω-path p starting from I. Define I' to be the set of all configurations in P that can be reached from a configuration in I. From Theorem 1, I' is Presburger-definable. One can see that

$$p = \langle s_0, \boldsymbol{V}^0\rangle \cdots \langle s_i, \boldsymbol{V}^i\rangle \cdots$$

is P-eventual and $\langle s_0, \boldsymbol{V}^0\rangle \in I$ iff there exists i such that $\langle s_i, \boldsymbol{V}^i\rangle \in I'$ and the ω-path $\langle s_i, \boldsymbol{V}^i\rangle \cdots$ is *true*-i.o. Therefore, the existence of a witness for the ∃-Presburger-eventual problem (given I and P) is equivalent to the existence of a witness for the ∃-Presburger-i.o. problem (given I' and *true*). From Theorem 4, we have:

Theorem 5. *The ∃-Presburger-eventual problem and the ∀-Presburger-always problem are decidable for dtas.*

It should be noted that, for *dtas*, there is a slight difference between the ∀-Presburger-always problem and the Presburger safety analysis problem mentioned before. The difference is that the Presburger safety analysis problem considers (finite) paths while the ∀-Presburger-always problem considers ω-paths.

4. Undecidability results

The next three subsections show that the undecidability of the ∀-Presburger-eventual problem and of the ∀-Presburger-i.o. problem. We start by demonstrating the fact that a two-counter machine can be implemented by a generalized version of a *dta*. This fact is then used in the following two subsections to show the undecidability results.

4.1. Counter machines and generalized discrete timed automata

Consider a counter machine M with counters $x_1,\ldots,x_k$ over nonnegative integers and with a finite set of locations $\{l_1,\ldots,l_n\}$. M can increment, decrement and test against 0 the values of the counters. It is well-known that a two-counter machine can simulate a Turing machine.

We now define *generalized* discrete timed automata. They are defined similarly to *dta*s but for each edge $\langle s,\lambda,l,s'\rangle$ the enabling condition l is of the form $\sum_i a_i x_i \# c$, where a_i and c are integers. The following lemma states that generalized *dta*s are Turing-complete, since they can simulate any counter machine. The proof can be found in the appendix.

Lemma 9. *Given a deterministic counter machine M, there exists a deterministic generalized dta that can simulate M.*

From now on, let M be a deterministic counter machine and let $\mathscr{A}$ be a deterministic generalized *dta* that implements M. We may assume that $\mathscr{A}$ is total (i.e., there are no deadlock configurations), since $\mathscr{A}$ can be made total by adding a new self-looped state s_f, and directing every deadlock configuration to this new state. Now we define the *static version* $\mathscr{A}^-$, to which $\mathscr{A}$ can be modified as follows. $\mathscr{A}^-$ is a discrete timed automaton with the enabling condition on each edge being simply *true*. Each state in $\mathscr{A}^-$ is a pair of states in $\mathscr{A}$.

$$\langle\langle s_1, s_1'\rangle, \lambda_1, \mathit{true}, \langle s_2, s_2'\rangle\rangle$$

is an edge of $\mathscr{A}^-$ iff there are edges $\langle s_1, \lambda_1, l_1, s_1'\rangle$ and $\langle s_2, \lambda_2, l_2, s_2'\rangle$ in $\mathscr{A}$ with $s_1' = s_2$. We define a set P, called the *path restriction of* $\mathscr{A}$, of configurations of $\mathscr{A}^-$ as follows. For each configuration $\langle\langle s, s'\rangle, \boldsymbol{V}\rangle$ of $\mathscr{A}^-$, $\langle\langle s, s'\rangle, \boldsymbol{V}\rangle \in P$ iff there exists an edge $e = \langle s, \lambda, l, s'\rangle$ in $\mathscr{A}$ such that the clock values $\boldsymbol{V}$ satisfy the linear relation l in e. Clearly, P is Presburger-definable. Since $\mathscr{A}$ is total and deterministic, the above edge e always exists and is unique for each configuration $\langle s, \boldsymbol{V}\rangle$ of $\mathscr{A}$. Using this fact, we have:

Theorem 6. *Let $\mathscr{A}$ be a total and deterministic generalized dta with path restriction P, and let $\mathscr{A}^-$ be the static version of $\mathscr{A}$. An ω-sequence*

$$\langle s_0, \boldsymbol{V}^0\rangle \cdots \langle s_k, \boldsymbol{V}^k\rangle \cdots$$

is an ω-path of $\mathscr{A}$ iff

$$\langle\langle s_0, s_1\rangle, \boldsymbol{V}^0\rangle \cdots \langle\langle s_k, s_{k+1}\rangle, \boldsymbol{V}^k\rangle \cdots$$

is an ω-path of $\mathscr{A}^-$ with $\langle\langle s_k, s_{k+1}\rangle, \boldsymbol{V}^k\rangle \in P$ for each k.

4.2. Undecidability of the ∀-Presburger-eventual problem

We consider the negation of the ∀-Presburger-eventual problem, i.e., the ∃-Presburger-always problem, which can be formulated as follows: given a discrete timed automaton $\mathscr{A}$ and two Presburger-definable sets I and P of configurations, decide whether there exists a $\neg P$-always ω-path of $\mathscr{A}$ starting from I.

Consider a deterministic counter machine M with the initial values of the counters being 0 and the first instruction labeled l_0. Let $\mathscr{A}$ be the deterministic generalized *dta* implementing M, as defined by Lemma 9, with P being the path restriction of $\mathscr{A}$. As before, $\mathscr{A}$ is total. Let $\mathscr{A}^-$ be the static version of $\mathscr{A}$. It is well known that the halting problem for (deterministic) counter machines is undecidable. That is, it is undecidable, given M and an instruction label l, whether M executes the instruction l. Define P' to be the set of configurations $\langle\langle s, s'\rangle, \boldsymbol{V}\rangle \in P$ with $s \neq l$. Let I be the set of initial configurations of $\mathscr{A}^-$ with all the clocks being 0 and the first component of the state (note that each state in $\mathscr{A}^-$ is a state pair of $\mathscr{A}$) being l_0. I is finite, thus Presburger-definable. From Theorem 6 and the fact that $\mathscr{A}$ implements M, we have:

M does not halt at l iff $\mathscr{A}^-$ has a P'-always ω-path starting from a configuration in I.

Thus, we reduce the negation of the halting problem to the $\exists$-Presburger-always problem for *dta*s with configuration sets P' and I. Therefore,

Theorem 7. *The $\exists$-Presburger-always problem and the $\forall$-Presburger-eventual problem are undecidable for discrete timed automata.*

4.3. Undecidability of the $\forall$-Presburger-i.o. problem

The $\forall$-Presburger-i.o. problem for discrete timed automata is to decide, given a discrete timed automaton $\mathscr{A}$ and two Presburger-definable sets I and P of configurations, whether p is P-i.o. for every ω-path p starting from I. The negation of the problem, i.e., the $\exists$-Presburger-almost-always problem, can be formulated as follows: whether there exists a $\neg P$-almost-always ω-path of $\mathscr{A}$ starting from I. In this subsection, we show that the $\exists$-Presburger-almost-always problem is also undecidable. Therefore, the $\forall$-Presburger-i.o. problem is also undecidable.

In the previous subsection we have shown that the existence of a P-always ω-path of $\mathscr{A}$ is undecidable. But this result does not directly imply that the existence of a P-almost-always ω-path is also undecidable.

Let $\mathscr{A}^-$ be the static version of a deterministic generalized discrete timed automaton $\mathscr{A}$ that implements a deterministic counter machine M, let P be the path restriction of $\mathscr{A}$, and let p be an ω-path of $\mathscr{A}^-$. In the previous subsection we argued that the existence of a P'-always ω-path p is undecidable where P' is $P \cap \{\langle\langle s, s'\rangle, \boldsymbol{V}\rangle : s \neq l\}$ with l being a given instruction label in M. But when considering a P'-almost-always path p, the situation is different: p may have a prefix that does not necessarily satisfy P' (i.e., it does not obey the exact enabling conditions on the edges in $\mathscr{A}$).

Consider a deterministic two-counter machine M with an input tape, and denote with $M(i)$ the result of the computation of M when given $i \in \mathbf{N}$ as input. It is known that the finiteness problem for deterministic two-counter machines (i.e., finitely many i such that $M(i)$ halts) is undecidable. Now we reduce the finiteness problem to the $\exists$-Presburger-almost-always problem for *dta*s.

We can always assume that M halts when and only when it executes an operation labeled *halt*. Let M' be a counter machine (without input tape) that enumerates all the computations of M on every $i \in \mathbf{N}$. M' works as follows. We use $M_j(i)$ to denote the jth step of the computation of $M(i)$. If $M(i)$ halts in less than j steps, then we assume that $M_j(i)$ is a special null operation that does nothing. Thus, the entire computation of $M(i)$ is an ω-sequence $M_1(i), \ldots, M_j(i), \ldots$ (when $M(i)$ halts, the sequence is composed of a finite prefix, followed by the halt operation and then by infinitely many occurrences of the special null operation). Each step of the computation may or may not execute the instruction labeled *halt*, but of course a halt may be executed only at most once for each input value i. M' implements the following program:

$k := 0;\ z := 0;$
while true do
  $k := k + 1;$
  for $i := 0$ *to* $k - 1$ *do*

$z := 1$
simulate $M(i)$ *for the first* k *steps* $M_1(i), M_2(i), \ldots, M_k(i)$;
if $M_k(i)$ *executes the instruction labeled halt, then* $z := 0$;

M' is still a deterministic counter machine (with various additional counters to be able to simulate M and keep track of k, i, z). In the enumeration, whenever $M_k(i)$ executes the instruction labeled *halt* (at most once for each i, by the definition of M' as above), M' sets the counter z to 0, bringing it back to 1 immediately afterwards—M' resets z to 0 for only finitely many times iff the domain of M (i.e., the set of i such that $M(i)$ halts) is finite. Let $\mathscr{A}^-$ be the static version of a generalized discrete timed automaton $\mathscr{A}$ that implements M'. Let P be the path restriction of $\mathscr{A}$. P' is $P \cap \{\langle\langle s, s'\rangle, \boldsymbol{V}\rangle : \boldsymbol{V}_z \neq 0\}$. Then:

there are only finitely many i such that $M(i)$ halts
$\Leftrightarrow$
there are only finitely many configurations with $z = 0$ in the computation of M'
$\Leftrightarrow$
in the (unique) ω-path $\langle s_0, \boldsymbol{V}^0\rangle\langle s_1, \boldsymbol{V}^1\rangle \cdots$ of M',
there are only finitely many i such that $\boldsymbol{V}_z^i = 0$
$\Leftrightarrow$ (Lemma 9 and Theorem 6)
there is an ω-path p of $\mathscr{A}^-$ such that p is P'-almost-always and p starts from the initial configuration with all clocks being 0 (since all counters in M' start from 0)
$\Leftrightarrow$
$\mathscr{A}^-$ is $\exists$-Presburger-almost-always against P' and I contains only the initial configuration.

Therefore,

Theorem 8. *The* $\exists$*-Presburger-almost-always problem and the* $\forall$*-Presburger-i.o. problem are undecidable for discrete timed automata.*

5. A verification example

In this section, we illustrate how the paper's results on decidability of Presburger safety and liveness properties may help in the verification of real-time systems.

We consider a traditional benchmark in real-time specifications: the railroad crossing problem, following the description given in [15]. The system to be developed operates a gate at a railroad crossing. We assume that there is only one track, and the train may cross only in one direction. The behavior of the train is shown by the nondeterministic timed automaton whose transition graph is given in Fig. 2. A train may be *far* from the critical region around the crossing; when it enters the region, it is *approaching* the gate, until it enters the crossing. The train then exits and another train may enter, after a suitable delay, starting another cycle of far/approaching/crossing. Signals on the railroad ensure that at any time only one train may be approaching or crossing.

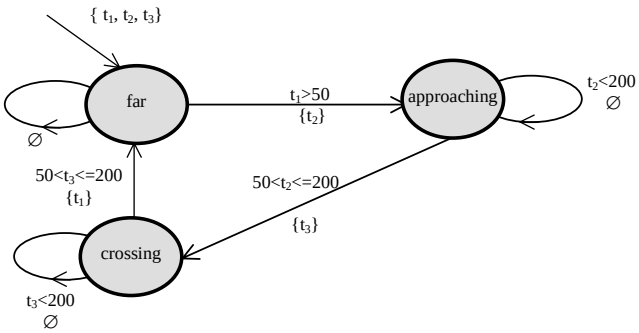

Fig. 2. The train.

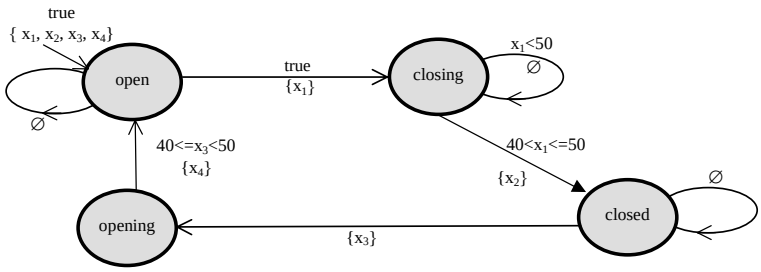

Fig. 3. The controller.

The transition graph of the train has a few integer delays: we assume that it takes 50 time units for another train to enter the critical region after a crossing (this time is measured by clock t_1, which is reset whenever entering state *far*). The train stays in the approaching region at least 50 but at most 200 time units (i.e., there are bounds on the minimum and maximum speed of the train). This is measured by the clock t_2, which is reset whenever entering state *approaching*. Then the train crosses the gate, again for an interval lasting between 50 and 200 time units, as measured by clock t_3, which is reset when entering state *crossing*. Notice that time passes only when the automaton is in a self-loop, since all other transitions reset a clock (and hence are zero-time by definition).

The behavior of the railroad crossing, composed of a controller, two sensors and a gate is shown by the nondeterministic timed automaton whose transition graph is given in Fig. 3. At start, the gate is *open* and no train is inside the critical region. When a train is approaching the gate, a sensor, placed at a suitable distance from the railroad crossing, determines the train's entering the critical region. The gate is then closed by the controller. The closing action takes a certain time, slightly variable with the particular plant (between 40 and 50 time units in the figure, as measured by clock x_1, which is reset when entering state *closing*). By construction the distance of the sensor from the gate is such that the fastest train cannot cross before the gate is completely closed, allowing the system to be safe.

Another sensor determines when the last car of the train leaves the crossing, and correspondingly the gate is opened. The action of opening the gate takes again a slightly variable delay depending on the plant (between 40 and 50 time units in the figure, as measured by clock x_3, which is reset when entering state *opening*).

The two above timed automata may be combined in a unique synchronous model M, with one global clock, by taking the cartesian product of the two automata and synchronizing a few transitions: the transition from *open* to *closing* of the controller must be synchronized with the transition from *far* to *approaching* of the train model; the transition from *closed* to *opening* of the controller must be synchronized with the transition from *crossing* to *far* of the train model (in both cases, we suppose that sensors react instantaneously). Hence, not every pair of states is possible in M.

Typical properties studied for a railroad crossing problem are *safety* and *utility*. Safety means that it never happens that the train is in the state *crossing* while the controller is not in the state *closed*.

Utility is a kind of liveness property, formalizable in different ways, representing the intuitive fact that the crossing must be crossed also by cars and people, without making them wait longer than is necessary. Notice that, in a sense, utility has the same level of importance as safety. For example, safety could be easily verified by building a gate that is always closed, but obviously this would be considered useless.

The kind of utility property we are considering here is a constraint on the total time T_{closed} that a gate is not open compared with the time T_{train} that the corresponding train spends inside the critical region. More precisely, we want to state that T_{closed} must not exceed T_{train} by more than 20%, which may be acceptable as a safety margin.

In what follows, α, β represent configurations. Let $I(\alpha)$ be a formula stating that α is the initial configuration, i.e. a configuration where M is in the initial state (*far*, *open*) and each clock is set to 0. Let $state_{\text{train}}(\beta)$ and $state_{\text{controller}}(\beta)$ be, respectively, the train component and the controller component of the state of M in configuration β; Let $P(\beta)$ be the following Presburger formula:

$$state_{\text{train}}(\beta) = far \wedge state_{\text{controller}}(\beta) = open \rightarrow x_1 - x_4 \leqslant 1.2(t_2 - t_1).$$

The term $x_1 - x_4$, when in state *open*, measures exactly the time T_{closed} spent by the controller in state different from *open* during the arrival of the last train. Similarly, $t_2 - t_1$, when in state *far*, measures exactly the total time T_{train} spent by the last train inside the critical region. Notice that $x_1 - x_4 \leqslant 1.2(t_2 - t_1)$ in P is not a clock region (because of the ratio 1.2).

Utility is an instance of the $\forall$-Presburger-always problem of M concerning I and P, since it must always be true during the (supposedly infinite) lifetime of the system. The problem is decidable as we have shown.

6. Discussions and future work

It is important to provide a uniform framework to clarify what kind of temporal Presburger properties can be automatically checked for timed automata. Given a *dta*

$\mathscr{A}$, the set of linear temporal logic formulas $\mathscr{L}_A$ with respect to $\mathscr{A}$ is defined by the following grammar:

$$\phi := P|\neg\phi|\phi \wedge \phi|\bigcirc\phi|\phi U\phi,$$

where P is a Presburger-definable set of configurations of $\mathscr{A}$, $\bigcirc$ denotes "next", and U denotes "until". Formulas in $\mathscr{L}_A$ are interpreted on ω-sequences p of configurations of $\mathscr{A}$ in the usual way. We use p^i to denote the ω-sequence resulting from the deletion of the first i configurations from p. We use p_i to indicate the ith element in p. The satisfiability relation $\models$ is recursively defined as follows, for each ω-sequence p and for each formula $\phi \in \mathscr{L}_A$ (written $p \models \phi$):

$p \models P$ if $p_1 \in P$,
$p \models \neg\phi$ if not $p \models \phi$,
$p \models \phi_1 \wedge \phi_2$ if $p \models \phi_1$ and $p \models \phi_2$,
$p \models \bigcirc\phi$ if $p^1 \models \phi$,
$p \models \phi_1 U \phi_2$ if $\exists j(p^j \models \phi_2$ and $\forall k < j(p^k \models \phi_1))$.

where the variables i, j, k range over $\mathbf{N}$.

We adopt the convention that $\Diamond\phi$ (eventual) abbreviates $(trueU\phi)$ and $\Box\phi$ (always) abbreviates $(\neg\Diamond\neg\phi)$.

Given $\mathscr{A}$ and a formula $\phi \in \mathscr{L}_A$, the model-checking problem is to check whether each ω-path p of $\mathscr{A}$ satisfies $p \models \phi$. The satisfiability-checking problem is to check whether there is an ω-path p of $\mathscr{A}$ satisfying $p \models \phi$. It is obvious that the model-checking problem on the formula ϕ is equivalent to the negation of the satisfiability-checking problem on the formula $\neg\phi$ (and thus it is its "dual").

The results of this paper show that:

- The satisfiability-checking problem is decidable for formulas in $\mathscr{L}_A$ in the form $I \wedge \Box\Diamond P$ and $I \wedge \Diamond P$, where I and P are Presburger sets of configurations.
- The model-checking problem is undecidable for formulas in $\mathscr{L}_A$, even when the formulas are in the form $\Box\Diamond P$ and $\Diamond P$.
- Therefore, both the satisfiability-checking problem and the model-checking problem are undecidable for the entire $\mathscr{L}_A$, even when the "next" operator $\bigcirc$ is excluded from the logic $\mathscr{L}_A$.

Future work may include investigating a fragment of $\mathscr{L}_A$ that has a decidable satisfiability-checking/model-checking problem following the work of Comon and Cortier [9]. There are problems that are still open. For instance, we do not know whether the satisfiability-checking problem is decidable for $I \wedge \Box\Diamond P \wedge \Box\Diamond Q$ (i.e., find an ω-path that is both P-i.o. and Q-i.o).

In [6], an extension of TPTL, called Presburger TPTL, is proposed and it is shown to be undecidable for discrete time. The proof in [6] does not imply (at least, not in an obvious way) the undecidability of the $\forall$-Presburger-i.o. problem and the $\forall$-Presburger-eventual problem in this paper. In that proof, the semantics of Presburger TPTL (over discrete time domain) is interpreted on timed state sequences (i.e., an infinite sequence of tuples of a control state and of a timestamp satisfying monotonicity and progress conditions [6]). TPTL provides $\bigcirc$ and U, as well as freeze quantifiers. Using a freeze quantifier, a timestamp can be "remembered". In this way, the transition relation of a two-counter machine can be encoded into Presburger TPTL by using $\bigcirc, U$ and

the freeze quantifier. This gives the undecidability of the logic [6]. On the other hand, $\Box\Diamond P$ and $\Diamond P$ in this paper are interpreted on sequences of configurations (in contrast to timed state sequences). Formulas like $\Box\Diamond P$ and $\Diamond P$ are state formulas. That is, without using $\bigcirc$ and without introducing freeze quantifiers, we have no way to remember clock values in one configuration and use them to compare those in another configuration along p. Therefore, the transition relation of a two-counter machine cannot be encoded in our logic $\mathscr{L}_A$. But we are able to show in this paper that computations of a two-counter machine can be encoded by ω-paths, restricted under $\Box\Diamond P$ or $\Diamond P$, of a *dta*, leading to the undecidability results of the paper.

We are also interested in considering the same set of liveness problems for a dense time domain. However, special attention must be paid in order to define the concepts of "infinitely often" and ω-paths under a dense time domain. We believe that the decidability results (for the $\exists$-Presburger-i.o. problem and the $\exists$-Presburger-eventual problem) also hold for dense time when the semantics of a timed automaton are carefully defined. We may look at Comon and Jurski's flattening construction [10] in arguing about the binary reachability of timed automata over a dense domain. We may also look at the recent generalization [11] of the work in [12] to the dense time domain and the dissertation of Dima [14], which addresses the issue of discretizing timed automata with dense clocks. On the other hand, the undecidability results in this paper can be naturally extended to the dense time domain when the ω-paths in this paper are properly redefined for dense time.

Acknowledgements

Thanks go to the anonymous referees for many useful suggestions.

Appendix A. Proof of Lemma 9

Let M be a deterministic counter machine. M has a finite number of instructions with each instruction chosen from the following three types (x is one of the counters $x_1,\ldots,x_k$):

- (addition)

  $$l_i \rightarrow^{x:=x+1} l_j.$$

  M transits from location l_i to location l_j with counter x incremented by 1 and the other counters unchanged.
- (subtraction)

  $$l_i \rightarrow^{x:=x-1} l_j.$$

  M transits from location l_i to location l_j (if x is positive) with counter x decremented by 1 and the other counters unchanged.

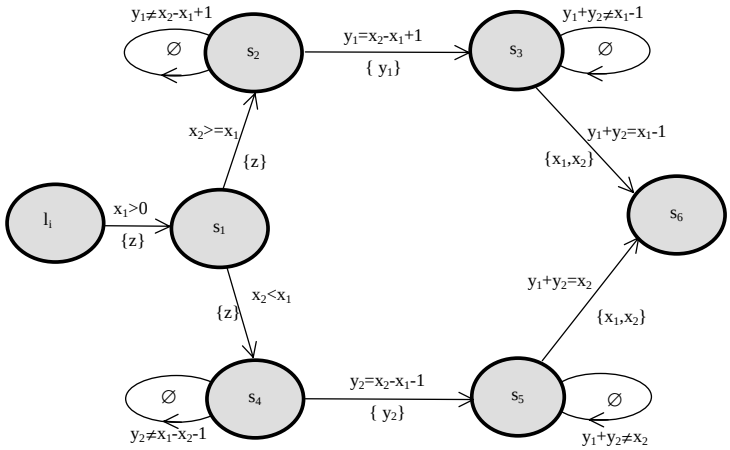

Fig. 4. The generalized *dta* implementing the mapping from $\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$ to $\langle x_1 = 0, x_2 = 0, y_1 = a_1 - 1, y_2 = a_2\rangle$ for all nonnegative integers a_1 and a_2.

- (testing against zero)

$$l_i \rightarrow^{x=0} l_j.$$

M transits from location l_i to location l_j if counter x is zero.

We construct a generalized *dta* to simulate each instruction in M. Without loss of generality, we consider only the instruction $l_i \rightarrow^{x_1 := x_1 - 1} l_j$. In order to simulate it, we regard the two counters x_1 and x_2 as two clocks and also introduce two other clocks y_1 and y_2, and a dummy clock z. Define two generalized *dta*s (illustrated in Figs. 4 and 5), whose sequential composition simulates the above instruction as follows, for all nonnegative integers a_1 and a_2:

- the first automaton (Fig. 4), when in state l_i with clock values $\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$ and $a_1 > 0$, sends the clock values to $\langle x_1 = 0, x_2 = 0, y_1 = a_1 - 1, y_2 = a_2\rangle$, and goes, through a sequence of steps, into a state denoted by s_6.
- the second automaton (Fig. 5), when in state s_6, sends the clock values $\langle x_1 = 0, x_2 = 0, y_1 = a_1, y_2 = a_2\rangle$ to $\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$, ending at state l_j.

Let us see how the automaton in Fig. 4 works. Given

$$\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$$

at state l_i, if $a_2 \geqslant a_1$, the automaton walks along the path s_1, s_2, s_3, ending at s_6. After the loop at s_2, the clock values are

$$\langle x_1 = a_2 + 1, x_2 = 2a_2 - a_1 + 1, y_1 = a_2 - a_1 + 1, y_2 = a_2 - a_1 + 1\rangle.$$

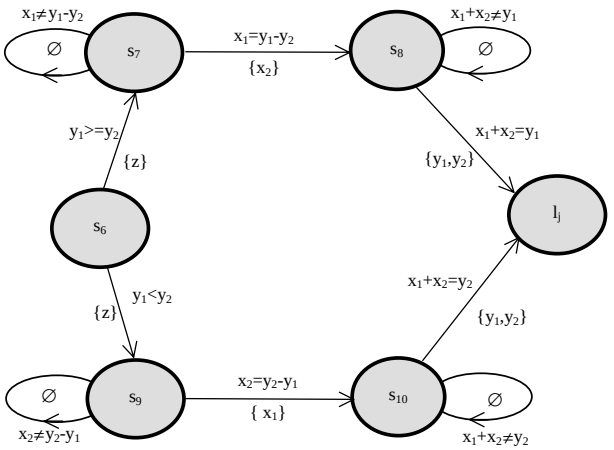

Fig. 5. The generalized *dta* implementing the mapping from $\langle x_1 = 0, x_2 = 0, y_1 = a_1, y_2 = a_2\rangle$ to $\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$ for all nonnegative integers a_1 and a_2.

After firing the edge from s_2 to s_3, the clock values are

$$\langle x_1 = a_2 + 1, x_2 = 2a_2 - a_1 + 1, y_1 = 0, y_2 = a_2 - a_1 + 1\rangle.$$

It is easy to check that the loop at s_3 will be fired exactly $a_1 - 1$ times. When the automaton exits the loop, the clock values are

$$\langle x_1 = a_2 + a_1, x_2 = 2a_2, y_1 = a_1 - 1, y_2 = a_2\rangle.$$

After x_1 and x_2 are reset on the edge from s_3 to s_6, the clock values are

$$\langle x_1 = 0, x_2 = 0, y_1 = a_1 - 1, y_2 = a_2\rangle,$$

which is exactly what we expected. It is also easy to check the case when $a_2 < a_1$ and the automaton walks along the path s_1, s_4, s_5 and ending at s_6.

Similarly, it can be argued that the automaton shown in Fig. 5 implements the mapping from

$$\langle x_1 = 0, x_2 = 0, y_1 = a_1, y_2 = a_2\rangle$$

to

$$\langle x_1 = a_1, x_2 = a_2, y_1 = 0, y_2 = 0\rangle$$

for all nonnegative integers a_1 and a_2.

The sequential composition of the two automata results in a generalized *dta* implementing the instruction $l_i \rightarrow^{x_1 := x_1 - 1} l_j$. Similar constructions can be worked out to implement the other two kinds of instructions for the counter machine, using generalized *dta*s. Therefore, a generalized discrete timed automaton $\mathscr{A}$ can now be constructed

to implement the entire counter machine M. The construction can be extended to implement counter machines with more than two counters. It should be noted that if M is deterministic, the resulting automaton $\mathscr{A}$ is also deterministic. □

References

[1] R. Alur, Timed automata, in: CAV'99, Lecture Notes in Computer Science, Vol. 1633, Springer, Berlin, 1999, pp. 8–22.

[2] R. Alur, C. Courcoubetis, D. Dill, Model-checking in dense real time, Inform. Comput. 104 (1) (1993) 2–34.

[3] R. Alur, D. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–236.

[4] R. Alur, T. Feder, T.A. Henzinger, The benefits of relaxing punctuality, J. ACM 43 (1) (1996) 116–146.

[5] R. Alur, T.A. Henzinger, Real-time logics: complexity and expressiveness, Inform. Comput. 104 (1) (1993) 35–77.

[6] R. Alur, T.A. Henzinger, A really temporal logic, J. ACM 41 (1) (1994) 181–204.

[7] A. Bouajjani, S. Tripakis, S. Yovine, On-the-fly symbolic model-checking for real-time systems, in: Proc. 18th Real Time Systems Symposium (RTSS'97), IEEE Computer Society Press, Silver Spring, MD, 1997, pp. 25–35.

[8] A. Coen-Porisini, C. Ghezzi, R. Kemmerer, Specification of real-time systems using ASTRAL, IEEE Trans. Software Eng. 23 (9) (1997) 572–598.

[9] H. Comon, V. Cortier, Flatness is not a weakness, in: CSL'00, Lecture Notes in Computer Science, Vol. 1862, Springer, Berlin, 2000, pp. 262–276.

[10] H. Comon, Y. Jurski, Timed automata and the theory of real numbers, in: CONCUR'99, Lecture Notes in Computer Science, Vol. 1664, Springer, Berlin, 1999, pp. 242–257.

[11] Z. Dang, Binary reachability analysis of pushdown timed automata with dense clocks, in: CAV'01, Lecture Notes in Computer Science, Vol. 2102, Springer, Berlin, 2001, pp. 506–517.

[12] Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su, Binary reachability analysis of discrete pushdown timed automata, in: CAV'00, Lecture Notes in Computer Science, Vol. 1855, Springer, Berlin, 2000, pp. 69–84.

[13] Z. Dang, P. San Pietro, R.A. Kemmerer, On Presburger liveness of discrete timed automata, in: STACS'01, Lecture Notes in Computer Science, Vol. 2010, Springer, Berlin, 2001, pp. 132–143.

[14] C. Dima, An algebraic theory of real-time formal languages, Ph.D. Dissertation, Verimag, Grenoble, 2001.

[15] C. Heitmeyer, N. Lynch, The generalized railroad crossing: a case study in formal verification of real-time systems, in: Proc. 15th Real-time Systems Symposium (RTSS'94), IEEE Computer Society Press, Silver Spring, MD, 1994, pp. 120–131.

[16] T. A. Henzinger, Z. Manna, A. Pnueli, What good are digital clocks?, in: ICALP'92, Lecture Notes in Computer Science, Vol. 623, Springer, Berlin, 1992, pp. 545–558.

[17] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, Symbolic model checking for real-time systems, Inform. Comput. 111 (2) (1994) 193–244.

[18] T.A. Henzinger, Pei-Hsin Ho, HyTech: the Cornell hybrid technology tool, in: Hybrid Systems II, Lecture Notes in Computer Science, Vol. 999, Springer, Berlin, 1995, pp. 265–294.

[19] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. ACM 25 (1) (1978) 116–133.

[20] F. Laroussinie, K.G. Larsen, C. Weise, From timed automata to logic—and back, in: MFCS'95, Lecture Notes in Computer Science, Vol. 969, Springer, Berlin, 1995, pp. 529–539.

[21] K.G. Larsen, P. Pattersson, W. Yi, UPPAAL in a nutshell, Internat. J. Software Tools Technol. Transfer 1 (1–2) (1997) 134–152.

[22] J. Raskin, P. Schobben, State clock logic: a decidable real-time logic, in: HART'97, Lecture Notes in Computer Science, Vol. 1201, Springer, Berlin, 1997, pp. 33–47.

[23] V. Weispfenning, The complexity of almost linear Diophantine problems, J. Symb. Comp. 10 (5) (1990) 395–403.
[24] T. Wilke, Specifying timed state sequences in powerful decidable logics and timed automata, Lecture Notes in Computer Science, Vol. 863, Springer, Berlin, 1994, pp. 694–715.
[25] S. Yovine, Kronos: a verification tool for real-time systems, Internat. J. Software Tools Technol. Transfer 1 (1–2) (1997) 123–133.
[26] S. Yovine, Model checking timed automata, in: Embedded Systems, Lecture Notes in Computer Science, Vol. 1494, Springer, Berlin, 1998, pp. 114–152.