


# A Unified Model for Real-Time Systems: Symbolic Techniques and Implementation

S Akshay ✉ 

Department of CSE, Indian Institute of Technology Bombay, Mumbai, India

Paul Gastin ✉ 


Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, 91190, Gif-sur-Yvette, France  
CNRS, ReLaX, IRL 2000, Siruseri, India

R Govind ✉ 

Department of CSE, Indian Institute of Technology Bombay, Mumbai, India

Aniruddha R Joshi ✉ 

Department of CSE, Indian Institute of Technology Bombay, Mumbai, India

B Srivathsan ✉ 

Chennai Mathematical Institute, India  
CNRS, ReLaX, IRL 2000, Siruseri, India

---

## Abstract

In this paper, we consider a model of *generalized timed automata (GTA)* with two kinds of clocks, *history and future*, that can express many timed features succinctly, including timed automata, event-clock automata with and without diagonal constraints, and automata with timers.

Our main contribution is a new simulation-based zone algorithm for checking reachability in this unified model. While such algorithms are known to exist for timed automata, and have recently been shown for event-clock automata without diagonal constraints, this is the first result that can handle event-clock automata with diagonal constraints and automata with timers. We also provide a prototype implementation for our model and show experimental results on several benchmarks. To the best of our knowledge, this is the first effective implementation not just for our unified model, but even just for automata with timers or for event-clock automata (with predicting clocks) without going through a costly translation via timed automata. Last but not least, beyond being interesting in their own right, generalized timed automata can be used for model-checking event-clock specifications over timed automata models.

**2012 ACM Subject Classification** Theory of computation → Timed and hybrid models; Theory of computation → Quantitative automata; Theory of computation → Logic and verification

**Keywords and phrases** Real-time systems, Timed automata, Event-clock automata, Clocks, Timers, Verification, Zones, Simulations, Reachability

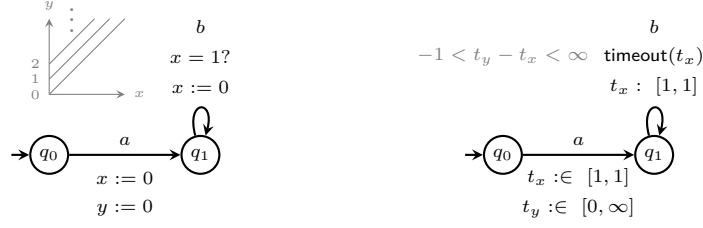
**Funding** This work was supported by DST/CEFIPRA/INRIA Project EQuaVE.

*S Akshay:* Supported in part by DST/SERB Matrics Grant MTR/2018/000744.

*Paul Gastin:* Partially supported by ANR project Ticktac (ANR-18-CE40-0015).

## 1 Introduction

The idea of adding real-time dynamics to formal verification models started as a hot topic of research in the 1980s [21, 5]. Over the years, timed automata [7, 8] has emerged as a leading model for finite-state concurrent systems with real-time constraints. Timed automata make use of *clocks*, real-valued variables which increase along with time. Constraints over clock values can be used as guards for transitions, and clocks can be reset to 0 along transitions. It is notable that *the early works in this area made use of timers* to deal with real-time [34, 22, 11]. Timers are started by setting them to some initial value within a given interval. Their values decrease with time, and an *timeout* event can be used in transitions



■ **Figure 1** An automaton with clocks on left, and timers on right for same constraints.

to detect the instant when the timers become 0. Quoting from [5], the shift from timers to clocks in timed automata, as we know them today, is attributed to the fact that: “*apart from some technical conveniences in developing the emptiness algorithm and proving its correctness, the reformulation allows a simple syntactic characterization of determinism for timed automata*”. Over the last thirty years, the study of timed automata has led to the development of rich theory and industry-strength verification tools. The use of clocks has also allowed for the extension of the model to more complex constraints and assignments to clocks in transitions [12, 15]. Furthermore, considering more sophisticated rates of evolution for clocks gives the yet another well-established model of hybrid automata [6].

When it comes to the reachability problem, timers do have some nice properties. Let us explain with an example. Figure 1 shows a timed automaton on the left, and an automaton with timers on the right, for the set of words  $ab^*$  such that the time between every consecutive letters is 1. The timed automaton sets clock  $x$  to 0 and checks for the guard  $x = 1?$  to enforce the timing constraint. The automaton with timers, on the right, sets a timer  $t_x$  to 1, and asks for its expiry in the immediate next action. Clock  $y$  and timer  $t_y$  are not necessary for the required timing property, but we add them to illustrate a different aspect that we will describe now. To solve the reachability problem, a symbolic enumeration of the state space is performed. In the timed automaton, at state  $q_1$ , the enumeration gives constraints  $y - x = n$  for every  $n \geq 0$ . Starting from  $y - x = n$  and executing  $b$  gives  $y - x = n + 1$ , due to the combination of guard  $x = 1?$  and reset  $x := 0$ . This shows that a naïve symbolic enumeration is not bound to terminate. The question of developing finite abstractions for timed automata has been a central problem of study which started in the late 90s and continues till date (see recent surveys [16, 40]). Such an issue does not occur with timers. In the automaton with timers on the right,  $t_x$  is set to 1 and  $t_y$  is set to some arbitrary value in the transition to  $q_1$ . This gives  $-1 \leq t_y - t_x \leq \infty$  for the set of all possible timer values. When  $t_x$  times out, the value of  $t_y$  could still be any value from 0 to  $\infty$ . When  $t_x$  is set to 1 again, the set of possible timer values still satisfies the same constraint  $-1 \leq t_y - t_x \leq \infty$  leading to a fixed point with a finite reachable state space. The fact that **symbolic enumeration terminates on an automaton with timers** was already observed in [22]. To our knowledge, later works on timed automata reachability never went back to timers, and there is no tool support that we know of to deal with models with timers directly. We find this surprising given that timers occur naturally while modeling real-time systems and moreover they enjoy this finiteness property.

In addition to clocks and timers, *event-clocks* are another special type of clock variables that are used to deal with timing constraints [9], which are attached to events. An event-recording clock for event  $a$  maintains the time since the previous occurrence of  $a$ , whereas an event-predicting clock for  $a$  gives the time to the next occurrence of  $a$ . Event-clocks have been used in the model of event-clock automata (ECA), and also in the logic of event-clocks [38]. These works argue that event-clocks can express typical real-time requirements. Theoretically,

ECA can be determinized, and hence complemented. Therefore, model-checking an event-clock (logic or automaton) specification  $\varphi$  over a timed automaton  $\mathcal{A}$  can be reduced to reachability on the product of  $\mathcal{A}$  and the ECA for  $\neg\varphi$ . This makes event-clocks a convenient feature in specifications.

Recently, a symbolic enumeration algorithm for ECA was proposed [2]. It was noticed that when restricted to event-predicting clocks, the symbolic enumeration terminates without any additional checks (similar to the case of timers), whereas for the combination involving event-recording clocks, one needs simulation techniques from the timed automata literature. The same work showed how to adapt the best known simulation technique from timed automata into the setting of ECA. However, as discussed above, for model-checking we need a model containing both conventional clocks, timers and event-clocks. To our knowledge, no tool can directly work on such models.

Our goal in this work is to provide a one stop solution to real-time verification, be it reachability analysis or model-checking (over event-clock specifications), be it using models with clocks, or models with timers. We consider a unified model of a timed automaton over variables that can simulate normal clocks, timers and event-clocks. Here are our key contributions:

1. We define a new model of generalized timed automata (GTA) which have two types of variables, called *history* clocks and *future* clocks. **History clocks generalize normal clocks as well as event-recording clocks, while future clocks generalize event-predicting clocks and timers.** However, unlike event-clocks, clocks in GTA are not necessarily associated with events. We also consider a generic syntax that allows for diagonal constraints between variables.
2. **We show undecidability of reachability for GTA,** and study a **safe subclass** that makes the model decidable. Safe GTA already subsume timed automata, event-clock automata (with diagonal constraints) and automata with timers.
3. We adapt state-of-the-art symbolic enumeration techniques from timed automata literature to safe GTA. While we make use of ideas presented in [22] and [2], these works do not contain diagonal constraints between variables. Our main technical and theoretical innovation lies in a new termination analysis of the symbolic enumeration in the presence of diagonal constraints. Surprisingly, we show that **the enumeration terminates as long as the diagonal constraints are restricted to usual clocks and event-clocks, but not timers.**
4. We develop a prototype implementation of our model and algorithm in TCHECKER, an open-source platform for timed automata analysis, and show promising results on several existing and new benchmarks. To the best of our knowledge, our tool is the first that can handle event-clock automata, a model that till date has been the subject of many theoretical results.

*Related works.* In the work that first introduced ECA, a translation from ECA to a timed automaton was also proposed. However, this translation is not efficient: in the worst case, this translation incurs a blowup in the number of clocks and states. In [27, 28], an extrapolation approach using maximal constants has been studied for ECA. However, it has been observed that simulation-based techniques are both more effective [12, 14] and efficient [24, 25, 26, 4] than extrapolation for checking reachability. Recently, [2] proposed a zone-based reachability algorithm for diagonal-free ECA, using simulations for finiteness, but there was no accompanying implementation. Diagonal constraints have long been known to allow succinct modeling [13] for the class of timed-automata, but only recently a zone-based algorithm that directly works on such automata, was proposed. ECA with diagonals are more expressive than ECA [17]. In this work, we propose a zone-based algorithm for a unified

model that subsumes ECA with diagonals.

The use of history clocks and prophecy clocks in ECAs is in the same spirit as past and future modalities in temporal logics - this makes ECAs an attractive model for writing timed specifications. Indeed, this has also led to a development of various temporal logics with event-clocks [23, 1, 38]. ECA with diagonal constraints have been well-studied, such as in the context of timeline based planning [17, 18]. Finally, while there has been substantial advances in the theory of ECA, to the best of our knowledge, the only tool that handles ECA is TEMPO [39], and even this tool is restricted to just history clocks.

*Structure of the paper.* In Section 2 we start by defining the generalized model. Section 3 examines its expressiveness, while Section 4 deals with the reachability problem and the safe subclass. Section 5 develops the symbolic enumeration technique, while Section 6 explains how distance graphs can be extended to this setting. In Section 7, we discuss some of the properties of distance graphs of reachable zones. Section 8 is dedicated to finiteness. Finally, we provide our experimental results in Section 9 and conclude with Section 10.

## 2 Generalized Timed Automata

In this section we introduce the unified model. While we build on classical ideas from timed automata, almost every aspect is extended and below we highlight these changes.

### 2.1 Extending clocks and constraints

We define  $X = X_H \uplus X_F$  to be a finite set of real-valued variables called *clocks*, where  $X_H$  is the set of *history clocks*, and  $X_F$  is the set of *future clocks*.

Let  $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$  denote the set of all real numbers along with  $-\infty$  and  $+\infty$ . The usual  $<$  order on reals is extended to deal with  $\{-\infty, +\infty\}$  as:  $-\infty < c < +\infty$  for all  $c \in \mathbb{R}$  and  $-\infty < \infty$ . Similarly,  $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$  denotes the set of all integers along with  $-\infty$  and  $+\infty$ . Let  $\mathbb{R}_{\geq 0}$  (resp.  $\mathbb{R}_{\leq 0}$ ) be the set of non-negative (resp. non-positive) reals.

► **Definition 1 (Weights).** Let  $\mathcal{C} = \{(\triangleleft, c) \mid c \in \overline{\mathbb{R}} \text{ and } \triangleleft \in \{\leq, <\}\}$ , called the set of weights.

Let  $X \cup \{0\}$  be the set obtained by extending the clocks of GTA with the special constant clock 0. Note that this clock will always have the value 0. Let  $\Phi(X)$  denote a set of clock constraints generated by the following grammar:

$$\varphi ::= x - y \triangleleft c \mid \varphi \wedge \varphi$$

where  $x, y \in X \cup \{0\}$ ,  $(\triangleleft, c) \in \mathcal{C}$  and  $c \in \overline{\mathbb{Z}}$ . The introduction of the special constant clock 0 allows us to treat constraints with just a single clock as special cases. Note that the constraint  $x \triangleleft c$  is equivalent to  $x - 0 \triangleleft c$  and the constraint  $c \triangleleft x$  is equivalent to  $0 - x \triangleleft -c$ . We often write  $x = c$  as a shorthand for  $x \leq c \wedge c \leq x$ . The base constraints of the form  $x - y \triangleleft c$  will be called *atomic constraints*. The atomic constraints of the form  $x - y \triangleleft c$  such that  $x \neq 0 \neq y$  are called *diagonal constraints*, and all other atomic constraints are called *non-diagonal constraints*.

### 2.2 Extending valuations

We first recall and discuss the *extended algebra* of weights that was proposed in a recent work on event-clock automata [2, 3]. To evaluate the constraints allowed by  $\Phi(X)$ , we extend addition on real numbers with the convention that  $(+\infty) + \alpha = \alpha + (+\infty) = +\infty$  for all

$\alpha \in \overline{\mathbb{R}}$  and  $(-\infty) + \beta = \beta + (-\infty) = -\infty$ , as long as  $\beta \neq +\infty$ . We also extend the unary minus operation from real numbers to  $\overline{\mathbb{R}}$  by setting  $-(+\infty) = -\infty$  and  $-(-\infty) = +\infty$ . Abusing notation, we write  $\beta - \alpha$  for  $\beta + (-\alpha)$ . Notice that with this definition of extended addition, the minus operation does not distribute over addition.<sup>1</sup>

We first highlight a few more important features of the definition of extended addition operation.

► **Remark 2.** [3] This extended addition has the following properties that are easy to check:

1.  $(\overline{\mathbb{R}}, +, 0)$  is a monoid with 0 as neutral element. In particular, the extended addition is associative.
2.  $(\overline{\mathbb{R}}, +, 0)$  is not a group, since  $-\infty$  and  $+\infty$  have no opposite values. Note that,  $\alpha + (-\alpha) = 0$  when  $\alpha \in \mathbb{R}$  is finite but  $\alpha + (-\alpha) = +\infty$  when  $\alpha \in \{-\infty, +\infty\}$ . As a consequence, in an equation  $\alpha + \beta = \alpha + \gamma$ , we can cancel  $\alpha$  and deduce  $\beta = \gamma$  when  $\alpha$  is finite, but not when  $\alpha$  is infinite.
3. The order  $\leq$  is monotone on  $\overline{\mathbb{R}}$ :  $b \leq c$  implies  $a + b \leq a + c$ , but the converse implication only holds when  $a$  is finite.
4. The strict order  $<$  is only monotone with respect to finite values: when  $a$  is finite,  $b < c$  iff  $a + b < a + c$ .
5. For all  $a, b \in \overline{\mathbb{R}}$  and  $(\triangleleft, c) \in \mathcal{C}$ , we have  $a \triangleleft b$  iff  $-b \triangleleft -a$ . Further,  $a - b \triangleleft c$  implies  $a \triangleleft b + c$ . The converse of the latter statement holds when  $b$  is finite. Note that the converse may be false when  $b$  is infinite.<sup>2</sup>

► **Definition 3 (Valuation).** A valuation of clocks is a function  $v: X \cup \{0\} \mapsto \overline{\mathbb{R}}$  which maps the special clock 0 to 0, history clocks to  $\mathbb{R}_{\geq 0} \cup \{+\infty\}$  and future clocks to  $\mathbb{R}_{\leq 0} \cup \{-\infty\}$ . We denote by  $\mathbb{V}(X)$  or simply by  $\mathbb{V}$  the set of valuations over  $X$ . We say that clock  $x$  is defined (resp. undefined) in  $v$  when  $v(x) \in \mathbb{R}$  (resp.  $v(x) \in \{-\infty, +\infty\}$ ).

► **Definition 4.** Let  $x, y \in X \cup \{0\}$  be clocks (including 0) and let  $(\triangleleft, c)$  be a weight. For valuations  $v \in \mathbb{V}$ , define  $v \models y - x \triangleleft c$  as  $v(y) - v(x) \triangleleft c$ . We say that a valuation  $v$  satisfies a constraint  $\varphi$ , denoted as  $v \models \varphi$ , when  $v$  satisfies all atomic constraints in  $\varphi$ .

► **Remark 5.** From Definition 4, we easily check that the constraint  $y - x \triangleleft c$  is equivalent to *true* (resp. *false*) when  $(\triangleleft, c) = (\leq, +\infty)$  (resp.  $(\triangleleft, c) = (<, -\infty)$ ). Constraints that are equivalent to *true* or *false* will be called trivial, whereas all others are non-trivial constraints.

If  $(\triangleleft, c) \neq (\leq, +\infty)$  then  $v \models y - x \triangleleft c$  never holds when  $v(x) = -\infty$ .

Also, if  $v(x) = v(y) \in \{-\infty, +\infty\}$  then  $v \models y - x \triangleleft c$  only holds for  $(\triangleleft, c) = (\leq, +\infty)$ .

For a non-trivial constraint  $y - x \triangleleft c$ , i.e.,  $(\triangleleft, c) \notin \{(<, -\infty), (\leq, +\infty)\}$ , we have

- $v \models y - x \triangleleft c$  iff  $v(y) < +\infty = v(x)$  or  $(v(x) \text{ is finite and } v(y) \triangleleft v(x) + c)$ .
- $v \models y - x \leq -\infty$  iff  $v(y) < +\infty = v(x)$  or  $v(y) = -\infty < v(x)$ .
- $v \models y - x < +\infty$  iff  $v(x) \neq -\infty$  and  $v(y) \neq +\infty$ . ◀

We abuse notation and for  $Y \subseteq X$ , we define  $Y \triangleleft c$  as  $\bigwedge_{y \in Y} y \triangleleft c$ , and  $Y = c$  as  $\bigwedge_{y \in Y} y = c$ .

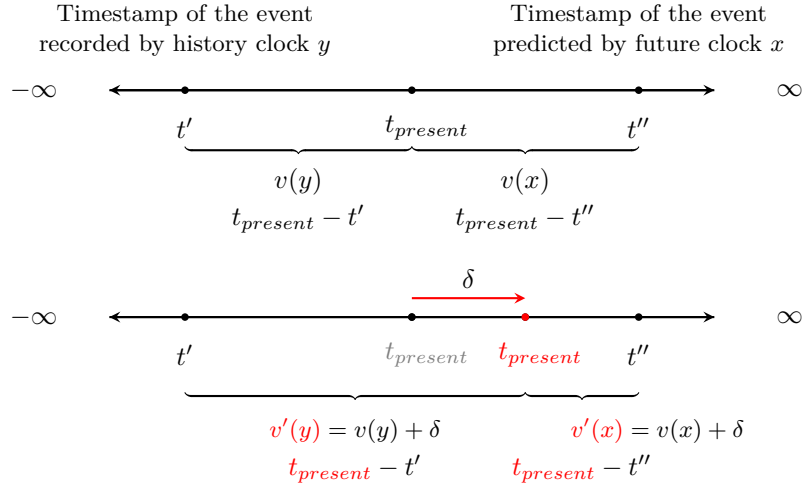
We denote by  $v + \delta$  the valuation obtained from valuation  $v$  by increasing by  $\delta \in \mathbb{R}_{\geq 0}$  the value of all clocks in  $X$ . Note that, from a given valuation, not all time elapse result in

<sup>1</sup> Notice that  $-(a + b) = (-a) + (-b)$  when  $a$  or  $b$  is finite or when  $a = b$ . But, when  $a = +\infty$  and  $b = -\infty$  then  $-(a + b) = -\infty$  whereas  $(-a) + (-b) = +\infty$ .

<sup>2</sup> For instance, if  $a < +\infty = b$  then  $a < b + (-\infty)$ , but  $a - b = -\infty \not< -\infty$ .

If  $a < +\infty$  and  $b = -\infty$  then  $a < b + \infty$ , but  $a - b = +\infty \not< +\infty$ .

If  $a = b \in \{-\infty, +\infty\}$  and  $c$  is finite then  $a \leq b + c$ , but  $a - b = +\infty \not\leq c$ .



■ **Figure 2** Representation of valuations in generalized timed automata. Here,  $v' = v + \delta$ .

valuations since future clocks need to stay at most 0. For example, from a valuation with  $v(x) = -3$  and  $v(y) = -2$ , where  $x, y$  are future clocks, one can elapse at most 2 time units.

### 2.3 Extending resets

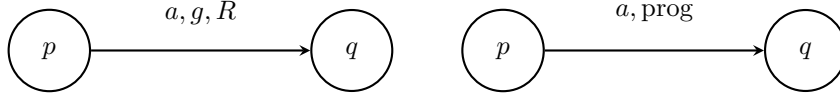
For history clocks, the reset operation sets the clock to 0. For future clocks, the reset operation says that all constraints on the clock must be discarded, i.e., the clock is *released*. Given that the set of clocks is partitioned into history clocks and future clocks, we use the same notation  $[R]v$  to talk about the change of clocks in  $R$ , whether it be reset/release, which operates differently depending on whether the clock is a history or future clock. More precisely, given a set of clocks  $R \subseteq X$ , we define  $R_F = R \cap X_F$  as the set of future clocks in  $R$  and  $R_H = R \cap X_H$  as the set of history clocks in  $R$ . We then define  $[R]v$  as follows:

$$[R]v := \{v' \in \mathbb{V} \mid v'(x) = 0 \ \forall x \in R_H \text{ and } v'(x) = v(x) \ \forall x \notin R\}$$

Observe that, *the release operation* is implicit: each future clock in  $R$  could take any value (not necessarily the same) from  $[-\infty, 0]$  in  $[R]v$ . Note that  $[R]v$  is a singleton when  $R$  contains only history clocks - this corresponds exactly to the reset operation in timed automata. In this case, we simply write  $v' = [R]v$  instead of  $\{v'\} = [R]v$ . When  $R$  contains only future clocks,  $[R]v$  is the set of valuations obtained by releasing each clock in  $R$  (i.e., setting each clock in  $R$  non-deterministically to some value in  $[-\infty, 0]$ , while keeping the value of other clocks unchanged). For  $W \subseteq \mathbb{V}$ , we let  $[R]W = \bigcup_{v \in W} [R]v$ . We have  $[R' \cup R'']W = [R']([R'']W)$ .

### 2.4 Extending guards and transitions

Before we define GTA, let us focus on the language to specify transitions. In normal timed automata, as shown in Figure 3, a transition reads a letter, checks a guard and then resets a subset of (history) clocks. The guard  $g$  can capture multiple constraints by allowing a conjunction of atomic constraints and resetting the subset  $R \subseteq X$  of clocks corresponds to resetting them one by one. But in any one transition only a pair of guard, reset is performed and one cannot interleave them.



■ **Figure 3** A transition of TA (left) and of a GTA (right)

We generalize this to our setting with history and future clocks but also to allow arbitrary interleaving of guards and changes<sup>3</sup>. Let us formalize this. An *instantaneous timed program* is generated by the following grammar:

$$\text{prog} := \text{guard} \mid \text{change} \mid \text{prog}; \text{prog} \quad \text{where,} \quad (1)$$

$$\text{guard} = g \in \Phi(X) \quad (2)$$

$$\text{change} = [R] \text{ for some } R \subseteq X \quad (3)$$

While guard and change are atomic programs,  $\text{prog}; \text{prog}$  refers to sequential composition. The set of all programs generated by the above grammar will be denoted *Programs*. Then on a transition, we simply have a pair of letter label and an instantaneous timed program, e.g.,  $(a, \text{prog})$  in Figure 3 (right).

The semantics for programs on a transition must generalize semantics for guards (defined using satisfaction relation  $\models$  above) and resets/release (defined using  $[R]$  above). But there is an obvious difference between these two: a guard may be crossed only if the valuation before the guard satisfies it, whereas a *change* (reset or release) defines a relation between the valuations before and after the change. To capture both in a uniform way, we define the semantics of programs as relations on pairs of valuations. Formally, for  $v, v' \in \mathbb{V}$ ,  $\text{prog} \in \text{Programs}$  we say  $(v, v') \models \text{prog}$ , more conveniently written as  $v \xrightarrow{\text{prog}} v'$ , inductively:

- $v \xrightarrow{g} v'$  if  $v \models g$  and  $v' = v$ ,
- $v \xrightarrow{[R]} v'$  if  $v' \in [R]v$ ,
- $v \xrightarrow{\text{prog}_1; \text{prog}_2} v'$  if  $\exists v'' \in \mathbb{V}$  such that  $v \xrightarrow{\text{prog}_1} v''$  and  $v'' \xrightarrow{\text{prog}_2} v'$ .

## 2.5 Extending the automaton model

Now, we have all the pieces necessary to define our generalized model.

► **Definition 6** (generalized timed automata). A GTA  $\mathcal{A}$  is given by a tuple  $(Q, \Sigma, X, \Delta, (q_0, g_0), (Q_f, g_f))$ , where

- $Q$  is a finite set of states,
- $\Sigma$  is a finite alphabet of actions,
- $X = X_F \uplus X_H$  is a set of clocks partitioned into future and history clocks,
- the initialization condition is a pair comprising of an initial state  $q_0 \in Q$  and an initial guard  $g_0 \in \Phi(X)$  which should be satisfied by initial valuations,
- similarly, the final condition is a pair comprising of a set of final states  $Q_f \subseteq Q$  along with a final guard  $g_f$  that must be satisfied by final valuations,
- $\Delta \subseteq (Q \times \Sigma \times \text{Programs} \times Q)$  is a finite set of transitions.  $\Delta$  contains transitions of the form  $(q, a, \text{prog}, q')$ , where  $q$  is the source state,  $q'$  is the target state,  $a$  is the action triggering the transition, and  $\text{prog}$  is the instantaneous timed program that is executed in sequence (from left to right) while firing the transition.

<sup>3</sup> To model this with a TA one may use a sequence of multiple transitions without delays in-between.



► **Definition 7** (Semantics of GTA). *The semantics of a GTA  $\mathcal{A} = (Q, \Sigma, X, \Delta, (q_0, g_0), (Q_f, g_f))$  is given by a transition system  $\text{TS}_{\mathcal{A}}$  whose states are configurations  $(q, v)$  of  $\mathcal{A}$ , where  $q \in Q$  and  $v \in \mathbb{V}$  is a valuation.*

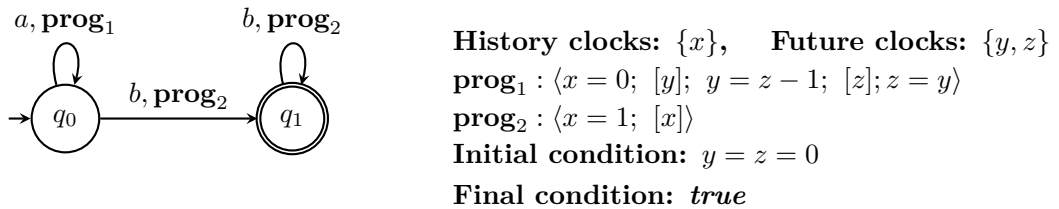
- A configuration  $(q, v)$  is *initial* if  $q = q_0$  and  $v \models g_0$ .
- A configuration  $(q, v)$  is *accepting* if  $q \in Q_f$  and  $v \models g_f$ .
- Transitions of  $\text{TS}_{\mathcal{A}}$  are of two forms:
  - **Delay transition:**  $(q, v) \xrightarrow{\delta} (q, v + \delta)$  if  $(v + \delta) \models X_F \leq 0$ .
  - **Discrete transition:**  $(q, v) \xrightarrow{t} (q', v')$  if  $t = (q, a, \text{prog}, q') \in \Delta$  and  $v \xrightarrow{\text{prog}} v'$ .

Thus, a discrete transition  $t = (q, a, \text{prog}, q')$ , where  $\text{prog} = \text{prog}_1; \dots; \text{prog}_n$  can be taken from  $(q, v)$  if there are valuations  $v_1, \dots, v_n$  such that  $v \xrightarrow{\text{prog}_1} v_1 \xrightarrow{\text{prog}_2} v_2 \dots \xrightarrow{\text{prog}_n} v_n = v'$ .

A *run* of a GTA is a finite sequence of transitions from an initial configuration of  $\text{TS}_{\mathcal{A}}$ . A run is said to be *accepting* if its last configuration is accepting.

### 3 Expressivity of GTA and examples

The GTA model defined above is rather expressive. Figure 4 illustrates an example which accepts words of the form  $a^n b^m$  with  $m \leq n$ , where each  $a$  occurs at time 0, after which  $b$ 's are seen one by one, with distance 1 between them. The history clock  $x$  is used to ensure the timing constraint. For every  $a$  that is read, the future clocks  $y, z$  decrease by 1. Hence the future clocks  $y, z$  maintain the opposite of the number of  $a$ 's seen. When the automaton starts reading  $b$ , the future clocks also start elapsing time and since they cannot go above 0, the number of  $b$ 's is at most the number of  $a$ 's. Such a language cannot be accepted by timed automata since the untimed language obtained by removing the time stamps needs to be regular in the case of timed automata. The GTA model is not only expressive, it is also convenient for use. To see this we now show that three classical models of timed systems can be easily captured using GTA. We also illustrate the modeling convenience provided by GTA in Section 9 based on experiments.



■ **Figure 4** Example of a GTA

#### 3.1 Timed automata

Timed automata (TA) of Alur-Dill [8] can be modeled as a GTA as follows:

- The set of states of the GTA is the same as the set of states of the TA.
- There are no future clocks in the GTA and its history clocks are the clocks of the TA.
- Each transition of the form  $q \xrightarrow{a, g, R} q'$  in a TA, where  $g$  is a guard,  $a$  a letter and  $R$  a subset of clocks to be reset, is replaced by a transition  $q \xrightarrow{a, \text{prog}} q'$  where  $\text{prog} = \langle g; [R] \rangle$ .
- Initially, all clocks must be 0, captured by setting  $g_0 = (X_H = 0)$ .
- The final guard is empty:  $g_f = \text{True}$ .



### 3.2 Event-clock automata

Event clock automata (ECA) of [9] can be modeled as a GTA as follows:

- The set of states of the GTA is the same as the set of states of the ECA.
- For each  $a \in \Sigma$ , the GTA has a history clock  $\overleftarrow{a}$  and a future clock  $\overrightarrow{a}$ .
- Each transition of the form  $q \xrightarrow{a,g} q'$  in a ECA, where  $g$  is a guard of the ECA,  $a$  a letter, is replaced by a transition  $q \xrightarrow{a,\text{prog}} q'$  where  $\text{prog} := \langle (\overrightarrow{a} = 0); [\overrightarrow{a}]; g; [\overleftarrow{a}] \rangle$ .
- At initialization, history clocks must be undefined (set to  $\infty$ ), captured by  $g_0 = (X_H = \infty)$ .
- At acceptance, all future clocks must be undefined, i.e.,  $g_f = (X_F = -\infty)$ .

### 3.3 Automata with timers

The third model we consider is that of automata with timers. Timers are timing constructs that are started/intialized with a certain time value at some point/event and *count down* to 0. They measure the time from when they were started till the timer hits 0, where the event of hitting 0 being called a *time-out* event. However, they can be stopped using a *stop* event at any intermediate point instead and in which case the timer must be freed for reuse later. Timers are a common construct in protocol specification, e.g., the ITU standard which uses timers rather than clocks [32] and Mealy machines with timers [33].

In our setting, a timer can be seen as a specific instance of a future clock. More precisely Automata with timers ( $A_{\Sigma}^{\text{timer}}$ ) can be modeled as GTA as follows:

- The set of states of the GTA is the same as the set of states of  $A_{\Sigma}^{\text{timer}}$ .
- The future clocks of GTA are the timers of  $A_{\Sigma}^{\text{timer}}$  and there are no history clocks.
- A transition of  $A_{\Sigma}^{\text{timer}}$  with action  $a$  from  $q$  to  $q'$  is encoded as a  $q \xrightarrow{a,\text{prog}} q'$  where  $\text{prog}$  is defined as follows:
  - if the transition starts timer  $x$  with value  $c \in \mathbb{R}_{\geq 0}$ , then  $\text{prog} = \langle x = -\infty; [x]; x = -c \rangle$ .
  - if the transition is guarded by  $\text{timeout}(x)$ , then  $\text{prog} = \langle x = 0; [x]; x = -\infty \rangle$ .
  - if the transition stops timer  $x$ , then  $\text{prog} = \langle [x]; x = -\infty \rangle$ .
- Initially, the timers are undefined, captured by  $g_0 = (X_F = -\infty)$  and the final guard is empty, i.e.,  $g_f = \text{True}$ .

We note that the timer above differs from a prophecy-event-clock (of ECA) though both are future clocks. Prophecy-clocks are released only when the event is seen, so at that point the value of the prophecy-clock must be 0. On the other hand timers can be stopped and released even when their value is not 0. This subtle difference has a surprising impact when we allow diagonal guards as we will see shortly.

## 4 The reachability problem for GTA

We are interested in the *reachability problem* for GTA. Formally,

► **Definition 8** (Reachability problem for GTA). *The reachability problem for a GTA  $\mathcal{A}$  is to decide whether  $\mathcal{A}$  has an accepting run.*

For normal TA, the reachability problem is decidable and PSPACE complete as shown in [8]. This was shown using the so-called region abstraction, by proving the existence of a finite time-abstract bisimulation. However, this is not the case for GTA. As explained in the previous subsection, GTA capture ECA, and as shown in [27, 28], there exists ECA for which there is no finite time-abstract bisimulation. However, reachability is still decidable in the specific case of ECA, as again shown in [9]. We note that for ECA model of [27, 28]

there are no diagonal constraints. In this case they show decidability via zone-extrapolation. In [2], another approach for decidability via zone simulations is shown. But again even in this model diagonal constraints are disallowed. Even more critically in GTA, we can capture timers and a priori we can have diagonal constraints even among timers. So, the question we ask is whether reachability is still decidable for GTA. Surprisingly, the answer is no. The intuition is that with future clocks and diagonal constraints, we get the ability to count (cf. Figure 4).

► **Theorem 9.** *Reachability for GTA is undecidable.*

**Proof.** We will do a reduction from counter machines. Given a counter machine, we will build a GTA with one future clock  $y_C$  for each counter  $C$  and one extra future clock  $z$ . The reduction uses diagonal constraints between  $z$  and the future clocks  $y_C$ .

Initially and after each transition, the value of the future clock  $z$  will be 0. Since a future clock has to be non-positive, time elapse is impossible. As an invariant, the value of the future clock  $y_C$  is the opposite of the value of counter  $C$ . The operations on counter  $C$  are encoded with the following programs:

$$\begin{aligned} \text{zero}_C &= \langle y_C = 0 \rangle \\ \text{inc}_C &= \langle [z]; z = y_C - 1; [y_C]; y_C = z; [z]; z = 0 \rangle \\ \text{dec}_C &= \langle y_C \leq -1; [z]; z = y_C + 1; [y_C]; y_C = z; [z]; z = 0 \rangle \end{aligned}$$

In the programs  $\text{inc}_C$  and  $\text{dec}_C$ , each release of a future clock is followed by a constraint which restricts the value non-deterministically chosen during the release. For instance,  $[z]; z = y_C - 1$  is equivalent to  $z := y_C - 1$ . Hence, the overall effect of  $\text{inc}_C$  is  $y_C := y_C - 1$ , maintaining all other clocks unchanged, including the invariant  $z = 0$ . ◀

Given this negative result, what can we do? A careful observation of the proof tells us that it is the interplay between diagonal constraints and arbitrary releases of future clocks that leads to undecidability. More precisely, the encoding depends on the fact that clocks  $z$  and  $y_C$  which are used in diagonal constraints ( $z = y_C - 1$ ,  $z = y_C + 1$  and  $y_C = z$ ) may have arbitrary values when they are released. This suggests a restricted subclass that we formalize next.

#### 4.1 $X_D$ -Safe GTA

► **Definition 10.** *Let  $X_D \subseteq X_F$  be a subset of future clocks.*

*A program  $\text{prog} = \langle g_1; [R_1]; g_2; [R_2]; \dots; g_k; [R_k]; g_{k+1} \rangle$  is  $X_D$ -safe if*

- *diagonal constraints between future clocks are restricted to clocks in  $X_D$ : if  $x - y \triangleleft c$  with  $x, y \in X_F$  occurs in some  $g_i$  then  $x, y \in X_D$ ;*
- *clocks in  $X_D$  should be 0 or  $-\infty$  before being released: if  $x \in X_D \cap R_i$  then  $x = 0$  or  $x = -\infty$  occurs in  $g_i$ .*

*A GTA  $\mathcal{A}$  is  $X_D$ -safe if it only uses  $X_D$ -safe programs on its transitions and the initial guard  $g_0$  sets each history clock to either 0 or  $\infty$ .*

► **Proposition 11.** *Timed automata, ECA (possibly with diagonal constraints) and Automata with timers (but without diagonal constraints) can all be captured by  $X_D$ -safe GTA.*

**Proof.** The proof follows by observing that in all the three cases, the safety condition holds. Timed automata do not have future clocks so the condition is vacuously true. In ECA, event-predicting clocks are always checked for 0 before being released, hence they are safe as

well with  $X_D = X_F$ . Automata with timers without diagonal constraints are also trivially safe with  $X_D = \emptyset$ .  $\blacktriangleleft$

The importance of safety is the following theorem which is the center-piece of this article.

► **Theorem 12.** *Reachability for  $X_D$ -safe GTA is decidable.*

We will establish this theorem by showing a finite, sound and complete zone based reachability algorithm for  $X_D$ -safe GTA. If the given GTA is not  $X_D$ -safe, then we lose proof of termination (unsurprisingly, since the problem is undecidable), but we still maintain soundness. Thus, even for such GTA when our algorithm does terminate it will give the correct answer.

## 5 Symbolic enumeration

We adapt the  $\mathcal{G}$ -simulation framework presented in [26] for timed automata with diagonal constraints to GTA. Diagonal constraints offer succinct modeling [13], but are quite challenging to handle efficiently in zone-based algorithms, and have led to pitfalls in the past: [12] showed that the erstwhile algorithm based on zone-extrapolations that was implemented in tools is incorrect for models with diagonal constraints; moreover no extrapolation based method can work for automata with diagonal constraints. The simulation framework by-passes this impossibility result and is the state-of-the-art for timed automata with diagonal constraints. The framework was extended to event-clock automata without diagonal constraints in [2]. We show that the ideas from [26] and [2] can be suitably combined to give an effective procedure for safe GTAs. This extension to GTAs enables us to understand the mechanics of diagonal constraints in future clocks.

The algorithm based on the  $\mathcal{G}$ -simulation framework involves:

1. computation of a set of constraints at every state of the automaton by a *static analysis* of the model,
2. a symbolic enumeration using *zones* to compute the *zone graph*,
3. a *simulation relation* between zones to ensure termination of the enumeration.

We will next adapt the static analysis to the GTA setting. The algorithm for the zone graph computation and the implementation of the simulation relation over zones is taken off-the-shelf from [26] and [2], except for a minor adaptation to include diagonal constraints involving future clocks. What is absent, and requires a non-trivial analysis, is the proof of termination. Therefore, we will mainly focus on this aspect and devote Section 8 for the termination argument.

### 5.1 A concrete simulation relation for GTA

We fix a GTA  $\mathcal{A} = (Q, \Sigma, X, T, (q_0, g_0), (Q_f, g_f))$  for this section. Our goal in this section is to define a simulation relation on the semantics of  $\mathcal{A}$ , i.e., on  $\text{TS}(\mathcal{A})$ . In the subsequent sections we will lift this to zones and show its finiteness. A simulation relation on  $\text{TS}(\mathcal{A})$  is a reflexive, transitive relation  $(q, v) \preceq (q', v')$  relating configurations with the same control state and (1) for every  $(q, v) \xrightarrow{\delta} (q, v + \delta)$ , we have  $(q', v') \xrightarrow{\delta} (q', v' + \delta)$  and  $(q, v + \delta) \preceq (q', v' + \delta)$ , (2) for every transition  $t$ , if  $(q, v) \xrightarrow{t} (q_1, v_1)$  for some valuation  $v_1$ , then  $(q', v') \xrightarrow{t} (q_1, v'_1)$  for some valuation  $v'_1$  with  $(q_1, v_1) \preceq (q_1, v'_1)$ .

For any set  $G$  of atomic constraints, we define a *preorder*  $\preceq_G$  on valuations by

$$v \preceq_G v' \quad \text{if } \forall \varphi \in G, \forall \delta \geq 0, \quad v + \delta \models \varphi \implies v' + \delta \models \varphi.$$

Notice that in the definition above, we *do not* restrict  $\delta$  to those such that  $v + \delta$  is a valuation: we may have  $v(x) + \delta > 0$  for some  $x \in X_F$ . In usual timed automata, this question does not arise, as elapsing any  $\delta$  from any given valuation always results in a valuation. But this is crucial for the proof of Theorem 14 below.

Intuitively, the preorder above is a simulation w.r.t. the constraints in  $G$  even after time elapse. But we need this to also be a simulation w.r.t. discrete transitions. To achieve this, the set of constraints  $G$  should depend on the available discrete transitions. In fact, we define a map  $\mathcal{G}$  from states to set of constraints, in such a way that it captures the simulation w.r.t. the discrete actions. In other words, our focus will be to choose state-dependent sets of constraints (given by the map  $\mathcal{G}$ ) depending on  $\mathcal{A}$  such that the resulting preorder induces a simulation on  $\mathbb{TS}(\mathcal{A})$ .

As a first step towards this, we define, for any set  $G$  of constraints and any program  $\text{prog}$ , a set of constraints  $G' = \text{pre}(\text{prog}, G)$  such that, if  $v \preceq_{G'} v'$  and  $v \xrightarrow{\text{prog}} v_1$  then there exists  $v' \xrightarrow{\text{prog}} v'_1$  such that  $v_1 \preceq_G v'_1$ . This set is defined inductively as follows ( $G$  is a set of atomic constraints,  $R$  is a set of clocks,  $g$  is an *arbitrary* constraint,  $y - x \triangleleft c$  is an *atomic* constraint):

$$\begin{aligned} \text{pre}(\text{prog}_1; \text{prog}_2, G) &= \text{pre}(\text{prog}_1, \text{pre}(\text{prog}_2, G)) \\ \text{pre}(g, G) &= \text{split}(g) \cup G \\ \text{pre}([R], G) &= \bigcup_{\varphi \in G} \text{pre}([R], \{\varphi\}) \\ \text{pre}([R], \{y - x \triangleleft c\}) &= \begin{cases} \{y - x \triangleleft c\} & \text{if } x, y \notin R \\ \{y \triangleleft c\} & \text{if } x \in R, y \notin R \\ \{-x \triangleleft c\} & \text{if } x \notin R, y \in R \\ \emptyset & \text{if } x, y \in R \end{cases} \end{aligned}$$

where  $\text{split}(g)$  is the set of atomic constraints occurring in  $g$ .

Now, the choice of suitable  $G$  will be obtained by static analysis, on the lines of what was done for timed automata with diagonals [24, 25, 26], but adapted to our more powerful model. More precisely, we define the map  $\mathcal{G}$  from  $Q$  to sets of atomic constraints as the least fixpoint of the set of equations:

$$\mathcal{G}(q) = \{x \leq 0 \mid x \in X_F\} \cup \bigcup_{q \xrightarrow{a, \text{prog}} q'} \text{pre}(\text{prog}, \mathcal{G}(q')) \quad (4)$$

Finally, based on  $\preceq_G$  and the  $\mathcal{G}(q)$  computation, we can define a preorder  $\preceq_{\mathcal{A}}$  between configurations of  $\mathbb{TS}(\mathcal{A})$  as  $(q, v) \preceq_{\mathcal{A}} (q', v')$  if  $q = q'$  and  $v \preceq_{\mathcal{G}(q)} v'$ .

We will need the following technical lemma.

► **Lemma 13.** *Let  $G$  be a set of atomic constraints and  $G' = \text{pre}([R], G)$ . Let  $R$  be a set of clocks. Let  $v_1, v_2 \in \mathbb{V}$  be valuations and let  $v'_1 \in [R]v_1$  and  $v'_2 \in [R]v_2$  be such that  $v'_2 \downarrow_R = v'_1 \downarrow_R$ . Then,  $v_1 \preceq_{G'} v_2$  implies  $v'_1 \preceq_G v'_2$ .*

**Proof.** Since  $v'_1 \in [R]v_1$  and  $v'_2 \in [R]v_2$ , we have  $v'_1 \downarrow_{X \setminus R} = v_1 \downarrow_{X \setminus R}$  and  $v'_2 \downarrow_{X \setminus R} = v_2 \downarrow_{X \setminus R}$ . Moreover, from our assumption, we have  $v'_2 \downarrow_R = v'_1 \downarrow_R$ .

We have  $v \preceq_G v'$  iff  $v \preceq_{\varphi} v'$  for all  $\varphi \in G$ . Hence, it is sufficient to prove the lemma when  $G = \{\varphi\}$  where  $\varphi$  is an atomic constraint  $y - x \triangleleft c$ . Let  $G' = \text{pre}([R], G)$ , which is either  $\emptyset$  when  $x, y \in R$  or a singleton  $\{\varphi'\}$ . Suppose  $v_1 \preceq_{G'} v_2$ .

■ Suppose that  $x, y \in R$ . In this case, we have  $v'_2(y) = v'_1(y)$  and  $v'_2(x) = v'_1(x)$ . We then have  $v'_1 \preceq_{\varphi} v'_2$  (we do not need any hypothesis on  $v_1, v_2$ ).

- Suppose that  $x, y \notin R$ . In this case, we have  $\varphi' = \varphi$ ,  $v'_1(y) = v_1(y)$  and  $v'_2(y) = v_2(y)$ ,  $v'_1(x) = v_1(x)$  and  $v'_2(x) = v_2(x)$ . Since  $v_1 \preceq_\varphi v_2$ , it follows that  $v'_1 \preceq_\varphi v'_2$ .
- Suppose that  $x \in R$  and  $y \notin R$ . In this case, we have  $v_1 \preceq_{\varphi'} v_2$  with  $\varphi' = y \triangleleft c$ . We need to show that  $v'_1 \preceq_\varphi v'_2$ . Let  $\delta \geq 0$  and assume that  $v'_1 + \delta \models \varphi$ , i.e.,  $v'_1(y) - v'_1(x) \triangleleft c$ . We have to show that  $v'_2(y) - v'_2(x) \triangleleft c$ .  
We have  $v'_1(y) = v_1(y)$ ,  $v'_2(y) = v_2(y)$  and  $v'_2(x) = v'_1(x) \leq 0$ . Let  $\delta' = -v'_1(x) \geq 0$ . We get  $v_1 + \delta' \models \varphi'$ . We deduce that  $v_2 + \delta' \models \varphi'$ , i.e.,  $v'_2(y) - v'_2(x) \triangleleft c$  as desired.
- Suppose that  $y \in R$  and  $x \notin R$ . The proof is symmetric to the case above, and proceeds by similar arguments. In this case, we have  $v_1 \preceq_{\varphi'} v_2$  where  $\varphi' = -x \triangleleft c$ . We need to show that  $v'_1 \preceq_\varphi v'_2$ . Let  $\delta \geq 0$  and assume that  $v'_1 + \delta \models \varphi$ , i.e.,  $v'_1(y) - v'_1(x) \triangleleft c$ . We have to show that  $v'_2(y) - v'_2(x) \triangleleft c$ .  
We have  $v'_1(x) = v_1(x)$ ,  $v'_2(x) = v_2(x)$  and  $v'_2(y) = v'_1(y) \leq 0$ . Let  $\delta' = -v'_1(y) \geq 0$ . We get  $-(v_1 + \delta')(x) \triangleleft c$ , i.e.,  $v_1 + \delta' \models \varphi'$ . We deduce that  $v_2 + \delta' \models \varphi'$ , i.e.,  $v'_2(y) - v'_2(x) \triangleleft c$  as desired.  $\blacktriangleleft$

Now, let us prove that  $\preceq_{\mathcal{A}}$  defined above is indeed a simulation relation.

► **Theorem 14.** *The relation  $\preceq_{\mathcal{A}}$  is a simulation on the transition system  $\text{TS}_{\mathcal{A}}$  of GTA  $\mathcal{A}$ .*

**Proof.** Assume that  $(q, v_1) \preceq_{\mathcal{A}} (q, v'_1)$ , i.e.,  $v_1 \preceq_{\mathcal{G}(q)} v'_1$ .

**Delay transition** Assume that  $(q, v_1) \xrightarrow{\delta} (q, v_1 + \delta)$  is a transition of  $\text{TS}_{\mathcal{A}}$ . Then,  $v_1 + \delta \models X_F \leq 0$ . Since  $\mathcal{G}(q)$  contains  $x \leq 0$  for all  $x \in X_F$  and  $v_1 \preceq_{\mathcal{G}(q)} v'_1$ , we deduce that  $v'_1 + \delta \models X_F \leq 0$ . Therefore,  $(q, v'_1) \xrightarrow{\delta} (q, v'_1 + \delta)$  is a transition in  $\text{TS}_{\mathcal{A}}$ . It is easy to see that  $v_1 + \delta \preceq_{\mathcal{G}(q)} v'_1 + \delta$ .

**Discrete transition** Let  $(q_1, v_1) \xrightarrow{a, \text{prog}} (q, v) \in \text{TS}_{\mathcal{A}}$  for some transition  $t = (q_1, a, \text{prog}, q)$  of  $\mathcal{A}$ . Then we need to show that there exists  $v'$  such that  $(q, v) \preceq_{\mathcal{A}} (q, v')$  and  $(q_1, v'_1) \xrightarrow{a, \text{prog}} (q, v')$ . Wlog, we can assume that  $\text{prog} = \langle g_1; [R_1]; \dots g_k; [R_k] \rangle$  i.e., an alternating sequences of guards and changes (reset/release). By definition, this means that there are  $v_2, \dots, v_{k+1}$  with  $v_{k+1} = v$  and  $v_i \xrightarrow{g_i; [R_i]} v_{i+1}$  for all  $1 \leq i \leq k$ . This means that for all  $1 \leq i \leq k$  we have  $v_i \models g_i$  and  $v_{i+1} \in [R_i]v_i$ . Define  $G_{k+1} = \mathcal{G}(q)$  and  $G_i = \text{pre}(\langle g_i; [R_i] \rangle, G_{i+1})$  for  $1 \leq i \leq k$  so that  $G_1 = \mathcal{G}(q_1)$ . Now, for each  $1 \leq i \leq k$  we construct below by induction valuations  $v'_2, \dots, v'_{k+1}$  such that  $v'_i \xrightarrow{g_i; [R_i]} v'_{i+1}$  and  $v_{i+1} \preceq_{G_{i+1}} v'_{i+1}$ . With  $v' = v'_{k+1}$  we get  $(q_1, v'_1) \xrightarrow{a, \text{prog}} (q, v')$  and  $(q, v) \preceq_{\mathcal{A}} (q, v')$  as desired.

For  $i = 1$ , we have  $v_1 \preceq_{G_1} v'_1$  by hypothesis. Now, assume that  $v_i \preceq_{G_i} v'_i$  for some  $1 \leq i \leq k$ . Since  $\text{split}(g_i) \subseteq G_i$  and  $v_i \models g_i$ , we deduce that  $v'_i \models g_i$ . Now, let  $v'_{i+1}$  be defined by  $v'_{i+1} \downarrow_{R_i} = v_{i+1} \downarrow_{R_i}$  and  $v'_{i+1} \downarrow_{X \setminus R_i} = v'_i \downarrow_{X \setminus R_i}$ . We have  $v'_{i+1} \in [R_i]v'_i$  and since  $v'_i \models g_i$  we deduce that  $v'_i \xrightarrow{g_i; [R_i]} v'_{i+1}$ . Notice that  $\text{pre}([R_i], G_{i+1}) \subseteq G_i$ . Hence, using Lemma 13,  $v'_{i+1} \downarrow_{R_i} = v_{i+1} \downarrow_{R_i}$  and  $v_i \preceq_{G_i} v'_i$  we can conclude that  $v_{i+1} \preceq_{G_{i+1}} v'_{i+1}$ , which completes the proof.  $\blacktriangleleft$

## 5.2 Zones for GTA and a symbolic reachability algorithm

The most widely used approach for checking reachability in a timed automaton (and more recently in event-clock automata) is based on reachability in a graph called the *zone graph* of a timed automaton [20]. Roughly, *zones* [10] are sets of valuations that can be represented efficiently using constraints between differences of clocks. In this section, we introduce an analogous notion for generalized timed automata. We consider *GTA zones*, which are special sets of valuations of generalized timed automata.

► **Definition 15 (GTA zones).** A GTA zone is a set of valuations satisfying a conjunction of constraints of the form  $y - x \triangleleft c$ , where  $x, y \in X \cup \{0\}$ ,  $c \in \mathbb{Z}$  and  $\triangleleft \in \{\leq, <\}$ .

Thus zones are an abstract representation of sets of valuations. Then, an abstract configuration, also called a *node*, is a pair consisting of a state and a zone. Firing a transition  $t := (q, a, \text{prog}, q')$  in a GTA  $\mathcal{A}$  from node  $(q, Z)$  will result in another node following a sequence of operations that we now define.

► **Definition 16 (Operations on GTA zones).** Let  $g$  be a guard,  $R \subseteq X$  be a set of clocks and  $Z$  be a GTA zone.

- *Guard intersection:*  $Z \wedge g := \{v \mid v \in Z \text{ and } v \models g\}$
- *Release/Reset:*  $[R]Z = \bigcup_{v \in Z} [R]v$  (as defined in Section 2)
- *Time elapse:*  $\vec{Z} = \{v + \delta \mid v \in Z, \delta \in \mathbb{R}_{\geq 0} \text{ s.t. } v + \delta \models (X_F \leq 0)\}$

From the above definition, it is easy to see that starting from a GTA zone  $Z$ , the successors after the above operations are also GTA zones. A guard  $g$  can be seen as yet another GTA zone and hence guard intersection is just an intersection operation between two GTA zones. Similarly, the change operation preserves GTA zones. Finally, as is usual with timed automata, zones are closed under the time elapse operation.

Thus, for a transition  $t := (q, a, \text{prog}, q')$  and a node  $(q, Z)$ , we can define the successor node  $(q', Z')$ , and we write  $(q, Z) \xrightarrow{t} (q', Z')$ , where  $Z'$  is the zone computed by the following sequence of operations: Let  $\text{prog} = \text{prog}_1; \dots; \text{prog}_n$ , where each  $\text{prog}_i$  is an atomic program, i.e., a guard  $g$  or a change  $R \subseteq X$ . Then we define zones  $Z_1, \dots, Z_{n+1}$  where,  $Z_1 = Z$ ,  $Z' = \vec{Z}_{n+1}$ , and for each  $1 \leq i \leq n$ ,

$$Z_{i+1} = \begin{cases} Z_i \wedge \text{prog}_i & \text{if } \text{prog}_i \text{ is a guard} \\ [\text{prog}_i]Z_i & \text{if } \text{prog}_i \text{ is a change} \end{cases}$$

Now, we can lift zone graphs, simulations from TA to GTA and obtain a symbolic reachability algorithm for GTA.

► **Definition 17 (GTA zone graph).** Given a GTA  $\mathcal{A}$ , its GTA zone graph, denoted  $\text{GZG}(\mathcal{A})$ , is defined as follows: Nodes are of the form  $(q, Z)$  where  $q$  is a state and  $Z$  is a GTA zone. The initial node is  $(q_0, \vec{Z}_0)$  where  $q_0$  is the initial state and  $Z_0$  is given by  $g_0 \wedge (X_F \leq 0) \wedge (X_H \geq 0)$  ( $Z_0$  is the set of all valuations which satisfy the initial constraint  $g_0$ ). For every node  $(q, Z)$  and every transition  $t := (q, a, \text{prog}, q')$  there is a transition  $(q, Z) \xrightarrow{t} (q', Z')$  in the GTA zone graph. A node  $(q, Z)$  is accepting if  $q \in Q_f$  and  $Z \cap g_f$  is non-empty, i.e., there exists a valuation in  $Z$  satisfying the final constraint.

Similar to the case of zone graphs for timed automata and event zone graphs for event-clock automata, the GTA zone graph can be used to decide reachability for generalized timed automata. A node  $(q, Z)$  is said to be reachable (in  $\mathcal{A}$ ) if there is a path from the initial node  $(q_0, \vec{Z}_0)$  to  $(q, Z)$  in  $\text{GZG}(\mathcal{A})$ . Thus, reachability of a final state in  $\mathcal{A}$  reduces to checking reachability of an accepting node in  $\text{GZG}(\mathcal{A})$ . However, as in the case of zone graphs for timed automata,  $\text{GZG}(\mathcal{A})$  is also not guaranteed to be finite. Hence, we need to compute a finite truncation of the GTA zone graph, which is still sound and complete for reachability.

► **Definition 18 (Simulation on GTA zones and finiteness).** Let  $\preceq$  be a simulation relation on  $\text{TS}(\mathcal{A})$ . For two GTA zones  $Z, Z'$ , we say  $(q, Z) \preceq (q, Z')$  if for every  $v \in Z$  there exists  $v' \in Z'$  such that  $(q, v) \preceq (q, v')$ . The simulation  $\preceq$  is said to be finite if for every sequence  $(q, Z_1), (q, Z_2), \dots$  of reachable nodes, there exists  $j > i$  such that  $(q, Z_j) \preceq (q, Z_i)$ .

Now, the reachability algorithm, as in TA, enumerates the nodes of the GTA zone graph and uses  $\preceq$  to truncate nodes that are smaller with respect to the simulation.

► **Definition 19 (Reachability algorithm).** *Let  $\mathcal{A}$  be a GTA and  $\preceq$  a simulation relation on  $\text{TS}(\mathcal{A})$ . Add the initial node of the GTA zone graph  $(q_0, \vec{Z}_0)$  to a Waiting list. Repeat the following until Waiting list is empty:*

- *Pop a node  $(q, Z)$  from the Waiting list and add it to the Passed list.*
- *For every  $(q, Z) \xrightarrow{t} (q_1, Z_1)$ : if there exists a  $(q_1, Z'_1)$  in the Passed or Waiting lists such that  $(q_1, Z_1) \preceq (q_1, Z'_1)$ , discard  $(q_1, Z_1)$ ; else add  $(q_1, Z_1)$  to the Waiting list.*

*If some accepting node is reached, the algorithm terminates and returns a Yes. Else, it continues until there are no further nodes to be explored and returns a No answer.*

The correctness of the above algorithm, follows from the correctness of the simulation approach in timed automata, with termination guaranteed when the simulation used is finite [31]. Thus, the following theorem is a straightforward adaptation of the corresponding proof [20, 31] from timed automata.

► **Theorem 20.** *Given GTA  $\mathcal{A}$ ,*

1. *GZG( $\mathcal{A}$ ) is sound and complete for reachability, i.e., an accepting node is reachability in GZG( $\mathcal{A}$ ) iff an accepting state is reachable in  $\mathcal{A}$ .*
2.  *$\mathcal{A}$  has an accepting run iff the reachability algorithm returns Yes.*
3. *The reachability algorithm is guaranteed to terminate, if the simulation  $\preceq$  used is finite.*

For the simulation, we will use  $\preceq_{\mathcal{A}}$  as defined in the previous section. But we still need to show that it is finite. This is the hardest and most technical part of this paper and will form the bulk of Sections 7, 8. Before that, in Section 6, we first address the question of implementability of the above algorithm for GTA and the data structures needed for it, in particular the notion of distance graphs. Importantly, the properties that we show on the distance graphs will also be used in showing finiteness later.

## 6 Computing with GTA zones using distance graphs

One of the main innovations towards implementability of timed automata was the development of Difference-Bound-Matrices (DBMs) as efficient data structures to represent and manipulate zones [10]. For this, the central step was to view zones as *distance graphs* that could be immediately represented as DBMs. To have a practical implementation of GTA, and to use the vast repertoire of existing tools and techniques for DBMs, a vital step is to be able to encode GTA zones as distance graphs.

We now show that GTA zones can be represented using Difference-Bound-Matrices (DBMs) and the operations required for the reachability algorithm can be implemented using DBMs. The first hurdle is that for normal timed automata, each edge of the distance graph (i.e., entry in a DBM) encodes a constraint of the form  $x - y \triangleleft c$ , where edges/entries are  $(<, \infty)$  or  $(\triangleleft, c)$  with  $c \in \mathbb{R}$  and  $\triangleleft \in \{<, \leq\}$ . But for GTA, we need to deal with valuations  $+\infty$  or  $-\infty$ . For this purpose, we use more general weights as introduced in Definition 1 and we extend the algebra of weights to the new entries in a natural way. Before discussing the representation of GTA zones as distance graphs, we briefly recall the extended algebra and some of the results we will use.

► **Definition 21 ([3]).** *[Order and sum of weights] Let  $(\triangleleft, c), (\triangleleft', c') \in \mathcal{C}$  be weights.*

**Order.** *Define  $(\triangleleft, c) < (\triangleleft', c')$  when either (1)  $c < c'$ , or (2)  $c = c'$  and  $\triangleleft$  is  $<$  while  $\triangleleft'$  is  $\leq$ . This is a total order with  $(<, -\infty) < (\leq, -\infty) < (\triangleleft, c) < (<, \infty) < (\leq, \infty)$  for all  $c \in \mathbb{R}$ .*



**Sum.** We define the commutative sum operation as follows.

$$\begin{aligned}
(<, -\infty) + \alpha &= (<, -\infty) && \text{if } \alpha \in \mathcal{C} \\
(\leq, \infty) + \alpha &= (\leq, \infty) && \text{if } \alpha \in \mathcal{C} \setminus \{(<, -\infty)\} \\
(\leq, -\infty) + \alpha &= (\leq, -\infty) && \text{if } \alpha \in \mathcal{C} \setminus \{(<, -\infty), (\leq, \infty)\} \\
(<, \infty) + \alpha &= (<, \infty) && \text{if } \alpha \in \mathcal{C} \setminus \{(<, -\infty), (\leq, -\infty), (\leq, \infty)\} \\
(\triangleleft, c) + (\triangleleft', c') &= (\triangleleft'', c + c') && \text{if } c, c' \in \mathbb{R} \text{ and } \triangleleft'' = \leq \text{ if } \triangleleft = \triangleleft' = \leq \text{ and } \triangleleft'' = < \text{ otherwise.}
\end{aligned}$$

Notice that sum of weights is an associative operation and  $\alpha + (\leq, 0) = \alpha$  for all  $\alpha \in \mathcal{C}$ .

The intuition behind the above definition of order is that when  $(\triangleleft, c) < (\triangleleft', c')$ , the set of valuations that satisfies a constraint  $x - y \triangleleft c$  is contained in the solution set of  $x - y \triangleleft' c'$ . For the sum, the following lemma gives the idea behind our choice of definition.

► **Lemma 22** ([3]). Let  $x, y, z \in X \cup \{0\}$  be clocks,  $(\triangleleft_1, c_1), (\triangleleft_2, c_2) \in \mathcal{C}$  be weights and  $(\triangleleft, c) = (\triangleleft_1, c_1) + (\triangleleft_2, c_2)$ . For all valuations  $v \in \mathbb{V}$ , if  $v \models y - x \triangleleft_1 c_1$  and  $v \models z - y \triangleleft_2 c_2$ , then  $v \models z - x \triangleleft c$ .

Equipped with the weights and the arithmetic over it, we can now define the representation of GTA zones as distance graphs.

## 6.1 Distance graphs over the extended algebra

► **Definition 23** (Distance graphs). A distance graph  $\mathbb{G}$  is a weighted directed graph without self-loops, with vertex set being  $X \cup \{0\} = X_F \cup X_H \cup \{0\}$ , edges being labeled with weights from  $\mathcal{C} \setminus \{(<, -\infty)\}$ .<sup>4</sup> We define  $\llbracket \mathbb{G} \rrbracket := \{v \in \mathbb{V} \mid v \models y - x \triangleleft c \text{ for all edges } x \xrightarrow{\triangleleft, c} y \text{ in } \mathbb{G}\}$ . The weight of edge  $x \rightarrow y$  is denoted  $\mathbb{G}_{xy}$  and we set  $\mathbb{G}_{xy} = (\leq, \infty)$  if there is no edge  $x \rightarrow y$ . The weight of a path is the sum of the weights of its edges. A cycle in  $\mathbb{G}$  is said to be negative if its weight is strictly less than  $(\leq, 0)$ .

We say that  $\mathbb{G}$  is in standard form if it satisfies the following conditions;

1.  $\mathbb{G}_{0x} \leq (\leq, 0)$  for all  $x \in X_F$  and  $\mathbb{G}_{x0} \leq (\leq, 0)$  for all  $x \in X_H$ .
2. For all  $x, y \in X$ , if  $\mathbb{G}_{xy} \neq (\leq, \infty)$  then  $\mathbb{G}_{x0} \neq (\leq, \infty)$  and  $\mathbb{G}_{0y} \neq (\leq, \infty)$ .

We extend the order on weights to distance graphs pointwise: Let  $\mathbb{G}, \mathbb{G}'$  be distance graphs, we write  $\mathbb{G} \leq \mathbb{G}'$  when  $\mathbb{G}_{xy} \leq \mathbb{G}'_{xy}$  for all edges  $x \rightarrow y$ . Notice that this implies  $\llbracket \mathbb{G} \rrbracket \subseteq \llbracket \mathbb{G}' \rrbracket$ .

The intuition of the standard form comes from the semantics of a distance graph. In classical timed automata the distance graph has no negative cycles iff its semantics is non-empty. However, for distance graphs over the extended algebra, we will see that this is true only when it is in standard form. To illustrate the need for general form, we provide an example from [3] here.

► **Example 24.** [3] Suppose that valuations are finite and that we have constraints:  $y - x \leq 1$  and  $-y \leq 2$ . From these constraints, we can infer  $-x \leq 3$  just by adding the inequalities. If there was another constraint  $x \leq -4$ , we will get unsatisfiability. In the language of distance graphs, the two initial constraints correspond to edges  $x \xrightarrow{\leq 1} y$  and  $y \xrightarrow{\leq 2} 0$ . The derived constraint  $-x \leq 3$  is obtained as the edge  $x \xrightarrow{\leq 3} 0$ . The constraint  $x \leq -4$  corresponds to

<sup>4</sup> If we allowed an edge with weight  $\mathbb{G}_{xy} = (<, -\infty)$  then we would get  $\llbracket \mathbb{G} \rrbracket = \emptyset$  since the constraint  $y - x < -\infty$  is equivalent to false.

$0 \xrightarrow{\leq -4} x$  and the unsatisfiability is witnessed by a negative cycle  $0 \xrightarrow{\leq -4} x \xrightarrow{\leq 3} 0$ . Basically, adding the weights of  $x \rightarrow y$  and  $y \rightarrow 0$ , we get the strongest possible constraint about  $x \rightarrow 0$  resulting from the two constraints  $x \rightarrow y$  and  $y \rightarrow 0$ .

This holds no more in the extended algebra, due to the fundamental difference while adding weight  $(\leq, \infty)$ . Consider the constraints  $y - x \leq 1$  and  $0 - y \leq \infty$ , corresponding to the edges  $x \xrightarrow{\leq 1} y$  and  $y \xrightarrow{\leq \infty} 0$  in the distance graph. Adding the two weights gives the edge  $x \xrightarrow{\leq \infty} 0$ , corresponding to the constraint  $0 - x \leq \infty$ . This is *not* the strongest possible constraint on  $x$  induced by the constraints  $y - x \leq 1$  and  $0 - y \leq \infty$ . Indeed, a valuation  $v$  with  $v(x) = -\infty$  satisfies the constraint  $0 - x \leq \infty$ . But, any valuation  $v$  satisfying  $y - x \leq 1$  should have  $v(x) \neq -\infty$ , irrespective of the value of  $v(y)$ . Now, if we also had a constraint  $x - 0 \leq -\infty$ , corresponding to edge  $0 \xrightarrow{\leq -\infty} x$ , we have no negative cycle in the corresponding distance graph. But the set of constraints  $y - x \leq 1$ ,  $0 - y \leq \infty$  and  $x - 0 \leq -\infty$  is not feasible. In order to get a correspondence between negative cycles and empty solution sets, we propose the standard form. The standard form equips the graph with the additional information that when  $y - x$  is bounded by a finite value, that is, edge  $x \rightarrow y$  does not have weight  $(\leq, \infty)$ , the constraint  $0 - x$  (and also  $y - 0$ ) is at most  $(<, \infty)$ . With this information, we get negative cycles whenever there is a contradiction.

Fortunately, it turns out that each distance graph  $\mathbb{G}$  can be transformed into an equivalent distance graph  $\mathbb{G}'$  which is in standard form. By equivalent, we mean  $\llbracket \mathbb{G} \rrbracket = \llbracket \mathbb{G}' \rrbracket$ . First, we set  $\mathbb{G}'_{0x} = \min(\mathbb{G}_{0x}, (\leq, 0))$  for  $x \in X_P$  and  $\mathbb{G}'_{x0} = \min(\mathbb{G}_{x0}, (\leq, 0))$  for  $x \in X_H$ . Moreover, if  $x \in X_P$  then we set  $\mathbb{G}'_{x0} = \min(\mathbb{G}_{x0}, (<, \infty))$  if  $\mathbb{G}_{xy} \neq (\leq, \infty)$  for some  $y \neq x$ , otherwise we keep  $\mathbb{G}'_{x0} = \mathbb{G}_{x0}$ . Similarly, if  $y \in X_H$  then we set  $\mathbb{G}'_{0y} = \min(\mathbb{G}_{0y}, (<, \infty))$  if  $\mathbb{G}_{xy} \neq (\leq, \infty)$  for some  $x \neq y$ , otherwise we keep  $\mathbb{G}'_{0y} = \mathbb{G}_{0y}$ . Finally, for  $x, y \in X$  with  $x \neq y$  we set  $\mathbb{G}'_{xy} = \mathbb{G}_{xy}$ . The graph  $\mathbb{G}'$  constructed above is called the standardization of  $\mathbb{G}$ .

► **Lemma 25.** [3] *The standardization  $\mathbb{G}'$  of a distance graph  $\mathbb{G}$  is in standard form and  $\llbracket \mathbb{G}' \rrbracket = \llbracket \mathbb{G} \rrbracket$ . Moreover,  $\llbracket \mathbb{G} \rrbracket \neq \emptyset$  iff  $\mathbb{G}$  has no negative cycles.*

Now, suppose  $\mathbb{G}'$  (in standard form) has no negative cycles, then we construct  $\mathbb{G}''$  by replacing the weight of an edge  $x \rightarrow y$  by the minimum of the weights of the paths from  $x$  to  $y$  in  $\mathbb{G}'$ . Such a  $\mathbb{G}''$  is called the *normalization* of  $\mathbb{G}'$  and has several useful properties.

► **Lemma 26 (Normalization).** [3] *Let  $\mathbb{G}$  be a standard distance graph with no negative cycles. The normalization  $\mathbb{G}''$  of  $\mathbb{G}$  is normalized and  $\llbracket \mathbb{G}'' \rrbracket = \llbracket \mathbb{G} \rrbracket$ .*

Let  $Z$  be a nonempty zone. Writing the constraints in  $Z$  as a distance graph, followed by standardizing and normalizing it, results in *its canonical distance graph*  $\mathbb{G}(Z)$ :  $\llbracket \mathbb{G}(Z) \rrbracket = Z$  and  $\mathbb{G}(Z)$  is minimal among the standard graphs  $G$  with  $\llbracket G \rrbracket = Z$ . We denote by  $Z_{xy}$  the weight of the edge  $x \rightarrow y$  in  $\mathbb{G}(Z)$ . Formally,

► **Lemma 27.** [3] *Let  $\mathbb{G}, \mathbb{G}'$  be two distance graphs where  $\mathbb{G}$  is normalized. If  $\llbracket \mathbb{G} \rrbracket \subseteq \llbracket \mathbb{G}' \rrbracket$  then  $\mathbb{G} \leq \mathbb{G}'$ . In particular, if both  $\mathbb{G}, \mathbb{G}'$  are normalized and if  $\llbracket \mathbb{G} \rrbracket = \llbracket \mathbb{G}' \rrbracket$  then  $\mathbb{G} = \mathbb{G}'$ .*

## 6.2 Successor computation for GTA zones

Next, we show how we can perform GTA zone operations on the respective distance graphs of the zones. Thanks to the algebra over the new weights, the arguments are very similar to the cases for normal timed automata and event-clock automata [2, 3]. One important technical difference from these earlier works is that due to the presence of diagonal constraints among future clocks, after a guard intersection, we need to explicitly standardize the zone in order to

check its emptiness by looking for a negative cycle. When we had only non-diagonal guards, this was not necessary, as non-diagonal guards cannot change the weight of  $x \rightarrow y$  edges.

► **Definition 28 (Operations on distance graphs).** Let  $\mathbb{G}$  be a normalized distance graph, let  $g$  be a guard and let  $R \subseteq X$  be a set of clocks.

- **Guard intersection:** a distance graph  $\mathbb{G}_g$  is obtained from  $\mathbb{G}$  as follows,
  - for each constraint  $y - x \triangleleft c$  in  $g$ , replace weight of edge  $x \rightarrow y$  with  $\min(\mathbb{G}_{xy}, (\triangleleft, c))$ ,
  - standardize the graph obtained in the above step,
  - normalize the resulting graph if it has no negative cycles.
- **Release/Reset:** a distance graph  $[R]\mathbb{G}$  is obtained from  $\mathbb{G}$  by
  - removing all edges involving clocks  $x \in R$  and then
  - adding the edges  $0 \xrightarrow{(\leq, 0)} x$  and  $x \xrightarrow{(\leq, \infty)} 0$  for all  $x \in R_F$ ,
  - adding the edges  $0 \xrightarrow{(\leq, 0)} x$  and  $x \xrightarrow{(\leq, 0)} 0$  for all  $x \in R_H$ , and then
  - normalizing the resulting graph.
- **Time elapse:** the distance graph  $\vec{\mathbb{G}}$  is obtained by the following transformation:
  - for all history clocks  $x$ , if  $\mathbb{G}_{0x} \neq (\leq, \infty)$  then replace it with  $(<, \infty)$ ,
  - for all future clocks  $x$ , if  $\mathbb{G}_{0x} \neq (\leq, -\infty)$  then replace it with  $(\leq, 0)$ ,
  - normalize the resulting graph.

The theorem below says that the operations on GTA zones translate easily to operations on distance graphs and that the successor of a GTA zone is a GTA zone. Except for the release operation  $[R_F]\mathbb{G}$ , the rest of the operations are standard in timed automata, but they do not use weights  $(\leq, -\infty)$ ,  $(\leq, +\infty)$ . We can perform all these operations in the new algebra with quadratic complexity, matching the best-known complexity for timed automata without diagonal constraints [42].

► **Theorem 29.** Let  $\mathbb{G}$  be a normalized distance graph,  $g$  be a guard and  $R \subseteq X$  be a set of clocks. We can compute, in  $\mathcal{O}(|X|^2)$  time, normalized distance graphs  $\mathbb{G}_g$ ,  $[R]\mathbb{G}$  and  $\vec{\mathbb{G}}$ , such that  $\llbracket \mathbb{G} \rrbracket \wedge g = \llbracket \mathbb{G}_g \rrbracket$ ,  $\llbracket [R]\mathbb{G} \rrbracket = \llbracket [R]\mathbb{G} \rrbracket$ , and  $\llbracket \vec{\mathbb{G}} \rrbracket = \llbracket \vec{\mathbb{G}} \rrbracket$ .

The proof of the theorem follows from Lemmas 30, 31 and 32 which we state below for completeness. We give only the proof of Lemma 30 which is a bit more involved than the non-diagonal case handled in [3]. The proofs of Lemma 31 and 32 can be found in [3].

► **Lemma 30.** Let  $\mathbb{G}$  be a normalized distance graph and let  $g$  be a guard. Then  $\llbracket \mathbb{G}_g \rrbracket = \llbracket \mathbb{G} \rrbracket \wedge g$ , and  $\mathbb{G}_g$  can be computed in time  $\mathcal{O}(|g| + |X|^2)$ .

**Proof.** According to the definition, we first construct an intermediate graph  $\mathbb{G}'$  by replacing weights of edges of the form  $x \rightarrow y$  depending on the atomic constraints in  $g$ . It is easy to see that  $\llbracket \mathbb{G}' \rrbracket = \llbracket \mathbb{G} \rrbracket \wedge g$  and that  $\mathbb{G}'$  is computed from  $\mathbb{G}$  in time  $\mathcal{O}(|g| + |X|^2)$ . The standardization process computes in time  $\mathcal{O}(|X|^2)$  a graph  $\mathbb{G}''$  in standard form with the same solution set. If  $\mathbb{G}''$  has no negative cycle, the normalization process does not change the solution set.

In general, checking for negative cycle and normalization of  $\mathbb{G}''$  may take time  $\mathcal{O}(|X|^3)$ . Alternatively, we can start by handling the modification of non-diagonal edges as we did in the ECA paper: see below how to check for negative cycles and normalize in time  $\mathcal{O}(|X|^2)$ . Then, for each diagonal constraint  $y - x \triangleleft c$  in  $g$ , we reduce the weight of each edge  $x' \rightarrow y'$  to  $\min(\mathbb{G}''_{x'y'}, \mathbb{G}''_{x'x} + (\triangleleft, c) + \mathbb{G}''_{yy'})$ . ◀

► **Lemma 31.** *Let  $\mathbb{G}$  be a normalized distance graph and  $R \subseteq X$ . Then,  $\llbracket [R]\mathbb{G} \rrbracket = [R]\llbracket \mathbb{G} \rrbracket$ , and  $[R]\mathbb{G}$  can be computed in time  $\mathcal{O}(|X|^2)$ .*

Moreover, the weight  $\mathbb{G}'_{xy}$  of edge  $x \rightarrow y$  in  $[R]\mathbb{G}$  is given by

$$\mathbb{G}'_{xy} = \begin{cases} (\leq, \infty) & \text{if } x \in R_F \\ (\leq, 0) & \text{if } x \in R_H, y \in R \\ \mathbb{G}_{0y} & \text{if } x \in R_H, y \notin R \\ \mathbb{G}_{x0} & \text{if } x \notin R, y \in R \\ \mathbb{G}_{xy} & \text{if } x, y \notin R \end{cases}$$

► **Lemma 32.** [3] *Let  $\mathbb{G}$  be a normalized distance graph. Then,  $\llbracket \vec{\mathbb{G}} \rrbracket = \overline{\llbracket \mathbb{G} \rrbracket}$ , and  $\vec{\mathbb{G}}$  can be computed in time  $\mathcal{O}(|X|^2)$ .*

## 7 Safely reachable GTA zones and their properties

Till now, we have shown properties of distance graphs for GTA zones in general. In this section, we show that GTA zones that are reachable from the initial zone in an  $X_D$ -safe GTA have additional special properties. As in normal TA, we also use the fact the maximal constant occurring in the programs (in the transitions) of a GTA.

Let  $M \in \mathbb{N}$ . We say that a constraint  $x - y \triangleleft c$  is  *$M$ -bounded* if either  $c \in \mathbb{R}$  is such that  $-M \leq c \leq M$  or  $(\triangleleft, c) \in \{(\leq, -\infty), (<, \infty), (\leq, \infty)\}$ . We say that a program is  *$M$ -bounded* if each of its constraints is  $M$ -bounded. Recall the definition of  $X_D$ -safe programs from Definition 10. We say that a program is  $(X_D, M)$ -safe if it is both  $M$ -bounded and  $X_D$ -safe.

► **Definition 33** ( $(X_D, M)$ -safe operations). *The following zone operations are  $(X_D, M)$ -safe:*

- Guard intersection with a safe guard:  $Z \wedge g$ , where  $g$  is  $(X_D, M)$ -safe.
- Reset of a history clock  $x$  or release of a future clock  $x \notin X_D$ :  $[x]Z$ , where  $x \notin X_D$ .
- Release of a future clock  $x \in X_D$  when its value is 0 or  $-\infty$ :  $[x](Z \wedge (x = c))$ , where  $x \in X_D$  and  $c \in \{0, -\infty\}$ .
- Time elapse:  $\vec{Z}$ .

We say that a zone  $Z$  is  $(X_D, M)$ -safely reachable if

- the initialization guard  $g_0$  sets each history clock to either 0 or  $\infty$ .
- if  $Z$  can be obtained starting from the initial zone  $Z_0$  and applying only  $(X_D, M)$ -safe zone operations.

► **Lemma 34.** *If  $\mathcal{A}$  is an  $X_D$ -safe GTA in which the maximum constant used is  $M$ , then its reachable zones are  $(X_D, M)$ -safe.*

In other words, for these systems, we need to only reason about  $(X_D, M)$ -safely reachable zones. When  $X_D$  and  $M$  are clear from the context, we will sometimes abuse notation and just say *safely reachable* zone instead of  $(X_D, M)$ -safely reachable zone, and use *safe* programs, constraints, accordingly.

### 7.1 Valuations of safely reachable zones

Next, we define an equivalence relation  $\simeq$  between valuations that relates valuations that agree on value of history clocks, and satisfy the same set of safe constraints involving non-history clocks.

► **Definition 35.**  $v_1 \simeq v_2$  if  $v_1 \downarrow_{X_H} = v_2 \downarrow_{X_H}$  and, for all  $x, y \in X_F \cup \{0\}$  and for all  $(X_D, M)$ -safe constraints  $y - x \triangleleft c$ , we have  $v_1 \models y - x \triangleleft c$  if and only if  $v_2 \models y - x \triangleleft c$ .

Let  $x \in X_F$  and  $v_1 \simeq v_2$ . Then  $v_1(x) = -\infty$  iff  $v_2(x) = -\infty$ . This is because  $x - 0 \leq -\infty$  is an  $(X_D, M)$ -safe constraint. Further,  $v_1(x) = v_2(x)$  if  $-M \leq v_1(x) \leq 0$ . This is because  $x - 0 \leq v_1(x)$  and  $0 - x \leq -v_1(x)$  are both  $(X_D, M)$ -safe. It follows that  $-\infty < v_1(x) < -M$  if and only if  $-\infty < v_2(x) < -M$  as well.

Definition 35 requires that  $v_1$  and  $v_2$  satisfy the same set of  $(X_D, M)$ -safe constraints involving non-history clocks. We will now show that if  $v_1 \simeq v_2$ , then  $v_1$  and  $v_2$  satisfy the same set of  $(X_D, M)$ -safe constraints involving any pair of clocks.

► **Lemma 36.** *If  $v_1 \simeq v_2$  then, for all  $x, y \in X \cup \{0\}$  and for all  $(X_D, M)$ -safe constraints  $y - x \triangleleft c$ , we have  $v_1 \models y - x \triangleleft c$  if and only if  $v_2 \models y - x \triangleleft c$ .*

**Proof.** The claim follows from Definition 35 for  $(X_D, M)$ -safe constraints involving non-history clocks. Since  $v_1 \simeq v_2$  implies  $v_1 \downarrow_{X_H} = v_2 \downarrow_{X_H}$ , the claim is easy to see for  $(X_D, M)$ -safe constraints (in fact all safe constraints, not just  $(X_D, M)$ -safe constraints) not involving future clocks. Finally, we consider  $M$ -bounded constraints involving a history clock  $y$  and a future clock  $x$ .

- Suppose that  $v_1(x) = v_2(x)$ . In this case, it is easy to see that  $v_1$  and  $v_2$  satisfy the same constraints involving  $y$  and  $x$ .
- Suppose that  $v_1(x) \neq v_2(x)$ . Then, since  $v_1 \simeq v_2$ , this implies that  $-\infty < v_1(x) < -M$  and  $-\infty < v_2(x) < -M$ .
  - $y - x \triangleleft c$ . Suppose  $v_1(y) = v_2(y) = \infty$ . Then,  $v_1(y) - v_1(x) = \infty = v_2(y) - v_2(x)$ . Otherwise,  $0 \leq v_1(y) = v_2(y) < \infty$ . Then,  $M < v_1(y) - v_1(x) < \infty$  and  $M < v_2(y) - v_2(x) < \infty$ . In both cases, we obtain  $v_1 \models y - x \triangleleft c$  if and only if  $v_2 \models y - x \triangleleft c$ .
  - $x - y \triangleleft c$ . We argue similarly, distinguishing two cases depending on whether  $v_1(y) = v_2(y)$  is finite or not. ◀

We will now state a lemma which highlights an important property of future clocks in safely reachable GTA zones - namely, that safely reachable zones are closed under  $\simeq$ -equivalence. The proof follows from the observation that the property is true in the initial zone, and is invariant under the zone operations.

► **Lemma 37.** *For all safely reachable zones  $Z$ , if  $v \in Z$  and  $v \simeq v'$ , then  $v' \in Z$ .*

**Proof.** We will prove that the statement of the lemma is an invariant over safely reachable zones. The property is true if  $Z = \mathbb{V}$  is the set of all valuations. We now show that the property is invariant under all the safe zone operations given in Definition 33. Notice that the initial zone is  $Z_0 = \overline{\mathbb{V}} \cap g_0$  and  $g_0$  is safe. Assume that  $Z$  is a zone that satisfies the property of the lemma.

**Guard intersection.** Let  $g$  be a guard, which is in general a conjunction of (possibly diagonal)  $(X_D, M)$ -safe constraints. We get directly from Lemma 36 that the property continues to hold in the zone  $Z \wedge g$ .

**Release of a clock  $x \in X_F \setminus X_D$ .** Let  $v \in [x]Z$  and  $v' \simeq v$ . We need to show that  $v' \in [x]Z$ . By definition of the release operation, we have  $v = u[x \mapsto \beta]$  for some  $u \in Z$  and  $-\infty \leq \beta \leq 0$ . Let  $u' = v'[x \mapsto u(x)]$ . Since  $Z$  is closed under  $\simeq$ -equivalence (by assumption), it suffices to show that  $u' \simeq u$ . We then have  $u' \in Z$  and  $v' = u'[x \mapsto v'(x)]$ , which implies  $v' \in [x]Z$ .

First, we have  $u \downarrow_{X_H} = v \downarrow_{X_H} = v' \downarrow_{X_H} = u' \downarrow_{X_H}$ . Next, consider a safe constraint  $y - z \triangleleft c$  with  $y, z \in X_F \cup \{0\}$ .

- Suppose that  $y, z \neq x$ . We have  $u(y) - u(z) = v(y) - v(z)$  and  $u'(y) - u'(z) = v'(y) - v'(z)$ . Using  $v \simeq v'$ , we deduce that  $u \models y - z \triangleleft c$  iff  $v \models y - z \triangleleft c$  iff  $v' \models y - z \triangleleft c$  iff  $u' \models y - z \triangleleft c$ .
- Suppose  $y = x \neq z$  (resp.  $y \neq x = z$ ). Since the constraint is  $X_D$ -safe and  $x \in X_F \setminus X_D$  we deduce that  $z = 0$  (resp.  $y = 0$ ). We have  $u(x) = u'(x)$ . We deduce that  $u \models y - z \triangleleft c$  iff  $u' \models y - z \triangleleft c$ .

**Release of a clock  $x \in X_D$ .** Let  $v \in [x](Z \wedge (x = a))$  with  $a = 0$  or  $a = -\infty$  and let  $v' \simeq v$ .

We need to show that  $v' \in [x](Z \wedge (x = a))$ . Note that we have  $u = v[x \mapsto a] \in Z$ . Let  $u' = v'[x \mapsto a]$ . Since  $Z$  is closed under  $\simeq$ -equivalence (by assumption), it suffices to show that  $u' \simeq u$ . We then have  $u' \in Z$  and we get  $v' \in [x](Z \wedge (x = a))$ .

First, we have  $u \downarrow_{X_H} = v \downarrow_{X_H} = v' \downarrow_{X_H} = u' \downarrow_{X_H}$ . Next, consider a  $M$ -bounded constraint  $y - z \triangleleft c$  with  $y, z \in X_F \cup \{0\}$ . We proceed as above if  $y, z \neq x$ , or if  $y = x$  and  $z = 0$ , or if  $y = 0$  and  $z = x$ .

- Suppose  $y \neq 0$  and  $z = x$ . We have  $u(z) = u'(z) = a$ ,  $u(y) = v(y)$  and  $u'(y) = v'(y)$ . We deduce that  $u \models y - z \triangleleft c$  iff  $v(y) - a \triangleleft c$  and  $u' \models y - z \triangleleft c$  iff  $v'(y) - a \triangleleft c$ . Finally, we have  $v(y) - a \triangleleft c$  iff  $v'(y) - a \triangleleft c$ . This is clear when  $a = -\infty$  and it follows from  $v \simeq v'$  when  $a = 0$  ( $y - 0 \triangleleft c$  is a safe constraint).
- We proceed similarly when  $y = x$  and  $z \neq 0$ . We have  $u \models y - z \triangleleft c$  iff  $a - v(z) \triangleleft c$  and  $u' \models y - z \triangleleft c$  iff  $a - v'(z) \triangleleft c$ . Notice that  $v(z) = -\infty$  iff  $v'(z) = -\infty$  since  $v \simeq v'$  and  $z \leq -\infty$  is a safe constraint. We deduce that  $a - v(z) \triangleleft c$  iff  $a - v'(z) \triangleleft c$ .

**Reset.** The reset operation of a history clock  $x$  takes each valuation in  $Z$  and sets  $x$  to 0.

Let  $v \in [x]Z$ . This implies that there exists  $u \in Z$  such that  $v = [x]u$ . Then,  $v(x) = 0$ , and  $u(y) = v(y)$  for all  $y \neq x$ .

Let  $v' \simeq v$ . We need to show that  $v' \in [x]Z$ . Notice that  $v'(x) = v(x) = 0$ . Let  $u' = v'[x \mapsto u(x)]$ . We have  $v' = [x]u'$ . Since  $Z$  is closed under  $\simeq$ -equivalence (by assumption), it suffices to show that  $u' \simeq u$ . We then have  $u' \in Z$  and  $v' = [x]u' \in [x]Z$ .

Since  $v \downarrow_{X_H} = v' \downarrow_{X_H}$ , we first get  $u \downarrow_{X_H} = u' \downarrow_{X_H}$ . It remains to show that  $u$  satisfies an  $(X_D, M)$ -safe constraint  $y - z \triangleleft c$  with  $y, z \in X_F \cup \{0\}$  if and only if  $u'$  also satisfies it. Since  $y, z \neq x$ , we have  $u(y) - u(z) = v(y) - v(z)$  and  $u'(y) - u'(z) = v'(y) - v'(z)$ . Using  $v \simeq v'$ , we deduce that  $u \models y - z \triangleleft c$  iff  $v \models y - z \triangleleft c$  iff  $v' \models y - z \triangleleft c$  iff  $u' \models y - z \triangleleft c$ .

**Time elapse.** Time elapse increases the value of all clocks in  $X$  in a synchronous manner, without affecting the differences between clocks in  $X$ . We will now show that our property is not affected by time elapse.

Suppose that  $v \in \vec{Z}$ , i.e.,  $v = u + \delta$  for some  $u \in Z$  and  $\delta \geq 0$ . Note that this means  $v(z) = (u + \delta)(z) \leq 0$  for all future clocks  $z$ . Let  $v' \simeq v$ . Take  $u' = v' - \delta$ . We show that  $u' \in Z$ , which implies  $v' = u' + \delta \in \vec{Z}$ .

Since  $Z$  is closed under  $\simeq$ -equivalence (by assumption), it suffices to show that  $u' \simeq u$ . Since  $v \downarrow_{X_H} = v' \downarrow_{X_H}$ , we first get  $u \downarrow_{X_H} = u' \downarrow_{X_H}$ . We consider the possible cases for a safe constraint  $y - z \triangleleft c$  with  $y, z \in X_F \cup \{0\}$ .

- If  $y, z \in X_F$ . We have  $u(y) - u(z) = v(y) - v(z)$  and  $u'(y) - u'(z) = v'(y) - v'(z)$ . Using  $v \simeq v'$ , we deduce that  $u \models y - z \triangleleft c$  iff  $v \models y - z \triangleleft c$  iff  $v' \models y - z \triangleleft c$  iff  $u' \models y - z \triangleleft c$ .
- $y = 0 \neq z$  (the case where  $z = 0 \neq y$  follows by a similar argument.)  
Suppose that  $u \models 0 - z \triangleleft c$ , i.e.,  $-u(z) \triangleleft c$ . Since  $u = v - \delta$ , we get  $-v(z) \triangleleft c - \delta$ . Recall that  $v(z) \leq 0$ . Hence, we have  $0 \leq c - \delta \leq c \leq M$ . Further, since  $v \simeq v'$  and  $0 - z \triangleleft c - \delta$  is a safe constraint, we get  $-v'(z) \triangleleft c - \delta$ . Using  $v' = u' + \delta$ , we get  $-u'(z) \triangleleft c$ , i.e.,  $u' \models 0 - z \triangleleft c$ .  $\blacktriangleleft$



► **Remark 38.** The proof crucially uses the fact that  $\mathcal{A}$  is  $X_D$ -safe. For the case of releasing a clock  $x \in X_F \setminus X_D$ , we use the fact that a diagonal constraint involving  $x$  may not use another future clock. For the case of releasing a clock  $x \in X_D$ , we use the fact that the value of the clock must be 0 or  $-\infty$  just before the release.

We remark that the claim does not hold for all zones (which could be reached by releasing a clock in  $X_D$  when its value is not necessarily 0 or  $-\infty$ ). As a non-example, consider Figure 4. Here,  $X_D = \{y, z\}$  and  $M = 1$ . After two iterations of  $a$ , the zone  $Z_2$  reached is  $x = 0 \wedge y = z = -2$ . Pick  $v : x = 0, y = z = -2$  and  $v' : x = 0, y = z = -3$ . Notice that both of them satisfy the same set of  $(X_D, M)$ -safe constraints, but  $v \in Z_2$ ,  $v' \notin Z_2$ . Indeed, the automaton is not  $X_D$ -safe since  $y$  and  $z$  are released arbitrarily.

► **Corollary 39.** *Let  $Z$  be a  $(X_D, M)$ -safely reachable zone and let  $v \in Z$  be a valuation. Let  $n = \max(1, |X_D|)$ .*

1. *Let  $x \in X_F \setminus X_D$ . If  $-\infty < v(x) < -M$  then, for every  $-\infty < \alpha < -M$ , then the valuation  $v' = v[x \mapsto \alpha]$  belongs to  $Z$ .*
2. *Let  $x, y \in X_D \cup \{0\}$ , if  $-\infty < v(x) - v(y) < -nM$  then, for every  $-\infty < \alpha < -nM$ , we have a valuation  $v' \in Z$  with  $v'(x) - v'(y) = \alpha$ .*
3. *Let  $x, y \in X_F \cup \{0\}$ , if  $-\infty < v(x) - v(y) < -nM$  then, for every  $-\infty < \alpha < -nM$ , we have a valuation  $v' \in Z$  with  $v'(x) - v'(y) = \alpha$ .*

**Proof.** 1. We show that  $v' \simeq v$ , and we deduce by Lemma 37 that  $v' \in Z$ . So we have to show that  $v, v'$  satisfy the same  $(X_D, M)$ -safe constraints. This is clear for a constraint which does not involve clock  $x$ . Since  $x \in X_F \setminus X_D$ , a safe constraint involving clock  $x$  must be of the form  $x \triangleleft c$  or  $-x \triangleleft c$ . We conclude easily since the constraint is  $M$ -bounded and  $-\infty < v(x), v'(x) < -M$ .

2. Let  $x, y \in X_D \cup \{0\}$  be such that  $-\infty < v(x) - v(y) < -nM$ . We have  $v(x) \neq -\infty \neq v(y)$ . Hence  $-\infty < v(x) < v(y) - nM < v(y) \leq 0$ . We first give a sufficient condition for a valuation  $v'$  to be equivalent to  $v$ . Consider the following conditions on a valuation  $v'$ :

- a.  $v' \downarrow_{X_H} = v \downarrow_{X_H}$  and  $v' \downarrow_{X_F \setminus X_D} = v \downarrow_{X_F \setminus X_D}$ ,
- b. for all  $z \in X_D$ , we have
  - $v'(z) = v(z)$  if  $v(z) = -\infty$  or  $v(y) \leq v(z) \leq 0$ , and
  - $v'(z) - v'(x) = v(z) - v(x)$  if  $-\infty < v(z) \leq v(x)$ ,
- c. for all  $x', y' \in X_D$  such that  $v(x) \leq v(x') \leq v(y') \leq v(y)$ , we have  $v'(x') - v'(y') = v(x') - v(y')$  or both  $-\infty < v'(x') - v'(y') < -M$  and  $-\infty < v(x') - v(y') < -M$ .

It is not hard to check that if a valuation  $v'$  satisfies the above conditions then  $v' \simeq v$ . We can also check that there is a valuation  $v'$  satisfying the conditions above and such that  $v'(x) - v'(y) = \alpha$ . The property follows.

3. This follows from (2) if  $x, y \in X_D \cup \{0\}$ . We assume below that  $x \in X_F \setminus X_D$  or  $y \in X_F \setminus X_D$ . As above, we have  $-\infty < v(x) < v(y) - nM < v(y) \leq 0$ . Assume that  $x \in X_F \setminus X_D$ . Then  $-\infty < v(x) < -M$ . We apply (1) with  $\alpha' = \alpha + v(y)$ . We get  $v' = v[x \mapsto \alpha'] \in Z$  and  $v'(x) - v'(y) = \alpha$ . Finally, assume that  $x \in X_D$  and  $y \in X_F \setminus X_D$ . We have  $-\infty < v(x) - 0 < -nM$ . We apply (2) to the pair of clocks  $x, 0$  and  $\alpha' = \alpha + v(y)$ . We get  $v' \in Z$  with  $v'(x) - 0 = \alpha + v(y)$ . Notice that from the construction above (2.a) we have  $v'(y) = v(y)$ . Therefore,  $v'(x) - v'(y) = \alpha$ . ◀

► **Remark 40.** Note that in the second and third parts of Corollary 39, we do not maintain the valuation of all the other clocks while changing the particular difference that we are interested in. This is in contrast with the first part, where we change the value of the future clock  $x \in X_F \setminus X_D$ , while keeping the valuation of the other clocks unchanged.



## 7.2 The dagger lemma: from finiteness to boundedness in safely reachable zones

We will now use Corollary 39 to prove the main invariants satisfied by the zones obtained during the enumeration. Essentially, the weights of edges involving non-history clocks come from a finite set which depends on the number of future clocks in  $X_D$  and the maximum constant  $M$  of the automaton. This also induces an invariant on the constraint between a history clock and a future clock.

Before proving Lemma 44, we first state two technical lemmas from [3].

- **Lemma 41** ([3]). 1. Let  $(\triangleleft, c)$  be a weight and  $\alpha \in \overline{\mathbb{R}}$ . Then,
- $\alpha \triangleleft c$  iff  $(\leq, \alpha) \leq (\triangleleft, c)$  iff  $(\leq, 0) \leq (\leq, -\alpha) + (\triangleleft, c)$ ,
  - $\alpha \not\triangleleft c$  iff  $(\triangleleft, c) < (\leq, \alpha)$  iff  $(\leq, -\alpha) + (\triangleleft, c) < (\leq, 0)$  iff  $(\leq, -\alpha) + (\triangleleft, c) \leq (<, 0)$ .
2. Let  $(\triangleleft, c), (\triangleleft', c'), (\triangleleft'', c'')$  be weights with  $(\leq, 0) \leq (\triangleleft, c) + (\triangleleft', c')$ . Then, there exists  $\alpha \in \overline{\mathbb{R}}$  such that  $\alpha \triangleleft c$  and  $-\alpha \triangleleft' c'$ . If in addition we have  $(\triangleleft'', c'') < (\triangleleft, c)$  then there exists such an  $\alpha$  with  $\alpha \not\triangleleft'' c''$ .

► **Lemma 42** ([3]). Let  $\mathbb{G} = \mathbb{G}(Z)$  for a non-empty GTA zone  $Z$ , and let  $x, y \in X \cup \{0\}$  be a pair of distinct nodes and  $\alpha \in \overline{\mathbb{R}}$ . There is a valuation  $v \in \llbracket \mathbb{G} \rrbracket$  with  $v(y) - v(x) = \alpha$  if and only if

1.  $(\leq, \alpha) \leq \mathbb{G}_{xy}$  and  $(\leq, -\alpha) \leq \mathbb{G}_{yx}$ , and
2. if  $x, y \in X$  and  $\alpha \in \mathbb{R}$  is finite then the weights  $\mathbb{G}_{x0}, \mathbb{G}_{0x}, \mathbb{G}_{y0}, \mathbb{G}_{0y}$  are all different from  $(\leq, -\infty)$ , and
3. if  $x, y \in X$  and  $\alpha = -\infty$  then  $\mathbb{G}_{0x} \neq (\leq, -\infty) \neq \mathbb{G}_{y0}$ .

The following lemma extends the corresponding property of [3] by taking into account the initial guard  $g_0$  of a safe GTA.

► **Lemma 43.** Let  $Z$  be a nonempty reachable zone and let  $\mathbb{G}$  be its canonical distance graph.

1. For all  $x \in X_H$ , we have  $\mathbb{G}_{x0} = (\leq, -\infty)$  or  $\mathbb{G}_{0x} \leq (<, \infty)$ .
2. For all  $x, y \in X$ , if  $\mathbb{G}_{xy} = (\leq, -\infty)$  then  $\mathbb{G}_{x0} = (\leq, -\infty)$  or  $\mathbb{G}_{0y} = (\leq, -\infty)$ .

**Proof.** Let  $x \in X_H$  be a history clock. Since  $\mathcal{A}$  is safe, the initial guard  $g_0$  induces either the weight  $(\leq, -\infty)$  for edge  $x \rightarrow 0$  or the weight  $(\leq, 0)$  for edge  $0 \rightarrow x$ . If the weight of  $x \rightarrow 0$  is  $(\leq, -\infty)$ , it stays unchanged until we first apply the reset operation on  $x$ , resulting in the weight  $(\leq, 0)$  for edge  $0 \rightarrow x$ . Then, the weight of edge  $0 \rightarrow x$  may only be increased by the time elapse operation, which sets it to  $(<, \infty)$ . This proves the first property.

For the second property, consider  $x, y \in X$  with  $\mathbb{G}_{xy} = (\leq, -\infty)$  and  $\mathbb{G}_{x0} \neq (\leq, -\infty)$ . We have to show that  $\mathbb{G}_{0y} = (\leq, -\infty)$ . If  $x \in X_H$  then we get  $\mathbb{G}_{0x} \leq (<, \infty)$  by the first property. If  $x \in X_F$  then we have  $\mathbb{G}_{0x} \leq (\leq, 0)$ . In both cases, since  $\mathbb{G}$  is normal, we obtain  $\mathbb{G}_{0y} \leq \mathbb{G}_{0x} + \mathbb{G}_{xy} = (\leq, -\infty)$  and we are done. ◀

We next state the following central lemma that give the  $(\dagger)$  conditions, that says that for all safely reachable zones, the weight of edges of the form  $0 \rightarrow x$ ,  $x \rightarrow 0$  and  $x_1 \rightarrow x_2$  belong to the finite set  $\{(\leq, -\infty), (<, \infty), (\leq, \infty)\} \cup \{(\triangleleft, c) \mid c \in \mathbb{Z} \wedge -nM \leq c \leq nM\}$ , for all future clocks  $x, x_1, x_2 \in X_F$ . In other words, for safely reachable zones, the constraints between non-history clocks come from a finite set.

► **Lemma 44.** Let  $Z$  be a nonempty  $(X_D, M)$ -safely reachable zone and let  $n = \max(1, |X_D|)$ . Then, the normalized distance graph  $\mathbb{G}(Z)$  satisfies the following  $(\dagger)$  conditions:

- †<sub>1</sub> For all  $x \in X_F$ , if  $Z_{xy}$  is finite for some  $y \in X_H \cup \{0\}$ , then  $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$ .
- †<sub>2</sub> For all  $x \in X_F$ , if  $Z_{0x}$  is finite, then  $(<, -nM) \leq Z_{0x} \leq (\leq, 0)$ .

- †<sub>3</sub> For all  $x \in X_H$  and  $y \in X_F$ , if  $Z_{0y}$  is finite, then  $Z_{x0} + (<, -nM) \leq Z_{xy}$ .  
 †<sub>4</sub> For  $x, y \in X_F$ , if  $Z_{xy}$  is finite, then  $(<, -nM) \leq Z_{xy} \leq (\leq, nM)$ .

**Proof.**

†<sub>1</sub> For all  $x \in X_F$ , if  $Z_{xy}$  is finite for some  $y \in X_H \cup \{0\}$ , then  $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$ . In other words, if  $Z_{xy} < (<, \infty)$  for some  $y \in X_H \cup \{0\}$ , then  $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$ .

First, we consider the case where  $y = 0$ . So we assume that  $(\leq, 0) \leq Z_{x0} < (<, \infty)$  is finite. Towards a contradiction, suppose that  $(\leq, nM) < Z_{x0} < (<, \infty)$ . Since  $Z$  is non-empty, we know that  $(\leq, 0) \leq Z_{x0} + Z_{0x}$ . Then, using Lemma 41, we can find  $\alpha \in \mathbb{R}$  such that  $(\leq, \alpha) \leq Z_{x0}$ ,  $(\leq, -\alpha) \leq Z_{0x}$ , and  $nM < \alpha$ . Notice that  $\alpha < \infty$  since  $Z_{x0} < (<, \infty)$ . Further, using Lemma 42, we can get a valuation  $v \in Z$  such that  $0 - v(x) = \alpha$ . Since  $nM < \alpha < \infty$ , this implies  $-\infty < v(x) < -nM$ . Let  $Z_{x0} = (\triangleleft, c)$ . We have  $nM < c < \infty$ . Using Corollary 39(3), we can get a valuation  $v' \in Z$ , such that  $-\infty < v'(x) < -c$ , a contradiction as it violates the constraint  $0 - x \triangleleft c$  in the zone.

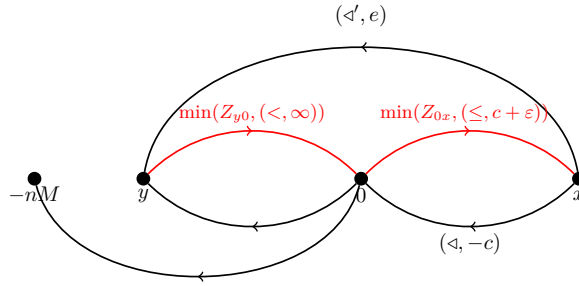
Next, assume that  $Z_{xy} < (<, \infty)$  for some  $y \in X_H$ . Since  $Z$  is normal, we have  $Z_{x0} \leq Z_{xy} + Z_{y0} < (<, \infty)$  as  $Z_{xy} < (<, \infty)$  and  $Z_{y0} \leq (\leq, 0)$ . We now conclude from the first case that  $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$ .

†<sub>2</sub> For all  $x \in X_F$ , if  $Z_{0x}$  is finite, then  $(<, -nM) \leq Z_{0x} \leq (\leq, 0)$ . This means that either  $Z_{0x} = (\leq, -\infty)$  or  $(<, -nM) \leq Z_{0x} \leq (\leq, 0)$ .

Let  $Z_{0x} = (\triangleleft, c)$ . Suppose  $(\leq, -\infty) < Z_{0x} < (<, -nM)$ . We have  $-\infty < c < -nM$ . By Lemma 41, we can find  $\alpha$  such that  $(\leq, \alpha) \leq Z_{0x}$ ,  $(\leq, -\alpha) \leq Z_{x0}$  and  $\alpha \neq -\infty$ . Then, by Lemma 42, we can find  $v \in Z$  with  $v(x) = \alpha$ . We have  $-\infty < v(x) \triangleleft c < -nM$ . Now, using Corollary 39(3), we can get a valuation  $v' \in Z$  such that  $c < v'(x) < -nM$ , which leads to a contradiction as it violates the constraint  $x - 0 \triangleleft c$  in the zone.

†<sub>3</sub> For all  $x \in X_H$  and  $y \in X_F$ , if  $Z_{0y}$  is finite, then  $Z_{x0} + (<, -nM) \leq Z_{xy}$ .

If  $Z_{x0} = (\leq, -\infty)$  then the inequality trivially holds. So, we assume for the rest of the proof that  $Z_{x0} \neq (\leq, -\infty)$ . Since  $Z_{0y}$  is finite, we know that  $Z_{0y} \neq (\leq, -\infty)$ . By Lemma 43, this implies  $Z_{xy} \neq (\leq, -\infty)$ . Let  $Z_{x0} = (\triangleleft, -c)$  and  $Z_{xy} = (\triangleleft', e)$ , as shown in Figure 5. We have  $0 \leq c < \infty$  and  $-\infty < e \leq 0$ .



■ **Figure 5** Distance graph  $\mathbb{G}(Z)$  (without the red edges) and  $\mathbb{G}'$  (with the red edges).

Fix  $\varepsilon > 0$ . Consider the distance graph  $\mathbb{G}'$  obtained from  $\mathbb{G}(Z)$  by setting the weight of  $0 \rightarrow x$  to  $\min(Z_{0x}, (\leq, c + \varepsilon))$ , and the weight of  $y \rightarrow 0$  to  $\min(Z_{y0}, (<, \infty))$ , as shown in Figure 5. It is easy to see that  $\mathbb{G}'$  is also in standard form.

Next, we show that there are no negative cycles in this graph. Since  $Z \neq \emptyset$ , the candidates for being negative must use the new weight  $(\leq, c + \varepsilon)$  of  $0 \rightarrow x$  or the new weight  $(<, \infty)$  of  $y \rightarrow 0$  or both. Then the possible negative cycles are:

- $0 \rightarrow x \rightarrow 0$  with weight  $(\leq, c + \varepsilon) + Z_{x0} = (\leq, c + \varepsilon) + (\triangleleft, -c) = (\triangleleft, \varepsilon)$ , which is not negative, since  $\varepsilon > 0$ .
- $0 \rightarrow y \rightarrow 0$  with weight  $Z_{0y} + (<, \infty)$  which is not negative since  $Z_{0y} \neq (\leq, -\infty)$ ,
- $y \rightarrow 0 \rightarrow x \rightarrow y$  with weight  $(<, \infty) + (\leq, c + \varepsilon) + Z_{xy}$  which is not negative since  $Z_{xy} \neq (\leq, -\infty)$ .

Since  $\mathbb{G}'$  has no negative cycles, Lemma 25 implies  $\llbracket \mathbb{G}' \rrbracket \neq \emptyset$ . Note that  $\llbracket \mathbb{G}' \rrbracket \subseteq \llbracket \mathbb{G}(Z) \rrbracket = Z$ . We know that for all  $v \in \llbracket \mathbb{G}' \rrbracket$ , we have  $c \triangleleft v(x) \leq c + \varepsilon$ .

We will now show that there exists a valuation  $v' \in \llbracket \mathbb{G}' \rrbracket$  such that  $-nM - \varepsilon \leq v'(y)$ . Let  $v \in \llbracket \mathbb{G}' \rrbracket$ . If  $-nM \leq v(y)$ , we let  $v' = v$  and we are done. Otherwise,  $-\infty < v(y) < -nM$ , where the first inequality is due to  $\mathbb{G}'_{y0} \leq (<, \infty)$ . Using Corollary 39(3), there exists a valuation  $v' \in \llbracket \mathbb{G}' \rrbracket$  such that  $v'(y) = -nM - \varepsilon$  since  $\varepsilon > 0$ .

Since  $v' \in \llbracket \mathbb{G}' \rrbracket$ , we have  $c \triangleleft v'(x) \leq c + \varepsilon$  and we obtain  $-nM - c - 2\varepsilon \leq v'(y) - v'(x) \triangleleft' e$ , where the last inequality uses again  $v' \in \llbracket \mathbb{G}' \rrbracket$  and  $\mathbb{G}'_{xy} = (\triangleleft', e)$ . Since this is true for all  $\varepsilon > 0$  we deduce that  $-nM - c \leq e$ . We deduce that  $(<, -nM - c) \leq (\triangleleft', e) = Z_{xy}$ . We conclude using  $(<, -nM - c) = (<, -nM) + (\triangleleft, -c)$ .

†<sub>4</sub> For  $x, y \in X_F$ , if  $Z_{xy}$  is finite, then  $(<, -nM) \leq Z_{xy} \leq (\leq, nM)$ .

Suppose that  $x, y \in X_F$ , and  $Z_{xy} = (\triangleleft, c) \notin \{(\leq, -\infty), (<, \infty), (\leq, \infty)\}$  is finite. Notice that, since  $\mathbb{G}(Z)$  is standard, this implies  $Z_{x0} \neq (\leq, \infty)$ .

The proof proceeds by application of Lemma 42, and for this, when  $x, y \in X$  and  $\alpha \in \mathbb{R}$  is finite, we need to first show that the weights  $Z_{x0}, Z_{0x}, Z_{y0}, Z_{0y}$  are all different from  $(\leq, -\infty)$ . We will now show this.

- We get this for free for weights  $Z_{x0}, Z_{y0}$ , as  $x$  and  $y$  are future clocks.
- Suppose that  $Z_{0x} = (\leq, -\infty)$ . Then, since  $Z$  is non-empty, we get  $Z_{x0} = (\leq, \infty)$ , a contradiction.
- Suppose  $Z_{0y} = (\leq, -\infty)$ . Since  $\mathbb{G}(Z)$  is normal, we have  $Z_{xy} \leq Z_{x0} + Z_{0y} = (\leq, -\infty)$  (since  $Z_{x0} \neq (\leq, \infty)$ ). Again this is a contradiction with  $Z_{xy} \neq (\leq, -\infty)$ .

Thus, we have shown that  $Z_{x0}, Z_{0x}, Z_{y0}, Z_{0y}$  are all different from  $(\leq, -\infty)$ .

Next, we consider the two possibilities for violation of the †<sub>4</sub> condition. We will show that both of them lead to a contradiction.

1.  $(\leq, nM) < Z_{xy} = (\triangleleft, c) < (<, \infty)$ . This implies that  $nM < c < \infty$ .

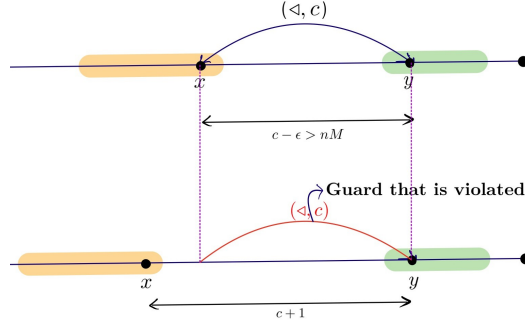
Using Lemma 41, we can find  $\alpha \in \mathbb{R}$  such that  $(\leq, \alpha) \leq Z_{xy}$ ,  $(\leq, -\alpha) \leq Z_{yx}$ , and  $nM < \alpha$ . Notice that  $\alpha \leq c < \infty$ . Further, using Lemma 42, we know that there exists a valuation  $v \in Z$  with  $v(y) - v(x) = \alpha$ . We get  $-\infty < -\alpha = v(x) - v(y) < -nM$  and by Corollary 39(3), we can find a valuation  $v' \in Z$  with  $-\infty < v'(x) - v'(y) = \beta < -c$  (for instance,  $\beta = -c - 1$ ), as illustrated in Figure 6. This is a contradiction with the constraint  $y - x \triangleleft c$  in  $Z$ .

2.  $(\leq, -\infty) < Z_{xy} = (\triangleleft, c) < (<, -nM)$ . This implies that  $-\infty < c < -nM$ .

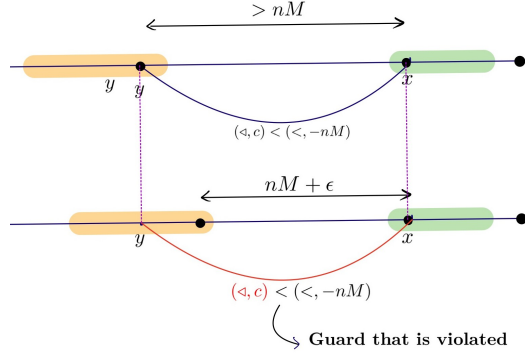
Using Lemma 41, we can find  $\alpha \in \mathbb{R}$  such that  $(\leq, \alpha) \leq Z_{xy}$ ,  $(\leq, -\alpha) \leq Z_{yx}$ , and  $-\infty < \alpha$ . Notice that  $\alpha \leq c < -nM$ . Further, using Lemma 42, we know that there exists a valuation  $v \in Z$  with  $v(y) - v(x) = \alpha$ . Since  $-\infty < \alpha < -nM$ , we use Corollary 39(3) to find a valuation  $v' \in Z$  with  $c < v'(y) - v'(x) = \beta < -nM$  (for instance,  $\beta = \frac{c - nM}{2}$ ), as illustrated in Figure 7. This is a contradiction with the constraint  $y - x \triangleleft c$  in  $Z$ .

Therefore, if  $Z_{xy}$  is finite, then  $(<, -nM) \leq Z_{xy} \leq (\leq, nM)$ . ◀

► **Remark 45.** Note that in each of the dagger conditions in Lemma 44, we can differentiate the cases where the future clock belongs to the set  $X_D$  or not. In particular, when the future clock  $x$  being considered is not in  $X_D$ , the bound can be restricted to  $M$  (instead of  $nM$ ).



■ **Figure 6** Increasing the difference between  $x$  and  $y$  using  $\simeq$ -equivalence.



■ **Figure 7** Shrinking the difference between  $x$  and  $y$  using  $\simeq$ -equivalence.

Thus, we obtain as a corollary that, for event-predicting automata, we do not even need simulation to obtain finiteness of its zone graph.

► **Corollary 46.** *Let  $\mathcal{A}$  be an event-predicting automata with diagonal constraints. Then, the zone graph of  $\mathcal{A}$  is finite.*

## 8 Finiteness of the simulation relation

In this section, we will show that the simulation relation  $\preceq_{\mathcal{A}}$  defined in Section 5 is finite, which implies that the reachability algorithm terminates. Recall that given a GTA  $\mathcal{A}$ , we have an associated map  $\mathcal{G}$  from states of  $\mathcal{A}$  to sets of atomic constraints. Let  $M = \max\{|c| \mid c \in \mathbb{Z} \text{ is used in some constraint of } \mathcal{A}\}$ , the maximal constant of  $\mathcal{A}$ . We have  $M \in \mathbb{N}$  and constraints in the sets  $\mathcal{G}(q)$  use constants in  $\{-\infty, \infty\} \cup \{c \in \mathbb{Z} \mid |c| \leq M\}$ . We will refer to such constraints as  $M$ -bounded integral constraints.

Recall that the simulation relation  $\preceq_{\mathcal{A}}$  was defined on nodes of the zone graph of  $\mathcal{A}$  by  $(q, Z) \preceq_{\mathcal{A}} (q', Z')$  if  $q = q'$  and  $Z \preceq_{\mathcal{G}(q)} Z'$ . This simulation relation  $\preceq_{\mathcal{A}}$  is *finite* if for any infinite sequence  $(q, Z_0), (q, Z_1), (q, Z_2), \dots$  of *safely reachable* nodes in the zone graph of  $\mathcal{A}$  we find  $i < j$  with  $(q, Z_j) \preceq_{\mathcal{A}} (q, Z_i)$ , i.e.,  $Z_j \preceq_{\mathcal{G}(q)} Z_i$ . Notice that we restrict to *safely reachable* zones in the definition above. Our goal now is to prove that the relation  $\preceq_{\mathcal{A}}$  is finite. The structure of the proof is as follows.

1. We proved in Lemma 44 of Section 7 that for any *safely reachable* node  $(q, Z)$  of the zone graph of  $\mathcal{A}$ , the canonical distance graph  $\mathbb{G}(Z)$  satisfies a set of conditions, that we call  $(\dagger)$  conditions, which depend only on the maximal constant  $M$  of  $\mathcal{A}$  and the number of

future clocks in  $\mathcal{A}$ .

2. We will now introduce an equivalence relation  $\sim_M$  of *finite index* on valuations (depending on  $M$  only) and show in Lemma 54 of Section 8 that, if  $G$  is a set of atomic constraints using  $M$ -bounded integral constraints and if  $Z$  is a zone such that its canonical distance graph  $\mathbb{G}(Z)$  satisfies  $(\dagger)$  conditions, then  $\downarrow_G Z$  is a union of  $\sim_M^n$  equivalence classes.

**An equivalence relation of finite index on valuations.** We first define an equivalence relation of finite index  $\sim_M$  on valuations. First, we define  $\sim_M$  on  $\alpha, \beta \in \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$  by  $\alpha \sim_M \beta$  if  $(\alpha \triangleleft c \iff \beta \triangleleft c)$  for all  $(\triangleleft, c)$  with  $\triangleleft \in \{<, \leq\}$  and  $c \in \{-\infty, \infty\} \cup \{d \in \mathbb{Z} \mid |d| \leq M\}$ . In particular, if  $\alpha \sim_M \beta$  then  $(\alpha = -\infty \iff \beta = -\infty)$  and  $(\alpha = \infty \iff \beta = \infty)$ .

Next, for valuations  $v_1, v_2 \in \mathbb{V}$ , we define  $v_1 \sim_M^n v_2$  by two conditions:  $v_1(x) \sim_{nM} v_2(x)$  and  $v_1(x) - v_1(y) \sim_{(n+1)M} v_2(x) - v_2(y)$  for all clocks  $x, y \in X$ . Notice that we use  $(n+1)M$  for differences of values. Clearly,  $\sim_M^n$  is an equivalence relation of finite index on valuations. Using this, we can show that the zones that are reachable in a safe GTA are unions of  $\sim_M^n$ -equivalence classes.

**Distance graph for valuations that simulate a given valuation.** For a valuation  $v$ , we let  $\uparrow_G v = \{v' \in \mathbb{V} \mid v \preceq_G v'\}$ , i.e., the set of valuations  $v'$  which simulate  $v$ . We will define a distance graph, denoted  $\mathbb{G}_G(v)$ , such that  $\llbracket \mathbb{G}_G(v) \rrbracket = \uparrow_G v$ . We remark that  $\llbracket \mathbb{G}_G(v) \rrbracket$  is not really a zone since it may use constants that are not integers.

We will now define the distance graph  $\mathbb{G}_G(v)$  which denotes the set  $\uparrow_G v$ . We will define  $\mathbb{G}_G(v)$  as the intersection of a distance graphs  $\mathbb{G}_v^G$  and a guard  $g_v^G$ .

► **Definition 47.** The distance graph  $\mathbb{G}_v^G$  is defined as follows.

- For each future clock  $x \in X_F$ , we have the edges  $x \xrightarrow{(\leq, -v(x))} 0$  and  $0 \xrightarrow{(\leq, v(x))} x$ .
- For each history clock  $y \in X_H$ , we have
  - the edge  $0 \rightarrow y$  with weight  $(\leq, v(y))$  if there is a constraint  $y \triangleleft c \in G$  with  $c < \infty$  and  $v \models y \triangleleft c$ .
  - the edge  $y \rightarrow 0$  with weight  $(\leq, -v(y))$  if there is a constraint  $c \triangleleft y \in G$  with  $c < \infty$  and  $v \not\models c \triangleleft y$ .

► **Definition 48.** The guard  $g_v^G$  is given by the set of all constraints of the form  $y - x \triangleleft c$  in  $G$  where  $x, y \in X \cup \{0\}$  and  $v \models y - x \triangleleft c$ .

With this definition, we can show that if  $G$  is a set of atomic constraints containing both  $x \leq 0$  and  $0 \leq x$  for each clock  $x \in X_F$ , then  $\uparrow_G v = \llbracket \mathbb{G}_v^G \rrbracket \cap \llbracket g_v^G \rrbracket$ .

► **Lemma 49.** Let  $G$  be a set of constraints such that for all future clock  $x \in X_F$  we have both  $x \leq 0$  and  $0 \leq x$  in  $G$ . We have  $\uparrow_G v = \llbracket \mathbb{G}_v^G \rrbracket \cap \llbracket g_v^G \rrbracket$ .

**Proof.**  $\subseteq$ : Let  $v'$  be such that  $v \preceq_G v'$ . By definition of the simulation relation, for all  $g' = y - x \triangleleft c$  in  $G$  such that  $v \models g'$ , we have  $v' \models g'$ . Hence,  $v' \models g_v^G$ . Next, let  $x \in X_F$  be a future clock. If  $v(x) = -\infty$  then for all  $0 \leq \delta < \infty$  we have  $v + \delta \models x \leq 0$ . Since  $v \preceq_G v'$  we get  $v' + \delta \models x \leq 0$ , which implies  $v'(x) = -\infty = v(x)$ . Otherwise, let  $0 \leq \delta = -v(x) < \infty$ . Since  $v + \delta \models x \leq 0 \wedge 0 \leq x$  and  $v \preceq_G v'$ , we get  $v' + \delta \models x \leq 0 \wedge 0 \leq x$ . We deduce that  $v'(x) = v(x)$ . Therefore,  $v'$  satisfies the edges  $x \xrightarrow{(\leq, -v(x))} 0$  and  $0 \xrightarrow{(\leq, v(x))} x$  of  $\mathbb{G}_v^G$ .

Now, let  $x \in X_H$  be a history clock. Assume that  $v \models x \triangleleft c$  for some  $x \triangleleft c$  in  $G$  with  $0 \leq c < \infty$ . Using  $v \preceq_{x \triangleleft c} v'$ , we get  $v'(x) \leq v(x)$ . Hence,  $v'$  satisfies the edge  $0 \xrightarrow{(\leq, v(x))} x$  of  $\mathbb{G}_v^G$ . Assume that  $v \not\models c \triangleleft x$  for some  $c \triangleleft x$  in  $G$  with  $0 \leq c < \infty$ . Again, we obtain  $v(x) \leq v'(x)$  from  $v \preceq_{c \triangleleft x} v'$ . Hence,  $v'$  satisfies the edge  $x \xrightarrow{(\leq, -v(x))} 0$  of  $\mathbb{G}_v^G$ . Thus,  $v'$  satisfies all constraints of  $\mathbb{G}_v^G$ , i.e.,  $v' \in \llbracket \mathbb{G}_v^G \rrbracket$ .

$\supseteq$ : Let  $v \in \llbracket \mathbb{G}_v^G \rrbracket$  with  $v \models g_v^G$ . Let  $g' = y - x \triangleleft c$  be a diagonal constraint in  $G$  with  $x, y \in X$ . If  $v \models g'$  then  $g'$  is in  $g_v^G$  and  $v' \models g'$ . Therefore,  $v \preceq_{g'} v'$ .

Now, let  $g'$  be a non-diagonal constraint on a future clock, i.e.,  $x \triangleleft c$  or  $c \triangleleft x$  with  $x \in X_F$ . Since  $v \in \llbracket \mathbb{G}_v^G \rrbracket$  we get  $v'(x) = v(x)$  and we deduce that  $v \preceq_{g'} v'$ . Let  $g'$  be an upper non-diagonal constraint  $x \triangleleft c$  on a history clock  $x \in X_H$ . If  $v \not\models g'$  then  $v \preceq_{g'} v'$ . If  $v \models g'$  and  $c$  is finite then we get  $v'(x) \leq v(x)$  from the edge  $0 \xrightarrow{\leq, v(x)} x$  of  $\mathbb{G}_v^G$ . Hence,  $v \preceq_{g'} v'$ . If  $g'$  is  $x < \infty$  and  $v \models g'$  then  $g'$  is in  $g_v^G$  and we get  $v'(x) < \infty$  from  $v' \models g_v^G$ . We deduce that  $v \preceq_{g'} v'$ . If  $g'$  is  $x \leq \infty$  then  $g'$  is equivalent to *true* and  $v \preceq_{g'} v'$ . Let  $g'$  be a lower non-diagonal constraint  $c \triangleleft x$  on a history clock  $x \in X_H$ . If  $v \models g'$  then  $g'$  is in  $g_v^G$  and we get  $v' \models g'$ . Therefore,  $v \preceq_{g'} v'$ . Assume now that  $v \not\models g'$ . If  $c$  is finite then we get  $v(x) \leq v'(x)$  from the edge  $x \xrightarrow{\leq, -v(x)} 0$  of  $\mathbb{G}_v^G$ . We deduce that  $v \preceq_{g'} v'$ . If  $g'$  is  $\infty < x$  then  $g'$  is equivalent to *false* and  $v \preceq_{g'} v'$ . Lastly, when  $g'$  is  $\infty \leq x$  and  $v(x)$  is finite. Then, for all  $0 \leq \delta < \infty$  we have  $v + \delta \not\models g'$ . Therefore,  $v \preceq_{g'} v'$ .  $\blacktriangleleft$

- **Remark 50.** 1.  $\mathbb{G}_v^G$  is in standard form, but not necessarily in normal form.  
 2.  $\llbracket \mathbb{G}_v^G \rrbracket$  is non-empty, since  $v \in \llbracket \mathbb{G}_v^G \rrbracket$ .  
 3.  $g_v^G$  is a conjunction of atomic constraints, each of which is  $(X_D, M)$ -safe.

Further, we show that if  $\mathbb{G}_v^G \cap Z'$  is empty and  $\mathbb{G}'$  is the normalized distance graph of  $Z'$ , then there is a small witness, i.e., a negative cycle in  $\min(\mathbb{G}_v^G, \mathbb{G}')$  containing at most three edges, and belonging to one of three specific forms. This also gives us an efficient simulation check for GTA zone graphs.

► **Lemma 51.** *Let  $v$  be a valuation,  $Z'$  a non-empty reachable event zone with canonical distance graph  $\mathbb{G}'$  and  $G$  a set of atomic constraints. Then,  $\mathbb{G}_v^G \cap Z'$  is empty iff there is a negative cycle in one of the following forms:*

1.  $0 \rightarrow x \rightarrow 0$  with  $0 \rightarrow x$  from  $\mathbb{G}_v^G$  and  $x \rightarrow 0$  from  $\mathbb{G}'$ ,
2.  $0 \rightarrow y \rightarrow 0$  with  $0 \rightarrow y$  from  $\mathbb{G}'$  and  $y \rightarrow 0$  from  $\mathbb{G}_v^G$ , and
3.  $0 \rightarrow x \rightarrow y \rightarrow 0$ , with weight of  $x \rightarrow y$  from  $\mathbb{G}'$  and the others from  $\mathbb{G}_v^G$ .

**Proof.** Since the distance graph  $\mathbb{G}'$  is in normal form, it has no negative cycle. Similarly,  $\mathbb{G}_v^G$  has no negative cycle since  $v \in \mathbb{G}_v^G \neq \emptyset$ . We know that  $\mathbb{G}_v^G \cap Z' = \emptyset$  iff there is a (simple) negative cycle in  $\min(\mathbb{G}_v^G, \mathbb{G}')$ . Since  $\mathbb{G}'$  is in normal form, we may restrict to negative cycles which do not use two consecutive edges from  $\mathbb{G}'$ . Further, note that all edges of  $\mathbb{G}_v^G$  are adjacent to node 0. Hence, if a simple cycle uses an edge from  $\mathbb{G}'$  which is adjacent to 0, it consists of only two edges  $0 \rightarrow x \rightarrow 0$ , one from  $\mathbb{G}'$  and one from  $\mathbb{G}_v^G$ . Otherwise, the simple cycle is of the form  $0 \rightarrow x \rightarrow y \rightarrow 0$  where the edge  $x \rightarrow y$  is from  $\mathbb{G}'$  and the other two edges are from  $\mathbb{G}_v^G$ .  $\blacktriangleleft$

► **Lemma 52.** *Let  $v \sim_M^n v'$  and  $G$  be a set of  $M$ -bounded integral constraints. Then, we have the following*

1.  $g_{v'}^G = g_v^G$ .
2. The graph  $\mathbb{G}_{v'}^G$  is obtained by replacing the weights  $(\leq, v(x))$  (resp.  $(\leq, -v(x))$ ) by  $(\leq, v'(x))$  (resp.  $(\leq, -v'(x))$ ) in the graph  $\mathbb{G}_v^G$ .

**Proof.** 1.  $g_{v'}^G = g_v^G$  is easy to see from the definition of  $\mathbb{G}_{v'}^G$  and  $\mathbb{G}_v^G$ , and the fact that  $v \sim_{(n+1)M} v'$ .

2. For a future clock  $x \in X_F$ , this is easy to see from the definition for edges  $x \rightarrow 0$  and  $0 \rightarrow x$  adjacent to  $x$ .

We consider now edges adjacent to history clocks  $y \in X_H$ .



- Consider the edge  $0 \rightarrow y$ . If its weight is  $(\leq, v(y))$  in  $\mathbb{G}_v^G$  then there is some  $y \triangleleft c \in G$  with  $c < \infty$  and  $v(y) \triangleleft c$ . Since  $v \sim_{(n+1)M} v'$ , we deduce that  $v'(y) \triangleleft c$  and the edge  $0 \rightarrow y$  has weight  $(\leq, v'(y))$  in  $\mathbb{G}_{v'}^G$ .
- Consider the edge  $y \rightarrow 0$ . If its weight is  $(\leq, -v(y))$  in  $\mathbb{G}_v^G$ , then there is some  $c \triangleleft y \in G$  with  $c < \infty$  and  $c \not\triangleleft v(y)$ . Since  $v \sim_{(n+1)M} v'$ , we deduce that  $c \not\triangleleft v'(y)$  and the edge  $y \rightarrow 0$  has weight  $(\leq, -v'(y))$  in  $\mathbb{G}_{v'}^G$ .  $\blacktriangleleft$

Using all the results above, we can now show that the zones that are reachable in a safe GTA are unions of  $\sim_M^n$ -equivalence classes.

► **Remark 53.** Before we state the lemma, we list some properties that we will use extensively in the proof of the lemma.

1.  $-b \triangleleft a$  iff  $-a \triangleleft b$  iff  $(\leq, 0) \leq (\triangleleft, a + b)$ .
2.  $a \triangleleft b$  iff  $\neg(b \tilde{\triangleleft} a)$  where  $\tilde{\leq} = <$  and  $\tilde{<} = \leq$ .
3.  $\alpha \sim_M \beta$  and  $c \in \mathbb{R}$  is such that  $-M \leq c \leq M$  or  $(\triangleleft, c) \in \{(\leq, -\infty), (<, \infty), (\leq, \infty)\}$ , then,  $c \triangleleft \alpha$  iff  $c \triangleleft \beta$ . This is because
  - $c \triangleleft \alpha$  iff  $\neg(\alpha \tilde{\triangleleft} c)$  by (2) above.
  - $\neg(\alpha \tilde{\triangleleft} c)$  iff  $\neg(\beta \tilde{\triangleleft} c)$  by definition of  $\sim_M$  equivalence.
  - $\neg(\beta \tilde{\triangleleft} c)$  iff  $c \triangleleft \beta$  by (2) above.

► **Lemma 54.** Let  $G$  be a set of  $X_D$ -safe  $M$ -bounded integral constraints which contains both  $x \leq 0$  and  $0 \leq x$  for each future clock  $x \in X_F$ . Let  $Z$  be a zone with a canonical distance graph  $\mathbb{G}(Z)$  satisfying the  $(\dagger)$  conditions of Lemma 44. Let  $v_1, v_2 \in \mathbb{V}$  be valuations with  $v_1 \sim_M^n v_2$ . Then,  $v_1 \in \downarrow_G Z$  iff  $v_2 \in \downarrow_G Z$ .

**Proof.** Notice that  $v \in \downarrow_G Z$  iff  $\uparrow_G v \cap Z \neq \emptyset$ . We need to show that  $\uparrow_G v_1 \cap Z \neq \emptyset$  iff  $\uparrow_G v_2 \cap Z \neq \emptyset$ . Using the characterization of up-sets given by Lemma 49, this amounts to  $Z \cap g_{v_1}^G \cap \llbracket \mathbb{G}_{v_1}^G \rrbracket \neq \emptyset$  iff  $Z \cap g_{v_2}^G \cap \llbracket \mathbb{G}_{v_2}^G \rrbracket \neq \emptyset$ .

Further, since  $v_1 \sim_M^n v_2$ , using Lemma 52, it follows that  $g_{v_2}^G = g_{v_1}^G$ . Let  $Z' = Z \cap g_{v_2}^G = Z \cap g_{v_1}^G$ . If  $Z'$  is empty then the equivalence holds. Otherwise, let  $\mathbb{G}(Z')$  be the normalized distance graph of  $Z'$ . Note that since  $Z$  was an  $(X_D, M)$ -safely reachable zone and  $g_{v_1}^G$  is a conjunction of atomic constraints, each of which is  $(X_D, M)$ -safe, it follows that  $Z'$  is an  $(X_D, M)$ -safely reachable zone. As a consequence, the  $\dagger$  conditions of Lemma 44 apply to  $Z'$ .

In the rest of the proof, we will now work with the zone  $Z'$  (using its normalized distance graph representation  $\mathbb{G}(Z')$ ) and the standard distance graphs  $\mathbb{G}_{v_1}^G$  and  $\mathbb{G}_{v_2}^G$ . The proof proceeds by contradiction. We assume that  $\uparrow_G v_1 \cap Z \neq \emptyset$  and  $\uparrow_G v_2 \cap Z = \emptyset$ . This is equivalent to  $Z' \cap \llbracket \mathbb{G}_{v_1}^G \rrbracket \neq \emptyset$  and  $Z' \cap \llbracket \mathbb{G}_{v_2}^G \rrbracket = \emptyset$ . By Lemma 51, we can find a negative cycle  $C_2$  using one edge from  $\mathbb{G}(Z')$  and one or two edges from  $\mathbb{G}_{v_2}^G$ . By Lemma 52, we have a corresponding cycle  $C_1$  using the same edge from  $\mathbb{G}(Z')$  and the same one or two edges from  $\mathbb{G}_{v_1}^G$  (with weights using  $v_1$  instead of  $v_2$ ). The cycle  $C_1$  is not negative since  $Z' \cap \llbracket \mathbb{G}_{v_1}^G \rrbracket \neq \emptyset$ .

The rest of the proof involves a case analysis of the various forms that the cycle  $C_2$  can take, which we provide below. We consider the different cases.

1. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(y))} y \xrightarrow{Z'_{y0}} 0$  for some history clock  $y \in X_H$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(y))} y \xrightarrow{Z'_{y0}} 0$ .

Since we have the edge  $0 \xrightarrow{(\leq, v_1(y))} y$  in  $\mathbb{G}_{v_1}^G$ , there is a constraint  $y \triangleleft' c'$  in  $G$  with  $c' < \infty$  and  $v_1(y) \triangleleft' c'$ . We deduce that  $0 \leq v_1(y) \leq M$ .

Let  $Z'_{y0} = (\triangleleft, c)$ . Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(y))$ , which is equivalent to  $-c \triangleleft v_1(y)$ . Using  $0 \leq v_1(y) \leq M$  and  $v_1 \sim_M^n v_2$  we deduce that  $-c \triangleleft v_2(y)$ . This is equivalent to  $(\leq, 0) \leq (\triangleleft, c + v_2(y))$ , a contradiction with  $C_2$  being a negative cycle.



2. Cycle  $C_2 = 0 \xrightarrow{Z'_{0y}} y \xrightarrow{(\leq, -v_2(y))} 0$  for some history clock  $y \in X_H$ .

We have  $C_1 = 0 \xrightarrow{Z'_{0y}} y \xrightarrow{(\leq, -v_1(y))} 0$ .

Since we have the edge  $y \xrightarrow{(\leq, -v_1(y))} 0$  in  $\mathbb{G}_{v_1}^G$ , there is a constraint  $c' \triangleleft y$  in  $G$  with  $c' < \infty$  and  $c' \triangleleft v_1(y)$ . We deduce that  $0 \leq v_1(y) \leq M$ .

Let  $Z'_{0y} = (\triangleleft, c)$ . Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c - v_1(y))$ , which is equivalent to  $v_1(y) \triangleleft c$ . Using  $v_1 \sim_M^n v_2$  and  $0 \leq v_1(y) \leq M$ , we deduce that  $v_2(y) \triangleleft c$ . This is equivalent to  $(\leq, 0) \leq (\triangleleft, c - v_2(y))$ , a contradiction with  $C_2$  being a negative cycle.

3. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(x))} x \xrightarrow{Z'_{x0}} 0$  for some future clock  $x \in X_F$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(x))} x \xrightarrow{Z'_{x0}} 0$ .

Since  $C_2$  is negative, we have  $Z'_{x0} \neq (\leq, \infty)$ . Also, if  $Z'_{x0} = (<, \infty)$  then we must have  $v_2(x) = -\infty$ , which implies  $v_1(x) = -\infty$  since  $v_1 \sim_M^n v_2$ , a contradiction with  $C_1$  being non-negative. Hence,  $Z'_{x0} = (\triangleleft, c)$  is finite and by  $(\dagger_1)$ , we infer  $0 \leq c \leq nM$ .

Since  $C_1$  is not negative, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(x))$ , which is equivalent to  $-c \triangleleft v_1(x)$ . Using  $v_1 \sim_M^n v_2$  and  $0 \leq c \leq nM$  we deduce that  $-c \triangleleft v_2(x)$ . This is equivalent to  $(\leq, 0) \leq (\triangleleft, c + v_2(x))$ , a contradiction with  $C_2$  being a negative cycle.

4. Cycle  $C_2 = 0 \xrightarrow{Z'_{0x}} x \xrightarrow{(\leq, -v_2(x))} 0$  for some future clock  $x \in X_F$ .

We have  $C_1 = 0 \xrightarrow{Z'_{0x}} x \xrightarrow{(\leq, -v_1(x))} 0$ .

Let  $Z'_{0x} = (\triangleleft, c)$ . Since  $C_2$  is negative, we deduce that  $v_2(x) \neq -\infty$ . Using  $v_1 \sim_M^n v_2$ , we infer  $v_1(x) \neq -\infty$ . Since  $C_1$  is not negative, we get  $Z'_{0x} \neq (\leq, -\infty)$ . From  $(\dagger_2)$ , we infer  $(<, -nM) \leq Z'_{0x} \leq (\leq, 0)$  and  $-nM \leq c \leq 0$ .

Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c - v_1(x))$ , which is equivalent to  $v_1(x) \triangleleft c$ . Using  $v_1 \sim_M^n v_2$  and  $-nM \leq c \leq 0$ , we deduce that  $v_2(x) \triangleleft c$ . This is equivalent to  $(\leq, 0) \leq (\triangleleft, c - v_2(x))$ , a contradiction with  $C_2$  being a negative cycle.

5. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(y))} y \xrightarrow{Z'_{yx}} x \xrightarrow{(\leq, -v_2(x))} 0$  for some history clock  $y \in X_H$  and future clock  $x \in X_F$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(y))} y \xrightarrow{Z'_{yx}} x \xrightarrow{(\leq, -v_1(x))} 0$ .

Let  $Z'_{yx} = (\triangleleft, c)$ . As in case 1 above, we get  $0 \leq v_1(y) \leq M$ . From the fact that the cycle  $0 \xrightarrow{(\leq, v_1(y))} y \xrightarrow{Z'_{y0}} 0$  is not negative, we get  $(\leq, -M) \leq Z'_{y0}$ . Since  $C_2$  is negative, we get  $v_2(x) \neq -\infty$ . Using  $v_1 \sim_M^n v_2$ , we infer  $v_1(x) \neq -\infty$ . From the fact that the cycle  $0 \xrightarrow{Z'_{0x}} x \xrightarrow{(\leq, -v_1(x))} 0$  is not negative, we deduce  $Z'_{0x} \neq (\leq, -\infty)$ . Using  $(\dagger_3)$  we obtain

$$(\leq, -M) + (<, -nM) \leq Z'_{y0} + (<, -nM) \leq Z'_{yx} = (\triangleleft, c)$$

and we deduce that  $-(n+1)M \leq c \leq 0$ .

Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(y) - v_1(x))$ , which is equivalent to  $-c \triangleleft v_1(y) - v_1(x)$ . Using  $v_1 \sim_M^n v_2$  and  $-(n+1)M \leq c \leq 0$  we deduce that  $-c \triangleleft v_2(y) - v_2(x)$ . We conclude as in the previous cases.

6. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_2(y))} 0$  for some history clock  $y \in X_H$  and future clock  $x \in X_F$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_1(y))} 0$ .

Since  $C_2$  is negative but not  $C_1$ , we get first  $Z'_{xy} \neq (\leq, \infty)$  and then  $v_1(x) \neq -\infty$ . As in case 2 above, we get  $0 \leq v_1(y) \leq M$ . We deduce that  $Z'_{xy} = (\triangleleft, c) < (<, \infty)$  and  $c \neq \infty$ .

From  $(\dagger_1)$  we obtain  $Z'_{x0} \leq (\leq, nM)$ . Since  $0 \xrightarrow{(\leq, v_1(x))} x \xrightarrow{Z'_{x0}} 0$  is not a negative cycle, we get  $-nM \leq v_1(x) \leq 0$ . Finally, we obtain  $0 \leq v_1(y) - v_1(x) \leq (n+1)M$ .

Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(x) - v_1(y))$ , which is equivalent to  $v_1(y) - v_1(x) \triangleleft c$ . Using  $v_1 \sim_M^n v_2$  and  $0 \leq v_1(y) - v_1(x) \leq (n+1)M$ , we deduce that  $v_2(y) - v_2(x) \triangleleft c$ . We conclude as in the previous cases.

7. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_2(y))} 0$  with  $x \neq y$  for future clocks  $x, y \in X_F$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_1(y))} 0$ .

Since  $C_2$  is negative but not  $C_1$ , using  $v_1 \sim_M^n v_2$  we get successively  $Z'_{xy} \neq (\leq, \infty)$ ,  $v_2(y) \neq -\infty \neq v_1(y)$ ,  $v_1(x) \neq -\infty \neq v_2(x)$ , and finally  $(\leq, -\infty) < Z'_{xy} < (\leq, \infty)$ .

Let  $Z'_{xy} = (\triangleleft, c)$ . From  $(\dagger_4)$ , we deduce that  $-nM \leq c \leq nM$ .

Since  $C_1$  is not a negative cycle, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(x) - v_1(y))$ , which is equivalent to  $v_1(y) - v_1(x) \triangleleft c$ . Using  $v_1 \sim_M^n v_2$  and  $-nM \leq c \leq nM$ , we deduce that  $v_2(y) - v_2(x) \triangleleft c$ . We conclude as in the previous cases.

8. Cycle  $C_2 = 0 \xrightarrow{(\leq, v_2(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_2(y))} 0$  with  $x \neq y$  for history clocks  $x, y \in X_H$ .

We have  $C_1 = 0 \xrightarrow{(\leq, v_1(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_1(y))} 0$ .

As in case 1 above, we get  $0 \leq v_1(x) \leq M$ . As in case 2 above, we get  $0 \leq v_1(y) \leq M$ .

We obtain  $-M \leq v_1(y) - v_1(x) \leq M$ .

Let  $Z'_{xy} = (\triangleleft, c)$ . Since  $C_1$  is not negative, we get  $(\leq, 0) \leq (\triangleleft, c + v_1(x) - v_1(y))$ , which is equivalent to  $v_1(y) - v_1(x) \triangleleft c$ . Using  $v_1 \sim_M^n v_2$  and  $-M \leq v_1(y) - v_1(x) \leq M$ , we deduce that  $v_2(y) - v_2(x) \triangleleft c$ . We conclude as in the previous cases.

Notice that we have crucially used the “ $(n+1)M$ ” occurring in the definition of  $v_1 \sim_M^n v_2$  (as  $v_1(x) - v_1(y) \sim_{(n+1)M} v_2(x) - v_2(y)$ ) in the cases where we deal with cycles containing one future clock and one history clock (Cases 5 and 6). ◀

Finally, from Lemmas 44 and 54, we obtain our main theorem of the section.

► **Theorem 55.** *The simulation relation  $\preceq_{\mathcal{A}}$  is finite if  $\mathcal{A}$  is safe.*

**Proof.** Let  $(q, Z_0), (q, Z_1), (q, Z_2), \dots$  be an infinite sequence of *reachable* nodes in the zone graph of  $\mathcal{A}$ . By Lemma 44, for all  $i$ , the distance graph  $\mathbb{G}(Z_i)$  in canonical form satisfies conditions  $(\dagger)$ .

The set  $\mathcal{G}(q)$  contains only  $X_D$ -safe and  $M$ -bounded integral constraints. Let  $G$  be  $\mathcal{G}(q)$  together with the constraints  $x \leq 0$  and  $0 \leq x$  for each future clock  $x \in X_F$ . From Lemma 54 we deduce that for all  $i$ ,  $\downarrow_G Z_i$  is a union of  $\sim_M^n$ -classes. Since  $\sim_M^n$  is of finite index, there are only finitely many unions of  $\sim_M^n$ -classes. Therefore, we find  $i < j$  with  $\downarrow_G Z_i = \downarrow_G Z_j$ , which implies  $Z_j \preceq_G Z_i$ . Since  $\mathcal{G}(q) \subseteq G$ , this also implies  $Z_j \preceq_{\mathcal{G}(q)} Z_i$ . ◀

## 9 Experimental evaluation

We have implemented a prototype that takes as input a GTA, as given in Definition 6, and applies our reachability algorithm, in the open source tool TCHECKER [29]. To do so, we extend TCHECKER to allow clocks to be declared as one of *normal*, *history*, *prophecy*, or *timer*, and extend the syntax of edges to allow arbitrary interleaving of guards and clock changes (reset/release). Our tool, along with the benchmarks used in this paper, is publicly available and can be downloaded from <https://github.com/anirjoshi/GTA-Model>. We present selected results in Table 1, with further details in Appendix A.

First, we consider timed automata models from standard benchmarks [41, 19, 36]. Despite the overhead induced by our framework (e.g., maintaining general programs on transitions), we are only slightly worse off w.r.t. running time than the standard algorithm, while visiting and storing the same number of nodes. We illustrate this in rows 1-3 of Table 1 by providing a

Sl. No.	Models	$\mathcal{G}$ -Sim			GTA Reach		
		Visited nodes	Stored nodes	Time in sec.	Visited nodes	Stored nodes	Time in sec.
1	Dining Phi. (6)	5480	5480	4.911	5480	5480	6.410
2	FDDI (10)	10219	459	10.139	10219	459	16.797
3	Fischer (10)	447598	260998	29.1574	447598	260998	34.6517
4	ToyECA(10000, 4)	150049	49	4.22	3	3	0.0003
5	ToyECA(5000, 6)	315193	193	15.572	3	3	0.0006
6	ToyECA(1000, 100)	TIMEOUT			3	3	0.877
7	ToyECA(50000, 120)	TIMEOUT			3	3	1.52
8	Fire-alarm-pattern(5)	—			46	46	0.027
9	CSMACD-bounded(1)	—			34	26	0.0054
10	CSMACD-bounded(4)	—			4529	2068	2.597
11	ABP-prop1(1)	—			114	114	0.038
12	ABP-prop2(1)	—			168	168	0.026

■ **Table 1** Experimental results obtained by running our prototype implementation and, when possible, the standard reachability algorithm using  $\mathcal{G}$ -simulation implemented in TCHECKER. Both implementations use a breadth-first search with simulation. For each model, we give the parameters in parenthesis - for ToyECA, we explain the parameterization in Appendix A, while for others, we report the number of concurrent processes. All experiments were run on an Ubuntu machine with an Intel-i5 7th Generation processor and 8GB RAM, and timeout set to 60 seconds.

comparison of our tool with the implementation of the state-of-the-art zone-based reachability algorithm using  $\mathcal{G}$ -simulation introduced in [24, 25, 26].

Next, we consider models belonging to the class of ECA without diagonal constraints. We remark that ours is the first implementation of a reachability algorithm that can operate on the whole class of ECA directly. We compare against an implementation that first translates the ECA into a timed automaton using the translation proposed in [9], and then runs the state-of-the-art reachability algorithm of [24, 25, 26] on this timed automaton. From rows 4-7 of Table 1, we observe significant improvements, both in terms of running time as well as number of visited nodes and stored nodes w.r.t. the standard approach.

Finally, in Rows 8-12, we consider the unified model GTA. As already pointed out, model-checking an event-clock specification  $\varphi$  over a timed automaton model  $\mathcal{A}$  can be reduced to the reachability on the product of the TA  $\mathcal{A}$  and the ECA representing  $\neg\varphi$ . In this spirit, our implementation allows the model to use any combination of *normal* clocks, *history* clocks, *prophecy* clocks or *timers* and moreover, permits diagonal guards between any of these clocks. To the best of our knowledge, no existing tool allows all these features. We emphasize this by the — in the  $\mathcal{G}$ -Sim column of Table 1.

We model simple but useful properties using event-clocks, and check these properties on some standard models from literature such as Alternating-bit-protocol(ABP) [35], CSMACD [41] and Fire-alarm [37]. Note that for the benchmark Fire-alarm-pattern, the specification is modelled using an ECA with diagonals. As a consequence, the product automaton that we check reachability on contains normal clocks and event-clocks. Here, we consider the following ECA specification: no three  $a$ 's occur within  $k$  time units. The negation of this property can be easily modeled by an ECA with two states and a transition on  $a$  with the diagonal constraint  $\overleftarrow{a} - \overrightarrow{a} \leq k$ , where  $\overleftarrow{a}$  is the history clock recording time since the previous occurrence of  $a$ , and  $\overrightarrow{a}$  is a future clock predicting the time to the next  $a$  occurrence.

When reading an  $a$ , the quantity  $\overleftarrow{a} - \overrightarrow{a}$  gives the distance between the next and the previous occurrence. This language is used in [17] to observe that ECA with diagonals are more expressive than ECA. Finally, we remark that the model of ABP contains timers. For a more detailed discussion of the model and specifications in these benchmarks, see Appendix A.

In conclusion, as can be seen from the experimental results in Table 1, we are able to demonstrate the full power of our reachability algorithm for the unified model of generalized timed automata.

## 10 Conclusion

The success of timed automata verification can safely be attributed to the advances in the zone-based technology over the last three decades. In fact, [22], the precursor to the seminal works [7, 8], already laid the foundations for zones by describing the Difference-Bounds-Matrices (DBM) data structure. Our goal in this work has been to unify timing features defined in different timed models, while at the same time retain the ability to use efficient state-of-the-art algorithms for reachability. To do so, we have equipped the model with two kinds of clocks, history and future, and modified the transitions to contain a program that alternates between a guard and a change to the variables. For the algorithmic part, we have adapted the  $\mathcal{G}$ -simulation framework to this powerful model. The main challenge was to show finiteness of the simulation in this extended setting. To aid the practical use of this generic model, we have developed a prototype implementation that can answer reachability for GTA. We remark that decidability for GTA comes via zones, and not through regions. In fact, since we generalize event-clock automata, we do not have a finite region equivalence for GTA [28].

We conclude with some interesting avenues for future work. An immediate future work is to use generalized timed automata for model-checking timed specifications over real-time systems. Further, the complexity and expressivity of safe GTA are natural interesting theoretical open questions, but we believe they are not obvious. Both these questions are answered in the timed automata literature using regions. However, we cannot have a region equivalence for our model, since even for the subclass of ECA, it was shown that no finite bisimulation is possible. In particular, it would be interesting to investigate if it is possible to have a translation from safe GTA to timed automata. Note that even if such a translation exists, it is likely to incur an exponential blowup since even the translation from ECA to TA costs an exponential. Coming to the complexity of the reachability problem for safe GTA, it is easy to see that our procedure runs in EXPSPACE, as we have shown that each reachable zone is a union of equivalence classes of a finite index (see Lemma 54). On the other hand, PSPACE-hardness is inherited from timed automata [7, 5]. Closing the complexity gap is open. We note that even in timed automata, the precise complexity of the simulation based reachability algorithm is difficult to analyze, but its selling point is that it works well in practice. Finally, we would also like to investigate liveness verification for GTA, in particular what future clocks bring us when we consider the setting of  $\omega$ -words.

---

## References

- 1 S. Akshay, Benedikt Bollig, and Paul Gastin. Event clock message passing automata: a logical characterization and an emptiness checking algorithm. *Formal Methods Syst. Des.*, 42(3):262–300, 2013.
- 2 S. Akshay, Paul Gastin, R. Govind, and B. Srivathsan. Simulations for event-clock automata. In *CONCUR*, volume 243 of *LIPIcs*, pages 13:1–13:18, 2022.

- 3 S. Akshay, Paul Gastin, R. Govind, and B. Srivathsan. Simulations for event-clock automata. *CoRR*, abs/2207.02633, 2022.
- 4 S. Akshay, Paul Gastin, and Karthik R. Prakash. Fast zone-based algorithms for reachability in pushdown timed automata. In *CAV (1)*, volume 12759 of *LNCS*, pages 619–642, 2021.
- 5 Rajeev Alur. *Techniques for automatic verification of real-time systems*. PhD thesis, Stanford University, 1991.
- 6 Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.
- 7 Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *ICALP*, volume 443 of *LNCS*, pages 322–335. Springer, 1990.
- 8 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
- 9 Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211(1-2):253–273, 1999.
- 10 Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *ACPN 2003*, volume 3098 of *LNCS*, pages 87–124. Springer, 2003.
- 11 Arthur J. Bernstein and Paul K. Harter Jr. Proving real-time properties of programs with temporal logic. In *SOSP*, pages 1–11. ACM, 1981.
- 12 Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods Syst. Des.*, 24(3):281–320, 2004.
- 13 Patricia Bouyer and Fabrice Chevalier. On conciseness of extensions of timed automata. *J. Autom. Lang. Comb.*, 10(4):393–405, 2005.
- 14 Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *CAV (1)*, volume 9779 of *LNCS*, pages 513–530. Springer, 2016.
- 15 Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theor. Comput. Sci.*, 321(2-3):291–345, 2004.
- 16 Patricia Bouyer, Paul Gastin, Frédéric Herbreteau, Ocan Sankur, and B. Srivathsan. Zone-based verification of timed automata: Extrapolations, simulations and what next? In *FORMATS*, volume 13465 of *LNCS*, pages 16–42. Springer, 2022.
- 17 Laura Bozzelli, Angelo Montanari, and Adriano Peron. Taming the complexity of timeline-based planning over dense temporal domains. In *FSTTCS*, volume 150 of *LIPIcs*, pages 34:1–34:14, 2019.
- 18 Laura Bozzelli, Angelo Montanari, and Adriano Peron. Complexity issues for timeline-based planning over dense time under future and minimal semantics. *Theor. Comput. Sci.*, 901:87–113, 2022.
- 19 Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool KRONOS. In *Hybrid Systems*, volume 1066 of *LNCS*, pages 208–219. Springer, 1995.
- 20 Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In *TACAS*, volume 1384 of *LNCS*, pages 313–329. Springer, 1998.
- 21 Jacobus Willem de Bakker, C Huizing, Willem-Paul de Roever, and G Rozenberg. *Real-Time: Theory in Practice: REX Workshop, Mook, The Netherlands. Proceedings*, volume 600. 1992.
- 22 David L. Dill. *Timing assumptions and verification of finite-state concurrent systems*. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 197–212. Springer, 1989.
- 23 Deepak D’Souza and Nicolas Tabareau. On timed automata with input-determined guards. In *FORMATS/FTRTFT*, volume 3253 of *LNCS*, pages 68–83. Springer, 2004.
- 24 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in timed automata with diagonal constraints. In *CONCUR*, volume 118 of *LIPIcs*, pages 28:1–28:17, 2018.
- 25 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Fast algorithms for handling diagonal constraints in timed automata. In *CAV (1)*, volume 11561 of *LNCS*, pages 41–59, 2019.

- 26 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability for updatable timed automata made faster and more effective. In *FSTTCS*, volume 182 of *LIPICs*, pages 47:1–47:17, 2020.
- 27 Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. Event clock automata: From theory to practice. In *FORMATS*, volume 6919 of *LNCS*, pages 209–224. Springer, 2011.
- 28 Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. On regions and zones for event-clock automata. *Formal Methods Syst. Des.*, 45(3):330–380, 2014.
- 29 F. Herbretreau and G. Point. TChecker. <https://github.com/fredher/tchecker>, v0.2 - April 2019.
- 30 Frédéric Herbretreau, B. Srivathsan, Thanh-Tung Tran, and Igor Walukiewicz. Why liveness for timed automata is hard, and what we can do about it. *ACM Trans. Comput. Log.*, 21(3):17:1–17:28, 2020.
- 31 Frédéric Herbretreau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In *LICS*, pages 375–384. IEEE Computer Society, 2012.
- 32 ITU-TS Recommendation Z.120: Message Sequence Chart (MSC '99), 1999.
- 33 Bengt Jonsson and Frits Vaandrager. Learning mealy machines with timers. Technical report, 2018.
- 34 Ron Koymans, Jan Vytopyl, and Willem P. de Roever. Real-time programming and asynchronous message passing. In *PODC*, pages 187–197. ACM, 1983.
- 35 James F. Kurose and Keith W. Ross. *Computer networking - a top-down approach featuring the internet*. Addison-Wesley-Longman, 2001.
- 36 Denis Lugiez, Peter Niebert, and Sarah Zennou. A partial order semantics approach to the clock explosion problem of timed automata. *Theor. Comput. Sci.*, 345(1):27–59, 2005.
- 37 Marco Muñoz, Bernd Westphal, and Andreas Podelski. Timed automata with disjoint activity. In *FORMATS*, volume 7595 of *LNCS*, pages 188–203. Springer, 2012.
- 38 Jean-François Raskin and Pierre-Yves Schobbens. The logic of event clocks - decidability, complexity and expressiveness. *J. Autom. Lang. Comb.*, 4(3):247–282, 1999.
- 39 Maria Sorea. Tempo: A model checker for event-recording automata. Technical report, In Proceedings of RT-Tools'01, 2001.
- 40 B. Srivathsan. Reachability in timed automata. *ACM SIGLOG News*, 9(3):6–28, 2022.
- 41 Stavros Tripakis and Sergio Yovine. Analysis of timed systems using time-abstraction bisimulations. *Formal Methods Syst. Des.*, 18(1):25–68, 2001.
- 42 Jianhua Zhao, Xuandong Li, and Guoliang Zheng. A quadratic-time dbm-based successor algorithm for checking timed automata. *Inf. Process. Lett.*, 96(3):101–105, 2005.

## A Appendix for Section 9

### Benchmarks for GTA in Table 1

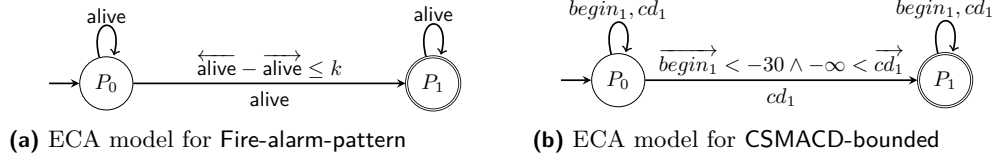
In each of the benchmarks, we consider a model for which we check a property. For each of these properties, we propose an event-clock automaton modelling the negation of the property. Then, whether the model satisfies the property may be checked by checking reachability on the product where the model synchronizes with the ECA on the actions of the ECA.

Note that we only provide here the ECA modelling the negation of the property that we want to check, and not the full product of the model and the ECA. We provide the model for the Alternating-bit-protocol (ABP) (Figure 9). The models for Fire-alarm and CSMACD are the standard models as given in [37] and [30], respectively.

While depicting event-clock automata, we will use  $\overleftarrow{e}$  to denote the history clock recording time since the previous occurrence of event  $e$ , and  $\overrightarrow{e}$  to denote the prophecy clock predicting the negative of the time to the next  $e$ .

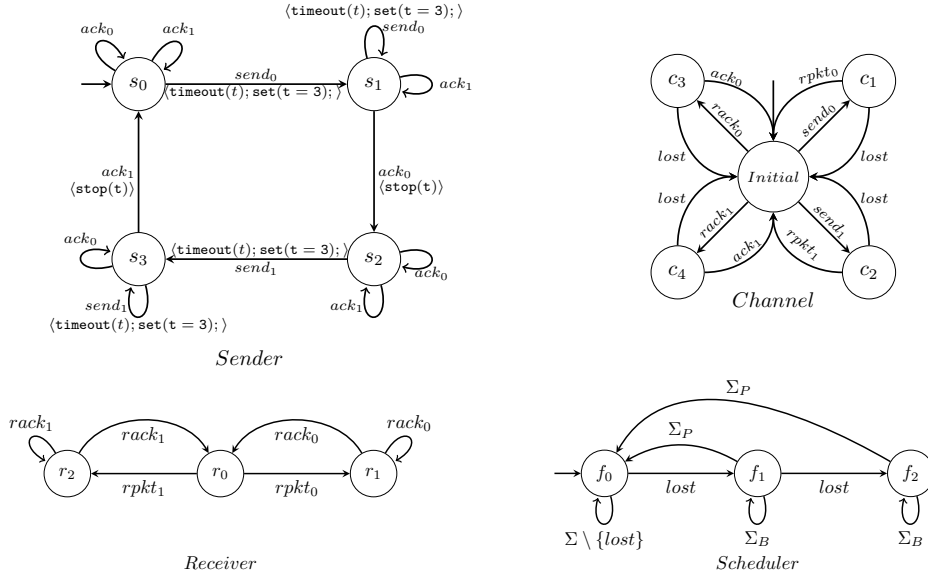
**Fire-alarm-pattern.** We consider the Fire-alarm model from [37]. The model is a network consisting of  $n$  processes, referred to as Sensor processes, and a server process. Each process





in the model is modelled using a timed automaton. Here, we check the property that no three *alive* actions are executed by the process  $\text{Sensor}_1$  in  $k$  time units. The negation of this property can be modeled by the ECA in Figure 8a with two states and a transition on *alive* with the diagonal constraint  $\overleftarrow{\text{alive}} - \overrightarrow{\text{alive}} \leq k$ . When reading an action *alive*, the quantity  $\overleftarrow{\text{alive}} - \overrightarrow{\text{alive}}$  gives the distance between the next and the previous occurrence.

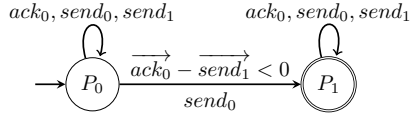
**CSMACD-bounded.** We consider the CSMACD model given in [30]. The model is a network consisting of  $n$  processes, referred to as **Station** processes, and a central **Bus** process. The property that we check here is: after each *detected collision* (modelled using a  $cd$  action), except the last one,  $\text{Station}_1$  sends a message (modelled using a  $\text{begin}_1$  action) in 30 time units. The negation of this property can be modeled by the ECA of Figure 8b. When reading an action  $cd_1$ , the constraints (1)  $\overrightarrow{\text{begin}_1} < -30$  says that  $\text{begin}_1$  (which denotes  $\text{Process}_1$  sending a message) cannot be seen within 30 time units, (2)  $-\infty < \overrightarrow{cd_1}$  says that this is not the last  $cd$  event (and therefore, at least one more collision will be detected in the future).



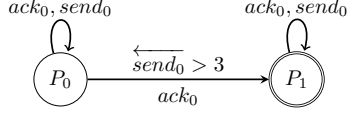
■ **Figure 9** Alternating-bit-protocol

**Alternating-bit-protocol.** We consider a variant of the Alternating-bit-protocol [35] as depicted in Figure 9. We model sending a packet with identifier  $i \in \{0, 1\}$  in Sender with the action  $\text{send}_i$ , and receiving an acknowledgement with identifier  $i \in \{0, 1\}$  in Sender with the action  $\text{ack}_i$ . The Sender uses a timer  $t$ . Recall that the timer operations are (1)  $\text{set}(t=3)$  that sets timer  $t$  to value  $c$ , (2)  $\text{timeout}(t)$  that checks whether  $t$  is 0, (3)  $\text{stop}(t)$  that forgets the value of the timer and sets it  $-\infty$  (to indicates that it is unused.) Note that in the automaton *Scheduler* in Figure 9,  $\Sigma_B = \{\text{send}_0, \text{send}_1, \text{rack}_0, \text{rack}_1\}$ ,  $\Sigma_P = \{\text{rpkt}_0, \text{rpkt}_1, \text{ack}_0, \text{ack}_1\}$ ,  $\Sigma = \Sigma_B \sqcup \Sigma_P \sqcup \{\text{lost}\}$ .

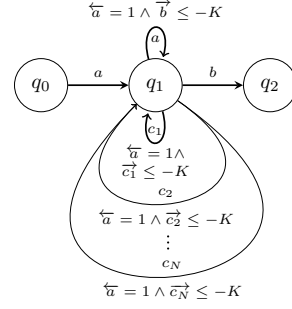




■ Figure 10 ABP-prop1



■ Figure 11 ABP-prop2



■ Figure 12 ToyECA( $K, N$ )

**ABP-prop1:** The property checks the following for the Sender process of ABP: after the sending  $send_0$ , the sender should receive an  $ack_0$  before sending  $send_1$ . We model the negation of this property using an ECA as given in Figure 10.

**ABP-prop2:** The property that after sending a  $send_0$ , the sender must receive an  $ack_0$  within 3 time units. We model the negation of this property with an ECA given in Figure 11. Note that this property does not hold for the sender in ABP.

## Synthetic Benchmarks in Table 1

**ToyECA( $K, N$ ):** As depicted in Figure 12, ToyECA( $K, N$ ) has two parameters -  $K$ , which is the maximal constant and  $N$ , which is the number of  $c_i$  loops (on state  $q_1$ ), in the automaton.

From  $q_0$ , on a transition  $a$ , the automaton goes to  $q_1$ . From  $q_1$ , it can either take the loop  $a$ , after which the  $b$  action taking it to state  $q_2$  can be taken only after  $K$  time units. Alternately, from  $q_1$  a sequence of distinct  $c_i$  loops can be taken in zero-time (because of the  $\overleftarrow{a} = 1$  guard). Note that two  $c_i$  actions can be taken only at an interval of greater than  $K$  time units.

From Table 1, we observe an order of magnitude improvement, both in terms of running time as well as number of visited and stored nodes w.r.t. the standard approach. Recall that there is a blow up (both in the number of states and clocks) while converting an ECA to a timed automaton. The effect caused by the blow up in clocks also affects the time taken for each zone operation. Further, note that even when we increase the parameters  $K, N$ , despite an increase in runtime, the number of visited and stored nodes does not increase - this is because even though we explore more nodes of the zone graph (because of an increase in number of transitions), these explorations lead to nodes that are subsumed by nodes that have already been visited.