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

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.

1 Introduction

The idea of adding real-time dynamics to formal verification models started as a hot topic of research in the 1980s [5, 10]. 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 [12, 21, 32]. 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 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 [13, 16]. Furthermore, considering more sophisticated rates of evolution for clocks gives the yet another well-established model of hybrid automata [6].

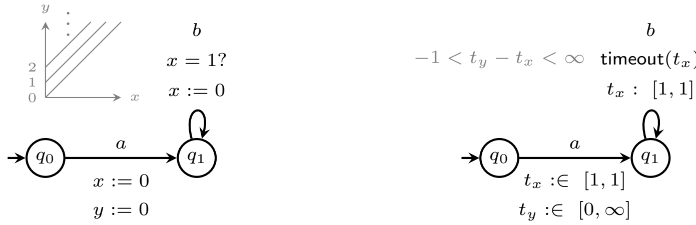


Fig. 1: An automaton with clocks on left, and timers on right for same constraints.

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 [17,38]). 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 ∞. 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 [21]. 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 [36]. 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 φ 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 [21] 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 [26, 27], an extrapolation approach using maximal constants has been studied for ECA. However, it has been observed that simulation-based techniques are both more effective [13, 15] and efficient [4, 23–25] 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 [14] 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 [18]. 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 [1,22,36]. ECA with diagonal constraints have been well-studied, such as in the context of timeline based planning [18, 19]. 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 [37], 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. Section 7 is dedicated to finiteness. Finally, we provide our experimental results in Section 8 and conclude with Section 9.

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. 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*. History clocks always have a non-negative value and can increase arbitrarily along with time. Future clocks always have a non-positive value and can only increase until their values hit 0. History clocks simulate the usual clocks in timed automata and recording clocks of event-clock automata (ECA), and future clocks simulate timers and prophecy clocks of ECA. Both these clocks can take a special "undefined value" which marks that they are inactive. To deal with this naturally, we consider an extension of the reals with $+\infty$ and $-\infty$ as in [2]. The difference here is that we also have the so-called diagonal constraints.

Extending clock constraints. 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. 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: 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 $x \leq c \wedge c \leq x$. Constraints of the form $x - y \triangleleft c$ will be called *atomic constraints*. A constraint of the form $x - y \triangleleft c$ is a *diagonal* (resp. *non-diagonal*) constraint if $x, y \neq 0$ (resp. $x = 0$ or $y = 0$).

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 extended addition, the minus operation does not distribute over addition[1].

Extending valuations. A valuation of clocks is a function $v \colon X \cup \{0\} \mapsto \overline{\mathbb{R}}$ 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. $v(x) \in \{-\infty, +\infty\}$). 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 φ in $\Phi(X)$, denoted as $v \models \varphi$, when v satisfies all atomic constraints in φ.

By definition, 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 $y - x \triangleleft c$, we have

- $v \models y - x \triangleleft c$ iff $v(y) < +\infty = v(x)$ or ($v(x)$ 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 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.

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. Formally, given a set of

[1] 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 $(-a) + (-b) = +\infty$.

Fig. 2: A transition of TA (left) and of a GTA (right)

clocks $R \subseteq X$, we define $[R]v$ as $\{v' \in \mathbb{V} \mid v'(x) = 0 \ \forall\ x \in R \cap X_H$ 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. Then, 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 while keeping the value of all other clocks unchanged. For $W \subseteq \mathbb{V}$, we let $[R]W = \bigcup_{v \in W}[R]v$. We have $[R' \cup R'']W = [R']([R'']W)$.

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 2, a transition reads a letter, checks a guard $g \in \Phi(X_H)$ and then resets a subset R of (history) clocks. But in any one transition only a pair of guard, reset is performed and one cannot interleave them.

We generalize this to our setting with history and future clocks but also to allow arbitrary interleaving of guards and changes (to model this with a TA one may use a sequence of multiple transitions without delays in-between.) Formally, an *instantaneous timed program* is generated by the following grammar:

$$\mathrm{prog} := \mathrm{guard} \mid \mathrm{change} \mid \mathrm{prog}; \mathrm{prog}$$

where guard $= g \in \Phi(X)$ and change $= [R]$ for some $R \subseteq X$. While guard and change are atomic programs, prog; 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, prog) in Figure 2 (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 $v, v' \in \mathbb{V}$, prog $\in$ Programs we define $(v, v') \models \mathrm{prog}$, more conveniently written as $v \xrightarrow{\mathrm{prog}} v'$, inductively:

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

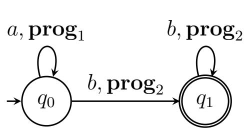
Now, we have all the pieces necessary to define our generalized model.

Definition 1 (Generalized timed automata). *A generalized timed automata $\mathcal{A}$ 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, Σ 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, and $\Delta \subseteq (Q \times \Sigma \times Programs \times Q)$ is a finite set of transitions. Δ contains transitions of the form $(q, a, prog, q')$, where q is the source state, q' is the target state, a is the action triggering the transition, and prog is the instantaneous timed program that is executed in sequence (from left to right) while firing the transition.*

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 $\mathbb{TS}_{\mathcal{A}}$ 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 $\mathbb{TS}_{\mathcal{A}}$ are of two forms: (1) *delay transition*: $(q, v) \xrightarrow{\delta} (q, v+\delta)$ if $(v+\delta) \models X_F \leq 0$, and (2) *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 $\text{prog} = \text{prog}_1; \ldots; \text{prog}_n$ can be taken from (q, v) if there are valuations $v_1, \ldots, v_n$ such that $v \xrightarrow{\text{prog}_1} v_1 \xrightarrow{\text{prog}_2} \cdots \xrightarrow{\text{prog}_n} v_n = v'$. A *run* of a GTA is a finite sequence of transitions from an initial configuration of $\mathbb{TS}_{\mathcal{A}}$. 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 3 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 8 based on experiments.



History clocks: $\{x\}$, **Future clocks:** $\{y, z\}$
$\mathbf{prog}_1 : \langle x = 0;\ [y];\ y = z - 1;\ [z]; z = y \rangle$ $\mathbf{prog}_2 : \langle x = 1;\ [x] \rangle$
Initial condition: $y = z = 0$ **Final condition:** ***true***

Fig. 3: Example of a GTA

Timed automata. Timed automata (TA) of Alur-Dill [8] can be modeled as a GTA as follows: (1) The set of states of the GTA is the same as the set of states of the TA. (2) There are no future clocks in the GTA and its history clocks are the clocks of the TA. (3) 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 $\text{prog} = \langle g; [R] \rangle$. (4) Initially, all clocks must be 0, captured by setting $g_0 = (X_H = 0)$. (5) The final guard is empty: $g_f = \texttt{True}$.

Event-clock automata. Event-clock automata (ECA) of [9] can be modeled as a GTA as follows: (1) The set of states of the GTA is the same as the set of states of the ECA. (2) For each $a \in \Sigma$, the GTA has a history clock $\overleftarrow{a}$ and a future clock $\overrightarrow{a}$. (3) 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 $\text{prog} := \langle (\overrightarrow{a} = 0); [\overrightarrow{a}]; g; [\overleftarrow{a}] \rangle$. (4) At initialization, history clocks must be undefined (set to ∞), captured by $g_0 = (X_H = \infty)$. (5) At acceptance, all future clocks must be undefined, i.e., $g_f = (X_F = -\infty)$.

Automata with timers. The third model we consider is that of automata with timers. Timers are timing constructs that are started/initialized 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 is called *timeout*. 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 [30] and Mealy machines with timers [31].

In our setting, a timer can be seen as a specific instance of a future clock. More precisely Automata with timers ($A_{⧖}$) can be modeled as GTA as follows: (1) The set of states of the GTA is the same as the set of states of $A_{⧖}$. (2) The future clocks of GTA are the timers of $A_{⧖}$ and there are no history clocks. Initially, the timers are undefined, captured by $g_0 = (X_F = -\infty)$ and $g_f = \texttt{True}$. (4) A transition of $A_{⧖}$ with action a from q to q' is encoded as $q \xrightarrow{a,\text{prog}} q'$ with:

- 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 $\texttt{timeout}(x)$, then $\text{prog} = \langle x = 0; [x]; x = -\infty \rangle$.
- if the transition stops timer x, then $\text{prog} = \langle [x]; x = -\infty \rangle$.

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.

4 The reachability problem for GTA

We are interested in the *reachability problem* for GTA: given a GTA $\mathcal{A}$, does it have 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 [26, 27], 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 [26,27] 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 3).

Theorem 2. *Reachability for GTA is undecidable.*

Proof. We reduce 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: (1) $\mathtt{zero}_C = \langle y_C = 0\rangle$ (2) $\mathtt{inc}_C = \langle [z]; z = y_C - 1; [y_C]; y_C = z; [z]; z = 0\rangle$ (3) $\mathtt{dec}_C = \langle y_C \leq -1; [z]; z = y_C + 1; [y_C]; y_C = z; [z]; z = 0\rangle$. In programs $\mathtt{inc}_C$ and $\mathtt{dec}_C$, 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 $\mathtt{inc}_C$ 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.

Definition 3 (Safe GTA). *Let $X_D \subseteq X_F$ be a subset of future clocks.*

A program $prog = \langle g_1; [R_1]; g_2; [R_2]; \dots; g_k; [R_k]; g_{k+1}\rangle$ 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 ∞.

Observe that the three examples discussed in Section 3 are safe. 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 $X_D = \emptyset$. The importance of safety is the following theorem which is the center-piece of this article.

Theorem 4. *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 [25] for timed automata with diagonal constraints to GTA. Diagonal constraints offer succinct modeling [14], but are quite challenging to handle efficiently in zone-based algorithms, and have led to pitfalls in the past: [13] 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 [25] 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 [25] 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 7 for the termination argument.

$\mathcal{G}$-simulation and the static analysis 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 is to define a simulation relation on the semantics of $\mathcal{A}$, i.e., on $\mathbb{TS}(\mathcal{A})$. In the subsequent sections we will lift this to zones and show its finiteness. A simulation relation on $\mathbb{TS}(\mathcal{A})$ 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:

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

Notice that in the definition above, we *do not* restrict δ 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 δ from any given valuation always results in a valuation. But this is crucial for the proof of Theorem 5 below.

Intuitively, the preorder above is a simulation wrt the constraints in G even after time elapse. But we need this to also be a simulation wrt 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 wrt 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 prog, a set of constraints $G' = \mathsf{pre}(\text{prog}, G)$ 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):

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

where $\mathsf{split}(g)$ 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 [23–25], 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:

$$\mathcal{G}(q) = \{x \leq 0 \mid x \in X_F\} \cup \bigcup_{q \xrightarrow{a,\text{prog}} q'} \mathsf{pre}(\text{prog}, \mathcal{G}(q')) \tag{1}$$

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 then show that $\preceq_{\mathcal{A}}$ defined above is indeed a simulation relation.

Theorem 5. *The relation $\preceq_{\mathcal{A}}$ is a simulation on the transition system $\mathbb{TS}_{\mathcal{A}}$.*

Zones for GTA and the zone graph computation. Roughly, *zones* [11] 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*, or simply *zones*, which are special sets of valuations of GTA. A GTA zone is a set of valuations satisfying a conjunction of constraints of the form $y - x \triangleleft c$, where $x, y \in X \cup \{0\}$, $c \in \overline{\mathbb{Z}}$ 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.

GTA zone operations. Let g be a guard, $R \subseteq X$ a set of clocks and Z a GTA zone.

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

Successor computation. We can show that starting from a zone Z, the successors after the above operations are also zones (see Thm 15). A guard g can be seen as yet another zone and hence guard intersection is just an intersection operation between two zones. Similarly, the change operation preserves 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 $\text{prog} = \text{prog}_1; \ldots; \text{prog}_n$, where each prog_i is an atomic program, i.e., a guard or a change. Then we define zones $Z_1, \ldots, Z_{n+1}$ where, $Z_1 = Z$, $Z' = \overrightarrow{Z_{n+1}}$, and for each $1 \leq i \leq n$, $Z_{i+1} = Z_i \cap g_i$ if prog_i is a guard g_i, and $Z_{i+1} = [R_i]Z_i$ if prog_i is a change $[R_i]$.

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

Definition 6 (GTA zone graph). *Given a GTA $\mathcal{A}$, its* GTA zone graph, denoted $\mathsf{GZG}(\mathcal{A})$*, 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 $(q_0, \overrightarrow{Z_0})$ where q_0 is the initial state and Z_0 is the set of all valuations which satisfy the* initial constraint *g_0: Z_0 is given by $g_0 \wedge \big(X_F \leq 0\big) \wedge \big(X_H \geq 0\big)$. For every node (q, Z) and every transition $t := (q, a, prog, q')$ of $\mathcal{A}$, 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 ECA, 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 $(q_0, \overrightarrow{Z_0})$ to (q, Z) in $\mathsf{GZG}(A)$. Thus, reachability of a final state in $\mathcal{A}$ reduces to checking reachability of an accepting node in $\mathsf{GZG}(A)$. However, as in the case of zone graphs for timed automata, $\mathsf{GZG}(A)$ 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 7 (Simulation on GTA zones and finiteness). *Let $\preceq$ be a simulation relation on $\mathbb{TS}(\mathcal{A})$. 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 $(q, Z_1), (q, Z_2), \ldots$ *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 the simulation $\preceq_{\mathcal{A}}$ from Theorem 5 to truncate nodes that are smaller with respect to the simulation. In Section 7, we will show that $\preceq_{\mathcal{A}}$ is finite when $\mathcal{A}$ is safe, which implies that the reachability algorithm terminates. But before that we discuss the issue of implementability.

6 Computing with GTA zones using distance graphs

To implement the reachability algorithm described above, we will view zones as *distance graphs*, as is usually done in the literature [11].

Recall the notion of weights $\mathcal{C} = \{(\triangleleft, c) \mid c \in \overline{\mathbb{R}}$ and $\triangleleft \in \{\leq, <\}$. An order relation $<$ between weights is defined as $(\triangleleft, c) < (\triangleleft', c')$ when either (1) $c < c'$, or (2) $c = c'$ and $\triangleleft$ is $<$ while $\triangleleft'$ is $\leq$. Note that since $(<, -\infty) < (\leq, -\infty) < (\triangleleft, c) < (<, \infty) < (\leq, \infty)$ for all $c \in \mathbb{R}$, this relation is a total order and therefore min of a finite set of weights is well defined. We also use the commutative and associative sum operation on weights defined in [3]. If $c, c' \in \mathbb{R}$ are finite, the definition is as usual: $(\triangleleft, c) + (\triangleleft', c') = (\triangleleft'', c + c')$ where $\triangleleft'' = \leq$ if $\triangleleft = \triangleleft' = \leq$ and $\triangleleft'' = <$ otherwise. Infinite weights α, β from the list $(<, +\infty), (\leq, -\infty), (\leq, +\infty), (<, -\infty)$ are all 'absorbants' wrt. weaker weights: $\alpha + \beta = \beta + \alpha = \alpha$ if α is stronger than β (i.e., α is listed after β). Also, $\alpha + (\triangleleft, c) = \alpha$ if $c \in \mathbb{R}$ is finite.

A distance graph $\mathbb{G}$ is a weighted directed graph without self-loops, with vertex set $X \cup \{0\} = X_F \cup X_H \cup \{0\}$, and edges labeled with weights from $\mathcal{C} \setminus \{(<, -\infty)\}$. We define its semantics $[\![\mathbb{G}]\!] := \{v \in \mathbb{V} \mid v \models y - x \triangleleft c$ for all edges $x \xrightarrow{\triangleleft c} y$ in $\mathbb{G}\}$. The weight of edge $x \to y$ is denoted $\mathbb{G}_{xy}$ and we set $\mathbb{G}_{xy} = (\leq, \infty)$ if there is no edge $x \to y$. 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 $(\leq, 0)$.

In classical timed automata, the significance of distance graphs stems from the observation that a distance graph has no negative cycles iff its semantics is non-empty. This property does not immediately hold for distance graphs over the extended algebra [3, Section 4.2] However, we can convert a distance graph $\mathbb{G}$ (in time polynomial in number of clocks) into a *standard form* where this characterization continues to hold. First, we set $\mathbb{G}'_{0x} = \min(\mathbb{G}_{0x}, (\leq, 0))$ for $x \in X_F$ and $\mathbb{G}'_{x0} = \min(\mathbb{G}_{x0}, (\leq, 0))$ for $x \in X_H$. Moreover, if $x \in X_F$ 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 $\mathbb{G}$, it is equivalent to $\mathbb{G}$ (i.e., $[\![\mathbb{G}']\!] = [\![\mathbb{G}]\!]$) and it has a negative cycle iff its semantics $[\![\mathbb{G}']\!]$ is empty [3].

Now, suppose $\mathbb{G}'$ (in standard form) has no negative cycles, then we construct $\mathbb{G}''$ by replacing the weight of an edge $x \to y$ 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.

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* $\mathbb{G}(Z)$: $[\![\mathbb{G}(Z)]\!] = Z$ and $\mathbb{G}(Z)$ is minimal among the standard graphs G with $[\![G]\!] = Z$. We denote by Z_{xy} the weight of the edge $x \to y$ in $\mathbb{G}(Z)$.

[2] contains the algorithms for the zone operations when there are no diagonal constraints. Successor computation can be done in $\mathcal{O}(|X|^2 \cdot |g|)$ and the simulation in $\mathcal{O}(|X|^2)$. Incorporating intersection with diagonal constraints requires an additional standardization step since diagonal constraints may break this property. For the sake of completeness, the successor computation of zones is explained in Appendix B. For the simulation, the algorithm from [25] is used. However, in the presence of diagonal constraints, the simulation check becomes NP-complete in general, and makes use of heuristics that allows for a faster check in practice. What remains is to show that $\preceq_{\mathcal{A}}$ is a finite simulation for X_D-safe GTA.

7 Finiteness of the simulation relation

In this section, we show that the simulation relation $\preceq_{\mathcal{A}}$ proposed in Section 5 is finite for safe GTA, which proves termination of the symbolic enumeration-based reachability algorithm. We do this in two parts: first, we show that the zones that are reached during the enumeration satisfy some invariants, in particular, only finitely many values occur in constraints among future clocks. This is however not necessarily true for history clocks. There the simulation comes into play. In the second part of the proof, we combine the invariants with an equivalence relation to show finiteness of the simulation. Below, we sketch these arguments and provide intuition leaving formal details to appendix due to lack of space.

Throughout this section, we fix an X_D-safe GTA $\mathcal{A}$. Let $M = \max\{|c| \mid c \in \mathbb{Z}$ is used in some constraint of $\mathcal{A}\}$, called the maximal constant of $\mathcal{A}$. We say that a zone Z is reachable if there is some reachable node (q, Z) in $\mathsf{GZG}(\mathcal{A})$.

Part 1: Invariants on zones. We start by showing an important property of reachable zones: closure under valuations that agree on the value of history clocks, and satisfy the same set of safe constraints involving non-history clocks.

We say that a constraint $x - y \triangleleft c$ is M-*bounded* if either $c \in \mathbb{R}$ is such that $|c| \leq M$ or $c \in \{-\infty; +\infty\}$. It is X_D-safe if $x, y \in X_F$ implies $x, y \in X_D$. We say that it is (X_D, M)-safe if it is both M-bounded and X_D-safe.

Lemma 8. *Let $v, v' \in \mathbb{V}$ be such that $v'{\downarrow_{X_H}} = v{\downarrow_{X_H}}$ and, for all (X_D, M)-safe constraints $y - x \triangleleft c$ with $x, y \in X_F \cup \{0\}$, we have $v' \models y - x \triangleleft c$ if and only if $v \models y - x \triangleleft c$. Let Z be a* reachable *zone. Then, $v \in Z$ if and only if $v' \in Z$.*

The proof (given in Appendix C) works by establishing that the property is true in the initial zone, and showing that it is invariant under the zone operations used to compute $\mathsf{GZG}(\mathcal{A})$. This 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. As a non-example, consider Figure 3. 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.

From Lemma 8, we get the following corollary (with a more precise statement and proof in Appendix C, Lemma 18). Namely, if a reachable zone Z contains a valuation v in which the difference between two future clocks x, y (including the zero clock) is finite and large enough, then Z contains valuations where the difference between x and y is any finite and large enough value.

Corollary 9. *Let Z be a reachable zone and let $v \in Z$. Let $n = \max(1, |X_D|)$. For all $x, y \in X_F \cup \{0\}$, if $-\infty < v(x) - v(y) < -nM$ then, for every α with $-\infty < \alpha < -nM$, we have a valuation $v' \in Z$ with $v'(x) - v'(y) = \alpha$.*

Notice that the property above does not hold if we simply take $n = 1$. For instance, if we have two clocks $x, z \in X_D$ then, applying the (X_D, M)-safe program $\langle [x, z]; z = -M \wedge x - z = -M\rangle$ from $\mathbb{V}$ results in a zone Z where all valuations v satisfy $v(x) = -2M$. So the property fails with $n = 1$, x and $y = 0$. This is a noteworthy difference between models with and without diagonals.

Using Corollary 9, we can 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.

Lemma 10. *Let Z be a nonempty reachable zone. Let $n = \max(1, |X_D|)$. Then, the normalized distance graph $\mathbb{G}(Z)$ satisfies the following ($\dagger$) conditions:*
- $\dagger_1$ *For all $x \in X_F$, $y \in X_H \cup \{0\}$, if Z_{xy} is finite, then $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$.*
- $\dagger_2$ *For all $x \in X_F$, if Z_{0x} is finite, then $(<, -nM) \leq Z_{0x} \leq (\leq, 0)$.*
- $\dagger_3$ *For all $x \in X_H$, $y \in X_F$, if Z_{0y} is finite, then $Z_{x0} + (<, -nM) \leq Z_{xy}$.*
- $\dagger_4$ *For $x, y \in X_F$, if Z_{xy} is finite, then $(<, -nM) \leq Z_{xy} \leq (\leq, nM)$.*

Proof. We focus on $\dagger_1, \dagger_2$, leaving the more complicated cases to Appendix C.

$\dagger_1$ 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 20 (Appendix C), we can find $\alpha \in \overline{\mathbb{R}}$ 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 21 (Appendix C), 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 9, we can get a valuation $v' \in Z$, such that $-\infty < v'(x) < -c$, a contradiction as it violates the constraint $0 - x \triangleleft c$ of Z. 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 $(\leq, 0) \leq Z_{x0} \leq (\leq, nM)$.

$\dagger_2$ We have to show 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 $-\infty < c < -nM$. As before, we can find α such that $(\leq, \alpha) \leq Z_{0x}$, $(\leq, -\alpha) \leq Z_{x0}$ and $\alpha \neq -\infty$. Then, by Lemma 21 (Appendix C), we can find $v \in Z$ with $v(x) = \alpha$. We have $-\infty < v(x) \triangleleft c < -nM$. Now, using Corollary 9, 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. $\square$

Part 2. Equivalence and Finiteness. We introduce below an equivalence relation $\sim_M^n$ of *finite index* on valuations, depending on $n = \max(1, |X_D|)$ and the maximal constant M, and show that, if G is a set of atomic M-bounded integral constraints and if Z is a zone such that its canonical distance graph $\mathbb{G}(Z)$ satisfies ($\dagger$) conditions, then the downward closure $\downarrow_G Z = \{v \in \mathbb{V} \mid \exists v' \in Z \text{ with } v \preceq_G v'\}$ is a union of $\sim_M^n$ equivalence classes.

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.

Lemma 11. *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 10. 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 $v_2 \in \downarrow_G Z$.*

Finally, from Lemmas 10 and 11, we obtain our main theorem of the section.

Theorem 12. *The simulation relation $\preceq_{\mathcal{A}}$ is finite if $\mathcal{A}$ is safe.*

Proof. Let $(q, Z_0), (q, Z_1), (q, Z_2), \ldots$ be an infinite sequence of *reachable* nodes in the zone graph of $\mathcal{A}$. By Lemma 10, 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 11 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 $Z_j \preceq_{\mathcal{G}(q)} Z_i$. $\square$

8 Experimental evaluation

We have implemented a prototype that takes as input a GTA, as given in Definition 1, and applies our reachability algorithm, in the open source tool Tchecker [28]. To do so, we extend Tchecker to allow clocks to be declared

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 D, 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.

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). We present selected results in Table 1, with further details in Appendix D.

First, we consider timed automata models from standard benchmarks [20, 34, 39]. Despite the overhead induced by our framework (e.g., maintaining general programs on transitions), we are only slightly worse off wrt. 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 comparison of our tool with the implementation of the state-of-the-art zone-based reachability algorithm using $\mathcal{G}$-simulation introduced in [23–25].

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 [23–25] 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 φ 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 CSMACD [39], Fire-alarm [35] and Alternating-bit-protocol(ABP) [33]. 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 [18] 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 D.

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.

9 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, [21], 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 [27]. As the next steps, we would like to investigate liveness verification for GTA, in particular what future clocks bring us when we consider the setting of ω-words.

References

1. Akshay, S., Bollig, B., Gastin, P.: Event clock message passing automata: a logical characterization and an emptiness checking algorithm. Formal Methods Syst. Des. **42**(3), 262–300 (2013)
2. Akshay, S., Gastin, P., Govind, R., Srivathsan, B.: Simulations for event-clock automata. In: CONCUR. LIPIcs, vol. 243, pp. 13:1–13:18 (2022)
3. Akshay, S., Gastin, P., Govind, R., Srivathsan, B.: Simulations for event-clock automata. CoRR **abs/2207.02633** (2022)
4. Akshay, S., Gastin, P., Prakash, K.R.: Fast zone-based algorithms for reachability in pushdown timed automata. In: CAV (1). LNCS, vol. 12759, pp. 619–642 (2021)
5. Alur, R.: Techniques for automatic verification of real-time systems. Ph.D. thesis, Stanford University (1991)
6. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid Systems. pp. 209–229 (1992)
7. Alur, R., Dill, D.L.: Automata for modeling real-time systems. In: ICALP. LNCS, vol. 443, pp. 322–335. Springer (1990)
8. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**, 183–235 (1994)
9. Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: A determinizable class of timed automata. Theor. Comput. Sci. **211**(1-2), 253–273 (1999)
10. de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G.: Real-Time: Theory in Practice: REX Workshop, Mook, The Netherlands. Proceedings, vol. 600 (1992)
11. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer (2003)
12. Bernstein, A.J., Jr., P.K.H.: Proving real-time properties of programs with temporal logic. In: SOSP. pp. 1–11. ACM (1981)
13. Bouyer, P.: Forward analysis of updatable timed automata. Formal Methods Syst. Des. **24**(3), 281–320 (2004)
14. Bouyer, P., Chevalier, F.: On conciseness of extensions of timed automata. J. Autom. Lang. Comb. **10**(4), 393–405 (2005)
15. Bouyer, P., Colange, M., Markey, N.: Symbolic optimal reachability in weighted timed automata. In: CAV (1). LNCS, vol. 9779, pp. 513–530. Springer (2016)
16. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. Theor. Comput. Sci. **321**(2-3), 291–345 (2004)
17. Bouyer, P., Gastin, P., Herbreteau, F., Sankur, O., Srivathsan, B.: Zone-based verification of timed automata: Extrapolations, simulations and what next? In: FORMATS. LNCS, vol. 13465, pp. 16–42. Springer (2022)
18. Bozzelli, L., Montanari, A., Peron, A.: Taming the complexity of timeline-based planning over dense temporal domains. In: FSTTCS. LIPIcs, vol. 150, pp. 34:1–34:14 (2019)
19. Bozzelli, L., Montanari, A., Peron, A.: Complexity issues for timeline-based planning over dense time under future and minimal semantics. Theor. Comput. Sci. **901**, 87–113 (2022)
20. Daws, C., Olivero, A., Tripakis, S., Yovine, S.: The tool KRONOS. In: Hybrid Systems. LNCS, vol. 1066, pp. 208–219. Springer (1995)
21. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Automatic Verification Methods for Finite State Systems. LNCS, vol. 407, pp. 197–212. Springer (1989)

22. D'Souza, D., Tabareau, N.: On timed automata with input-determined guards. In: FORMATS/FTRTFT. LNCS, vol. 3253, pp. 68–83. Springer (2004)
23. Gastin, P., Mukherjee, S., Srivathsan, B.: Reachability in timed automata with diagonal constraints. In: CONCUR. LIPIcs, vol. 118, pp. 28:1–28:17 (2018)
24. Gastin, P., Mukherjee, S., Srivathsan, B.: Fast algorithms for handling diagonal constraints in timed automata. In: CAV (1). LNCS, vol. 11561, pp. 41–59 (2019)
25. Gastin, P., Mukherjee, S., Srivathsan, B.: Reachability for updatable timed automata made faster and more effective. In: FSTTCS. LIPIcs, vol. 182, pp. 47:1–47:17 (2020)
26. Geeraerts, G., Raskin, J., Sznajder, N.: Event clock automata: From theory to practice. In: FORMATS. LNCS, vol. 6919, pp. 209–224. Springer (2011)
27. Geeraerts, G., Raskin, J., Sznajder, N.: On regions and zones for event-clock automata. Formal Methods Syst. Des. **45**(3), 330–380 (2014)
28. Herbreteau, F., Point, G.: TChecker. `https://github.com/fredher/tchecker` (v02 - April 2019)
29. Herbreteau, F., Srivathsan, B., Tran, T., Walukiewicz, I.: 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)
30. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC '99) (1999)
31. Jonsson, B., Vaandrager, F.: Learning mealy machines with timers. Tech. rep. (2018), `https://sws.cs.ru.nl/publications/papers/fvaan/MMT/`
32. Koymans, R., Vytopil, J., de Roever, W.P.: Real-time programming and asynchronous message passing. In: PODC. pp. 187–197. ACM (1983)
33. Kurose, J.F., Ross, K.W.: Computer networking - a top-down approach featuring the internet. Addison-Wesley-Longman (2001)
34. Lugiez, D., Niebert, P., Zennou, S.: A partial order semantics approach to the clock explosion problem of timed automata. Theor. Comput. Sci. **345**(1), 27–59 (2005)
35. Muñiz, M., Westphal, B., Podelski, A.: Timed automata with disjoint activity. In: FORMATS. LNCS, vol. 7595, pp. 188–203. Springer (2012)
36. Raskin, J., Schobbens, P.: The logic of event clocks - decidability, complexity and expressiveness. J. Autom. Lang. Comb. **4**(3), 247–282 (1999)
37. Sorea, M.: Tempo: A model checker for event-recording automata. Tech. rep., In Proceedings of RT-Tools'01 (2001)
38. Srivathsan, B.: Reachability in timed automata. ACM SIGLOG News **9**(3), 6–28 (2022)
39. Tripakis, S., Yovine, S.: Analysis of timed systems using time-abstracting bisimulations. Formal Methods Syst. Des. **18**(1), 25–68 (2001)
40. Zhao, J., Li, X., Zheng, G.: A quadratic-time dbm-based successor algorithm for checking timed automata. Inf. Process. Lett. **96**(3), 101–105 (2005)
41. Anonymised link for extended experiments for the paper. `https://drive.google.com/drive/folders/1kWmyk-Ex7E2lZOcHklbOLgxselq3NJOF?usp=sharing`

A Appendix for Section 5

Our goal is to show that $\preceq_{\mathcal{A}}$ defined in Section 5 is a simulation relation. For this, we first need to prove the following technical lemma.

Lemma 13. *Let R be a set of clocks. Let G be a set of atomic constraints and $G' = \mathsf{pre}([R], G)$. 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 suffices to prove the lemma when $G = \{\varphi\}$ where φ is an atomic constraint $y - x \triangleleft c$. Let $G' = \mathsf{pre}([R], G)$, 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 $\varphi' = \varphi$, $v'_1(y) = v_1(y)$, $v'_2(y) = v_2(y)$, $v'_1(x) = v_1(x)$ and $v'_2(x) = v_2(x)$. Since $v_1 \preceq_\varphi v_2$, we get $v'_1 \preceq_\varphi v'_2$.
- 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 obtain $v_2 + \delta' \models \varphi'$, i.e., $v'_2(y) - v'_2(x) \triangleleft c$.
- 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. □

Proof of Theorem 5. 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 $\mathbb{TS}_{\mathcal{A}}$. Then, $v_1 + \delta \models X_F \leq 0$. Since $\mathcal{G}(q)$ contains $x \leq 0$ for all $x \in X_F$ and $v_1 \preceq_{\mathcal{G}(q)} v_2$, 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 $\mathbb{TS}_{\mathcal{A}}$. It is easy to see that $v_1 + \delta \preceq_{\mathcal{G}(q)} v'_1 + \delta$.

Discrete transition Let $(q_1, v_1) \xrightarrow{a,\text{prog}} (q, v) \in \mathbb{TS}_{\mathcal{A}}$ 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 sequence 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 $G_i = \mathsf{pre}(\langle g_i; [R_i]\rangle, G_{i+1})$ 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 $\mathsf{split}(g_i) \subseteq G_i$ 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 $\mathsf{pre}([R_i], G_{i+1}) \subseteq G_i$. 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 conclude that $v_{i+1} \preceq_{G_{i+1}} v'_{i+1}$, which completes the proof. □

B Appendix for Section 6

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 $x \to y$ edges.

Definition 14 (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 $y - x \triangleleft c$ in g, replace weight of edge $x \to y$ 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 $\overrightarrow{\mathbb{G}}$ is obtained by doing the following:*
– *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 [40].

Theorem 15. *Let $\mathbb{G}$ be a normalized distance graph, g be a guard and $R \subseteq X$ be a set of clocks. We have $[\![\mathbb{G}]\!] \cap g = [\![\mathbb{G}_g]\!]$, $[R][\![\mathbb{G}]\!] = [\![[R]\mathbb{G}]\!]$, and $\overrightarrow{[\![\mathbb{G}]\!]} = [\![\overrightarrow{\mathbb{G}}]\!]$.*

Moreover, $\mathbb{G}_g$, $[R]\mathbb{G}$ and $\overrightarrow{\mathbb{G}}$ can be computed in time $\mathcal{O}(|X|^2)$.

The proof can be found in [3], except for the fact that guard intersection with diagonal constraints requires standardization, which can be done in time $\mathcal{O}(|X|^2)$, and the following normalization is a bit more complicated. We could use the general algorithm which checks for negative cycle and normalizes in time $\mathcal{O}(|X|^3)$. Alternatively, we can start by handling the non-diagonal edges of g as in [3]. Then, for each diagonal constraint $y - x \triangleleft c$ in g, we reduce the weight of all edges $x' \to y'$ if the path $x' \to x \xrightarrow{\triangleleft,c} y \to y'$ is shorter.

C Appendix for Section 7

Valuations of safely reachable zones. To prove Lemma 8, we define an equivalence that relates valuations that agree on value of history clocks, and satisfy the same set of M-bounded, X_D-safe constraints *involving non-history clocks*. We then show that all safely reachable zones are closed under this equivalence.

Definition 16. *Let $v_1, v_2 \in \mathbb{V}$ be valuations. We define $v_1 \simeq v_2$ if $v_1{\downarrow_{X_H}} = v_2{\downarrow_{X_H}}$ and, for all (X_D, M)-safe constraints $y - x \triangleleft c$ with $x, y \in X_F \cup \{0\}$, 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 $-\infty < v_1(x) < -M$ iff $-\infty < v_2(x) < -M$.

Definition 16 requires that v_1 and v_2 satisfy the same set of (X_D, M)-safe constraints involving *non-history* clocks. We now show that if $v_1 \simeq v_2$, then v_1 and v_2 satisfy the same (X_D, M)-safe constraints involving *any pair* of clocks.

Lemma 17. *If $v_1 \simeq v_2$ then, for all (X_D, M)-safe constraints $y - x \triangleleft c$ (with $x, y \in X \cup \{0\}$), 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 16 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, even if not M-bounded) not involving future clocks. Finally, we consider M-bounded constraints involving a history clock y and a future clock x.
(1) 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.
(2) 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 $-\infty < v_2(x) < -M$. We consider the two possibilities.

- $y - x \triangleleft c$. 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 $v_1 \models y - x \triangleleft c$ iff $v_2 \models y - x \triangleleft c$.
- $x - y \triangleleft c$. We argue similarly, distinguishing two cases depending on whether $v_1(y) = v_2(y)$ is finite or not. ☐

We are now ready to prove Lemma 8 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 zone operations.

Proof of Lemma 8. We have to prove that, if Z is a reachable zone and if $v, v' \in \mathbb{V}$ are valuations s.t. $v \simeq v'$, then $v \in Z$ implies $v' \in Z$. The property is true if $Z = \mathbb{V}$ is the set of all valuations. Notice that the initial zone is $Z_0 = \overrightarrow{\mathbb{V} \cap g_0}$ and g_0 is (X_D, M)-safe.

We will prove that this property is invariant under application of the zone operations used when computing the zone graph $\mathsf{GZG}(\mathcal{A})$ of an X_D-safe GTA $\mathcal{A}$. Assume that Z is a zone that satisfies the property.

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 17 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 $v \in \overrightarrow{Z}$, 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 $v' = u' + \delta \in \overrightarrow{Z}$.
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 constraints, we get $-v'(z) \triangleleft c - \delta$. Using $v' = u' + \delta$, we get $-u'(z) \triangleleft c$, i.e., $u' \models 0 - z \triangleleft c$. □

The proof above is for reachable zones Z in an X_D-safe GTA $\mathcal{A}$. 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 $-\infty$).

Using Lemma 8, we obtain Lemma 18, which subsumes Corollary 9.

Lemma 18. *Let Z be a reachable zone and let $v \in Z$. Let $n = \max(1, |X_D|)$.*

1. *Let $x \in X_F \setminus X_D$. If $-\infty < v(x) < -M$ then, for every $-\infty < \alpha < -M$, 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 α with $-\infty < \alpha < -nM$, 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 α with $-\infty < \alpha < -nM$, 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 8 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, $v'(x) - v'(y) = \alpha$. $\square$

Remark 19. Note that in the second and third parts of Lemma 18, 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 all other clocks unchanged.

Proof of Lemma 10. We first give two technical lemmas from [3].

Lemma 20 ([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 α with $\alpha \not\triangleleft'' c''$.*

Lemma 21 ([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 $v \in [\![\mathbb{G}]\!]$ 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 22. *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 $x \to 0$ or the weight $(\leq, 0)$ for edge $0 \to x$. If the weight of $x \to 0$ is $(\leq, -\infty)$, it stays unchanged until we first apply the reset operation on x, resulting in the weight $(\leq, 0)$ for edge $0 \to x$. Then, the weight of edge $0 \to x$ 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. □

Proof of $\dagger_3$ *and* $\dagger_4$ *of Lemma 10.*

$\dagger_3$ 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 22, this implies $Z_{xy} \neq (\leq, -\infty)$. Let $Z_{x0} = (\triangleleft, -c)$ and $Z_{xy} = (\triangleleft', e)$, as shown in Figure 4. We have $0 \leq c < \infty$ and $-\infty < e \leq 0$.

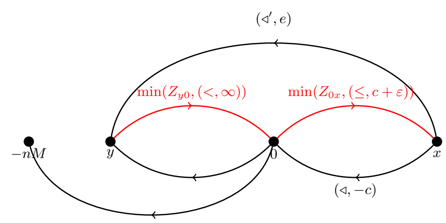


Fig. 4: Distance graph $\mathbb{G}(Z)$ (without the red edges) and $\mathbb{G}'$ (with the red edges).

Fix $\varepsilon > 0$. Consider the distance graph $\mathbb{G}'$ obtained from $\mathbb{G}(Z)$ by setting the weight of $0 \to x$ to $\min(Z_{0x}, (\leq, c + \varepsilon))$, and the weight of $y \to 0$ to $\min(Z_{y0}, (<, \infty))$, as shown in Figure 4. 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 $0 \to x$ or the new weight $(<, \infty)$ of $y \to 0$ or both. Then the possible negative cycles are:

- $0 \to x \to 0$ with weight $(\leq, c + \varepsilon) + Z_{x0} = (\leq, c + \varepsilon) + (\triangleleft, -c) = (\triangleleft, \varepsilon)$, which is non-negative, since $\varepsilon > 0$.
- $0 \to y \to 0$ whose weight $Z_{0y} + (<, \infty)$ is non-negative since $Z_{0y} \neq (\leq, -\infty)$,
- $y \to 0 \to x \to y$ with weight $(<, \infty) + (\leq, c + \varepsilon) + Z_{xy}$ which is non-negative since $Z_{xy} \neq (\leq, -\infty)$.

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

We will now show that there exists a valuation $v' \in [\![\mathbb{G}']\!]$ such that $-nM - \varepsilon \leq v'(y)$. Let $v \in [\![\mathbb{G}']\!]$. 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 Lemma 18, we get a valuation $v' \in [\![\mathbb{G}']\!]$ such that $v'(y) = -nM - \varepsilon$ since $\varepsilon > 0$. Since $v' \in [\![\mathbb{G}']\!]$, 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 $v' \in [\![\mathbb{G}']\!]$ and $\mathbb{G}'_{xy} = (\triangleleft', e)$. Since this holds for all $\varepsilon > 0$ we deduce that $-nM - c \leq e$. We obtain $(<, -nM - c) \leq (\triangleleft', e) = Z_{xy}$. We conclude using $(<, -nM - c) = (<, -nM) + (\triangleleft, -c)$.

$\dagger_4$ For $x, y \in X_F$, if Z_{xy} is finite, then $(<, -nM) \le Z_{xy} \le (\le, nM)$.

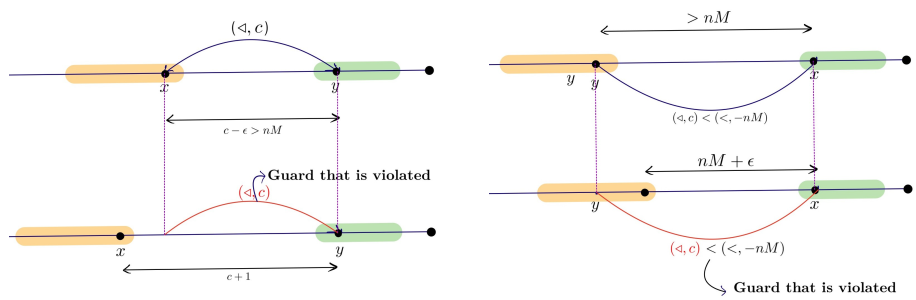
Suppose that $x, y \in X_F$, and $Z_{xy} = (\triangleleft, c)$ is finite, i.e., $c \in \mathbb{R}$. Notice that, since $\mathbb{G}(Z)$ is standard, this implies $Z_{x0} \neq (\le, \infty)$. The proof proceeds by application of Lemma 21, and for this, when $x, y \in X$ and $\alpha \in \mathbb{R}$ is finite, we need to first show that the weights $Z_{x0}, Z_{0x}, Z_{y0}, Z_{0y}$ are all different from $(\le, -\infty)$.

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

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

Next, we consider the two possibilities for violation of the $\dagger_4$ condition. We will show that both of them lead to a contradiction. (1) $(\le, nM) < Z_{xy} = (\triangleleft, c) < (<, \infty)$. This implies that $nM < c < \infty$.

Using Lemma 20, we can find $\alpha \in \mathbb{R}$ such that $(\le, \alpha) \le Z_{xy}$, $(\le, -\alpha) \le Z_{yx}$, and $nM < \alpha$. Notice that $\alpha \le c < \infty$. Further, using Lemma 21, 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 Lemma 18, 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 5a. This contradicts the constraint $y - x \triangleleft c$ in Z.



(a) Increasing the difference between x and y using $\simeq$-equivalence.

(b) Shrinking the difference between x and y using $\simeq$-equivalence.

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

Using Lemma 20, we can find $\alpha \in \mathbb{R}$ such that $(\le, \alpha) \le Z_{xy}$, $(\le, -\alpha) \le Z_{yx}$, and $-\infty < \alpha$. Notice that $\alpha \le c < -nM$. Further, using Lemma 21, we know that there exists a valuation $v \in Z$ with $v(y) - v(x) = \alpha$. Since $-\infty < \alpha < -nM$, we use Lemma 18 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 5b. This contradicts the constraint $y - x \triangleleft c$ in Z. Therefore, if Z_{xy} is finite, then $(<, -nM) \le Z_{xy} \le (\le, nM)$. $\square$

Proof of Lemma 11. Our goal is to show that $\downarrow_G Z$ is a union of $\sim^n_M$ equivalence classes. But we do not know how to represent $\downarrow_G Z$ using distance graphs. Instead we will reason via up-sets: 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. Notice that $v \in \downarrow_G Z$ iff $\uparrow_G v \cap Z \neq \emptyset$.

Now, we define a distance graph $\mathbb{G}_v^G$ and a guard g_v^G such that $\uparrow_G v = [\![\mathbb{G}_v^G]\!] \cap [\![g_v^G]\!]$. The guard g_v^G is the conjunction of all constraints in G satisfied by v. The distance graph $\mathbb{G}_v^G$ is defined as follows:

(1) For a future clock $x \in X_F$, we have the edges $x \xrightarrow{(\leq,-v(x))} 0$ and $0 \xrightarrow{(\leq,v(x))} x$;

(2) For each history clock $y \in X_H$, we have (a) $0 \to y$ with weight $(\leq, v(y))$ if there is a constraint $y \triangleleft c$ in G with $c < \infty$ and $v \models y \triangleleft c$, (b) $y \to 0$ with weight $(\leq, -v(y))$ if there is a constraint $c \triangleleft y$ in G with $c < \infty$ and $v \not\models c \triangleleft y$.

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 $\uparrow_G v = [\![\mathbb{G}_v^G]\!] \cap [\![g_v^G]\!]$.

Lemma 23. *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 $\uparrow_G v = [\![\mathbb{G}_v^G]\!] \cap g_v^G$.*

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., $v' \in [\![\mathbb{G}_v^G]\!]$.

$\supseteq$: Let $v \in [\![\mathbb{G}_v^G]\!]$ 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 $v \in [\![\mathbb{G}_v^G]\!]$ 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, $v \preceq_{g'} v'$. $\square$

Remark 24. (1) $\mathbb{G}_v^G$ is in standard form, but is not necessarily normalized.
(2) $[\![\mathbb{G}_v^G]\!]$ is non-empty, since $v \in [\![\mathbb{G}_v^G]\!]$.
(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 25. *Let v be a valuation, Z' a non-empty* reachable *GTA 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. *$0 \to x \to 0$ with $0 \to x$ from $\mathbb{G}_v^G$ and $x \to 0$ from $\mathbb{G}'$,*
2. *$0 \to y \to 0$ with $0 \to y$ from $\mathbb{G}'$ and $y \to 0$ from $\mathbb{G}_v^G$, and*
3. *$0 \to x \to y \to 0$, with weight of $x \to y$ from $\mathbb{G}'$ and the others from $\mathbb{G}_v^G$.*

Proof. Since $\mathbb{G}'$ is normalized, 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 normalized, 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 $0 \to x \to 0$, one from $\mathbb{G}'$ and one from $\mathbb{G}_v^G$. Otherwise, the simple cycle is of the form $0 \to x \to y \to 0$ where the edge $x \to y$ is from $\mathbb{G}'$ and the other two edges are from $\mathbb{G}_v^G$. $\square$

Lemma 26. *Let $v \sim_M^n v'$ and G be a set of M-bounded integral constraints. Then, we have the following (1) $g_{v'}^G = g_v^G$, and (2) $\mathbb{G}_{v'}^G$ 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 $g_{v'}^G$ and g_v^G, and the fact that $v \sim_{(n+1)M} v'$.
(2) For a future clock $x \in X_F$, this follows from the definition for edges $x \to 0$ and $0 \to x$ adjacent to x. We consider edges adjacent to history clocks $y \in X_H$.

- Consider the edge $0 \to y$. 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 $0 \to y$ has weight $(\leq, v'(y))$ in $\mathbb{G}_{v'}^G$.
- Consider the edge $y \to 0$. 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 $y \to 0$ has weight $(\leq, -v'(y))$ in $\mathbb{G}_{v'}^G$. $\square$

Proof of Lemma 11. 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 23, this amounts to $Z \cap g_{v_1}^G \cap [\![\mathbb{G}_{v_1}^G]\!] \neq \emptyset$ iff $Z \cap g_{v_2}^G \cap [\![\mathbb{G}_{v_2}^G]\!] \neq \emptyset$.

Further, since $v_1 \sim_M^n v_2$, using Lemma 26, 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 † conditions of Lemma 10 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 $\mathbb{G}^G_{v_1}$ and $\mathbb{G}^G_{v_2}$. 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 $Z' \cap [\![\mathbb{G}^G_{v_1}]\!] \neq \emptyset$ and $Z' \cap [\![\mathbb{G}^G_{v_2}]\!] = \emptyset$. By Lemma 25, we can find a negative cycle C_2 using one edge from $\mathbb{G}(Z')$ and one or two edges from $\mathbb{G}^G_{v_2}$. By Lemma 26, we have a corresponding cycle C_1 using the same edge from $\mathbb{G}(Z')$ and the same one or two edges from $\mathbb{G}^G_{v_1}$ (with weights using v_1 instead of v_2). The cycle C_1 is not negative since $Z' \cap [\![\mathbb{G}^G_{v_1}]\!] \neq \emptyset$

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 $\mathbb{G}^G_{v_1}$, there is a constraint $y \lhd' c'$ in G with $c' < \infty$ and $v_1(y) \lhd' c'$. We deduce that $0 \leq v_1(y) \leq M$. Let $Z'_{y0} = (\lhd, c)$. Since C_1 is not a negative cycle, we get $(\leq, 0) \leq (\lhd, c + v_1(y))$, which is equivalent to $-c \lhd v_1(y)$. Using $0 \leq v_1(y) \leq M$ and $v_1 \sim^n_M v_2$ we deduce that $-c \lhd v_2(y)$. This is equivalent to $(\leq, 0) \leq (\lhd, c + v_2(y))$, 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 $\mathbb{G}^G_{v_1}$, there is a constraint $c' \lhd' y$ in G with $c' < \infty$ and $c' \not\lhd' v_1(y)$. We deduce that $0 \leq v_1(y) \leq M$. Let $Z'_{0y} = (\lhd, c)$. Since C_1 is not a negative cycle, we get $(\leq, 0) \leq (\lhd, c - v_1(y))$, which is equivalent to $v_1(y) \lhd c$. Using $v_1 \sim^n_M v_2$ and $0 \leq v_1(y) \leq M$, we deduce that $v_2(y) \lhd c$. This is equivalent to $(\leq, 0) \leq (\lhd, c - v_2(y))$, 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 $v_1 \sim^n_M v_2$, a contradiction with C_1 being non-negative. Hence, $Z'_{x0} = (\lhd, c)$ is finite and by ($\dagger_1$), we infer $0 \leq c \leq nM$. Since C_1 is not negative, we get $(\leq, 0) \leq (\lhd, c + v_1(x))$, which is equivalent to $-c \lhd v_1(x)$. Using $v_1 \sim^n_M v_2$ and $0 \leq c \leq nM$ we deduce that $-c \lhd v_2(x)$. This is equivalent to $(\leq, 0) \leq (\lhd, c + v_2(x))$, 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 $Z'_{0x} = (\lhd, c)$. Since C_2 is negative, we deduce that $v_2(x) \neq -\infty$. Using $v_1 \sim^n_M v_2$, 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 $(\leq, 0) \leq (\lhd, c - v_1(x))$, which is equivalent to $v_1(x) \lhd c$. Using $v_1 \sim^n_M v_2$ and $-nM \leq c \leq 0$, we get $v_2(x) \lhd c$. This is equivalent to $(\leq, 0) \leq (\lhd, c - v_2(x))$, 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 $v_1 \sim^n_M v_2$, 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 $v_1 \sim^n_M v_2$ 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 $v_1 \sim^n_M v_2$ and $0 \leq v_1(y) - v_1(x) \leq (n+1)M$, we obtain $v_2(y) - v_2(x) \triangleleft c$. 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 $v_1 \sim^n_M v_2$ 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 $(\leq, -\infty) < Z'_{xy} < (<, \infty)$. 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 $v_1 \sim^n_M v_2$ 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 $C_2 = 0 \xrightarrow{(\leq, v_2(x))} x \xrightarrow{Z'_{xy}} y \xrightarrow{(\leq, -v_2(y))} 0$, $x \neq y$ 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 $v_1 \sim^n_M v_2$ and $-M \leq v_1(y) - v_1(x) \leq M$, we obtain $v_2(y) - v_2(x) \triangleleft c$. We conclude as in the previous cases.

Notice that we have crucially used the "$(n+1)M$" occurring in the definition of $v_1 \sim^n_M v_2$ (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). In the rest of the cases, it was sufficient to use $v_1(x) \sim_{nM} v_2(x)$. □

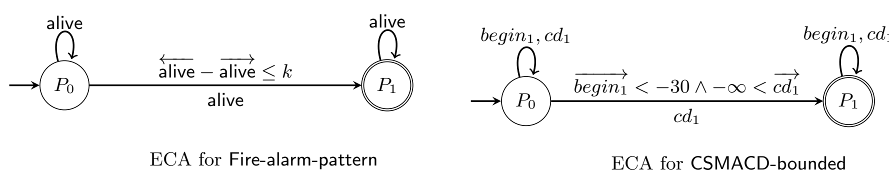
D Appendix for Section 8

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 7). The models for Fire-alarm and CSMACD are the standard models as given in [35] and [29], respectively. A detailed discussion of models and the product of the model with the ECA is available at [41].

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.



(a) ECA model for Fire-alarm-pattern (b) ECA model for CSMACD-bounded

Fire-alarm-pattern. We consider the Fire-alarm model from [35]. 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 Sensor_1 in k time units. The negation of this property can be modeled by the ECA in Figure 6a with two states and a transition on alive with the diagonal constraint $\overleftarrow{\mathsf{alive}} - \overrightarrow{\mathsf{alive}} \leq k$. When reading an action alive, the quantity $\overleftarrow{\mathsf{alive}} - \overrightarrow{\mathsf{alive}}$ gives the distance between the next and the previous occurrence.

CSMACD-bounded. We consider the CSMACD model given in [29]. 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, Station_1 sends a message (modelled using a $begin_1$ action) in 30 time units. The negation of this property can be modeled by the ECA of Figure 6b. When reading an action cd_1, the constraints (1) $\overrightarrow{begin_1} < -30$ says that $begin_1$ (which denotes 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).

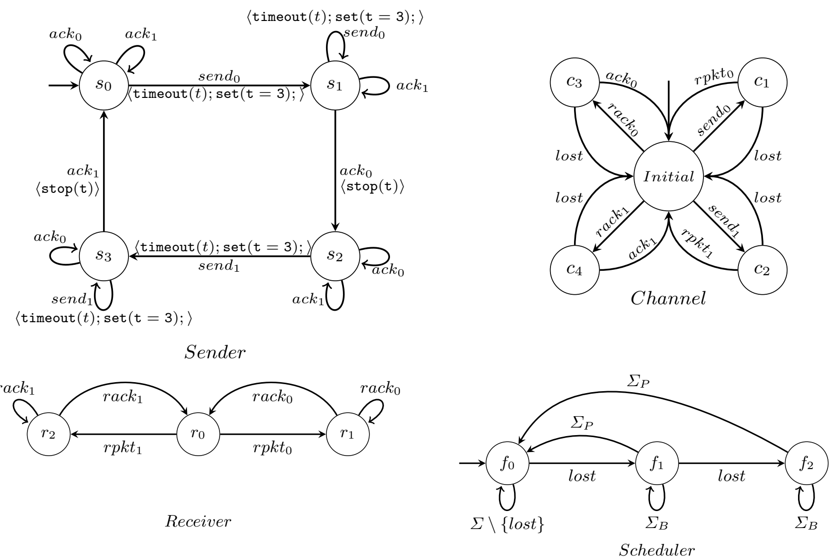


Fig. 7: Alternating-bit-protocol

Alternating-bit-protocol**.** We consider a variant of the Alternating-bit-protocol [33] as depicted in Figure 7. We model sending a packet with identifier $i \in \{0,1\}$ in Sender with the action $send_i$, and receiving an acknowledgement with identifier $i \in \{0,1\}$ in Sender with the action ack_i. The Sender uses a timer t. Recall that the timer operations are (1) $\mathtt{set(t=3)}$ that sets timer t to value c, (2) $\mathtt{timeout}(t)$ that checks whether t is 0, (3) $\mathtt{stop}(t)$ 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 7, $\Sigma_B = \{send_0, send_1, rack_0, rack_1\}$, $\Sigma_P = \{rpkt_0, rpkt_1, ack_0, ack_1\}$, $\Sigma = \Sigma_B \sqcup \Sigma_P \sqcup \{lost\}$.

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 8.

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 9. Note that this property does not hold for the sender in ABP.

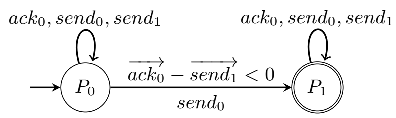


Fig. 8: ABP-prop1

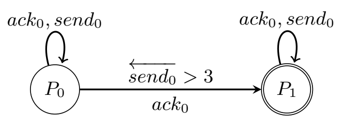


Fig. 9: ABP-prop2

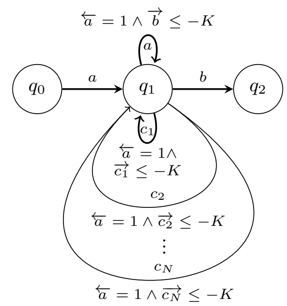


Fig. 10: ToyECA(K, N)

Synthetic Benchmarks in Table 1

ToyECA(K, N): As depicted in Figure 10, 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.

More experiments: A more detailed set of benchmarks and an extended discussion of experiments is available at [41].