Simulations for Event-Clock Automata

S. Akshay
Department of CSE, Indian Institute of Technology Bombay, Mumbai, India

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

R. Govind
Department of CSE, Indian Institute of Technology Bombay, Mumbai, India

B. Srivathsan
Chennai Mathematical Institute, India
CNRS, ReLaX, IRL 2000, Siruseri, India

Abstract

Event-clock automata are a well-known subclass of timed automata which enjoy admirable theoretical properties, e.g., determinizability, and are practically useful to capture timed specifications. However, unlike for timed automata, there exist no implementations for event-clock automata. A main reason for this is the difficulty in adapting zone-based algorithms, critical in the timed automata setting, to the event-clock automata setting. This difficulty was studied in [19, 20], where the authors also proposed a solution using zone extrapolations.

In this paper, we propose an alternative zone-based algorithm, *using simulations* for finiteness, to solve the reachability problem for event-clock automata. Our algorithm exploits the $\mathcal{G}$-simulation framework, which is the coarsest known simulation relation for reachability, and has been recently used for advances in other extensions of timed automata.

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

Keywords and phrases Event-clock automata, verification, zones, simulations, reachability

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2022.13

Related Version *Full Version*: `https://arxiv.org/abs/2207.02633`

Funding This work was supported by DST/CEFIPRA/INRIA Project EQuaVE.
S. Akshay: Supported in part by DST/SERB Matrics Grant MTR/2018/000744.
Paul Gastin: Partially supported by ANR project Ticktac (ANR-18-CE40-0015).

1 Introduction

Timed automata (TA) [4] are a well-established model for real-time systems and form the basis for employing model-checking techniques. The most popular property that has been considered in these systems is control state reachability. Reachability in timed automata is a well-studied problem and was shown to be decidable (and PSPACE-complete) using the so-called region construction [4]. This construction was primarily of theoretical interest, as the number of regions, which are collections of reachable configurations, explodes both in theory and in practice. On the other hand, timed automata have been implemented in several tools: UPPAAL [26, 6], KRONOS [10], PAT [29], RED [31], TChecker [21], Theta [30], LTS-Min [24], Symrob [28], MCTA [25], etc. Most of these tools have a common underlying algorithm which is an explicit enumeration of reachable configurations stored as *zones* [7]. Since the late 90s, a substantial effort has been invested in improving zone enumeration techniques, the common challenge being how to get a sound and complete enumeration while exploring as few zones as possible.

The more general model checking problem of whether the system represented by TA $\mathcal{A}$ satisfies the specification given by TA $\mathcal{B}$ reduces to the language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. There are two challenges here: first, the inclusion problem is undecidable in its full generality, and second, having clocks, though excellent for timed implementations, are often less than ideal for modeling timed specifications. This has led to the introduction of event-clocks and the corresponding model of *event-clock automaton (ECA)* [5]. Event-clock automata make use of special clocks that track the time since the last occurrence of an event (history clocks) or the time until the next occurrence of an event (prophecy clocks). On one hand this makes writing timed specifications more natural. Indeed, the role of prophecy clocks is in the same spirit as future modalities in temporal logics. This has led to several extensions of temporal logics with event-clocks [15, 1, 27], which are often used as specification languages and can be converted into ECA. On the other hand, ECA can be determinized and hence complemented. Observe that model-checking event-clock specifications over TA models can be reduced to the reachability problem on the product of the TA with an ECA. This product contains usual clocks, history clocks and prophecy clocks. The usual clocks can be treated in the same way as history clocks for the zone analysis. Therefore, if we solve ECA reachability (with history and prophecy clocks) using zones, we can incorporate usual clocks into the procedure seamlessly. The bottomline is that the well-motivated problem of model-checking event-clock specifications over TA models can be reduced to an ECA reachability problem.

Thus, in this paper, we focus on the core problem of building efficient, zone-based algorithms for reachability in ECA. This problem turns out to be significantly different compared to zone based reachability algorithms in usual TA, precisely due to prophecy clocks. Our goal is to align the zone-based reachability algorithms for ECA with recent approaches for TA that have shown significant gains.

As mentioned earlier, the core of an efficient TA reachability algorithm is an enumeration of zones, where the central challenge is that naïve enumeration does not terminate. One approach to guarantee termination is to make use of an *extrapolation* operation on zones: each new zone that is enumerated is extrapolated to a bigger zone. Any freshly enumerated zone that is contained in an existing zone is discarded. More recently, a new *simulation* approach to zone enumeration has been designed, where enumerated zones are left unchanged. Instead, with each fresh zone it is checked whether the fresh zone is simulated by an already seen zone. If yes, the fresh zone is discarded. Otherwise, it is kept for further exploration. Different simulations have been considered: the *LU*-simulation [22] which is based on *LU*-bounds, or the $\mathcal{G}$-simulation [18], which is based on a carefully-chosen set of constraints. Coarser simulations lead to fewer zones being enumerated. The $\mathcal{G}$-simulation is currently the coarsest-known simulation that can be efficiently applied in the simulation approach. The simulation based approach offers several gains over the extrapolation approach: (1) since concrete zones are maintained, one could use dynamic simulation parameters and dynamic simulations, starting from a coarse simulation and refining whenever necessary [23], (2) the simulation approach has been extended to richer models like timed automata with diagonal constraints [17, 16], updatable timed automata [18], weighted timed automata [9] and pushdown timed automata [3]. In these richer models, extrapolation has either been shown to be impossible [8] or is unknown.

Surprisingly, for ECA, an arguably more basic and well-known model, it turns out that there are no existing simulation-based approaches. However, an extrapolation approach using maximal constants has been studied for ECA in [19, 20]. In this work, the authors start by showing that prophecy clocks exhibit fundamental differences as compared to usual clocks. To begin with, it was shown that there is no finite time-abstract bisimulation for ECA in general. This is in stark contrast to TA where the region equivalence forms a finite time-abstract bisimulation. The correctness of extrapolation is strongly dependent on the region equivalence. Therefore, in order to get an algorithm, the authors define a weak semantics for ECA and a corresponding notion of weak regions which is a finite time-abstract bisimulation for the weak semantics and show that the weak semantics is sound for reachability. Building on this, they define an extrapolation operation for the zone enumeration.

Contributions. Given the advantages of using simulations with respect to extrapolations in the TA setting described above, we extend the $\mathcal{G}$-simulation approach to ECA. Here are the technical contributions leading to the result.

- We start with a slightly modified presentation of zones in ECA and provide a clean algebra for manipulating weights in the graph representation for such ECA-zones. This simplifies the reasoning and allows us to adapt many ideas for simulation developed in the TA setting directly to the ECA setting.
- The $\mathcal{G}$-simulation is parameterized by a set of constraints at each state of the automaton. We adapt the constraint computation and the definition of the simulation to the context of ECA, the main challenge being the handling of prophecy clocks.
- We give a simulation test between two zones that runs in time quadratic in the number of clocks. This is an extension of the similar test that exists for timed automata, but now it incorporates new conditions that arise due to prophecy clocks.
- Finally, we show that the reachability algorithm using the $\mathcal{G}$-simulation terminates for ECA: for every sequence $Z_0, Z_1, \ldots$ of zones that are *reachable* during a zone enumeration of an ECA, there exist $i < j$ such that Z_j is simulated by Z_i. This is a notable difference to the existing methods in TA, where finiteness is guaranteed for all zones, not only the reachable zones. In the ECA case, this is not true: we can construct an infinite sequence of zones which are incomparable with respect to the new $\mathcal{G}$-simulation. However, we show that finiteness does hold when restricting to reachable zones, and this is sufficient to prove termination of the zone enumeration algorithm. Our argument involves identifying some crucial invariants in reachable zones, specially, involving the prophecy clocks.

The fundamental differences in the behaviour of prophecy clocks as compared to usual clocks constitute the major challenge in developing efficient procedures for the analysis of ECAs. In our work, we have developed methods to incorporate prophecy clocks alongside the usual clocks. We prove a surprising property: in all *reachable* ECA-zones, the constraints involving *prophecy* clocks come from a finite set. A direct consequence of this observation is that the event zone graph of an ECA containing only prophecy clocks (known as Event-Predicting Automata EPA) is always finite. We wish to emphasize that, in this work, we are moving a step towards implementability, and at the same time towards more expressivity, since simulation approaches are amenable to extensions, e.g., with diagonal constraints.

Organization of the paper. Section 2 recalls ECA and describes a slightly modified presentation of the ECA semantics. Section 3 introduces event zones, event zone graph and the simulation based reachability framework. Section 4 introduces the new algebra for representing event zones and describes some operations needed to build the zone graph. Section 5 introduces the $\mathcal{G}$-simulation for event-clock automata and gives the simulation test. Section 6 proves finiteness of the simulation when restricted to reachable zones. All the missing proofs can be found in the full version of the paper [2].

2 Event Clock Automata and Valuations

Let X be a finite set of variables called *clocks*. Let $\Phi(X)$ denote a set of clock constraints generated by the following grammar: $\varphi ::= x \triangleleft c \mid c \triangleleft x \mid \varphi \wedge \varphi$ where $x \in X$, $c \in \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$ and $\triangleleft \in \{<, \leq\}$. The base constraints of the form $x \triangleleft c$ and $c \triangleleft x$ will be called *atomic constraints*. Constraints $x < -\infty$ and $+\infty < x$ are equivalent to *false* and constraints $-\infty \leq x$ and $x \leq +\infty$ are equivalent to *true*.

Given a finite alphabet Σ, we define a set $X_H = \{\overleftarrow{a} \mid a \in \Sigma\}$ of *history clocks* and a set $X_P = \{\overrightarrow{a} \mid a \in \Sigma\}$ of *prophecy clocks*. Together, history and prophecy clocks are called *event clocks*. In this paper, all clocks will be event clocks, thus we set $X = X_H \cup X_P$.

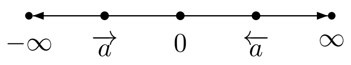


Figure 1 Range of valuations of event clocks. A valuation maps history clocks to $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ and prophecy clocks to $\mathbb{R}_{\leq 0} \cup \{-\infty\}$.

▶ **Definition 1** (Valuation). *A valuation of event clocks is a function $v \colon X \mapsto \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ which maps history clocks to $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ and prophecy clocks to $\mathbb{R}_{\leq 0} \cup \{-\infty\}$. We say a history clock $\overleftarrow{a}$, for some $a \in \Sigma$ is* undefined *(resp.* defined*) when $v(\overleftarrow{a}) = +\infty$ (resp. $v(\overleftarrow{a}) < +\infty$) and a prophecy clock $\overrightarrow{a}$ is undefined (resp. defined) when $v(\overrightarrow{a}) = -\infty$ (resp. $-\infty < v(\overrightarrow{a})$). We denote by $\mathbb{V}(X)$ or simply by $\mathbb{V}$ the set of valuations over X.*

We remark that the history clock and the prophecy clock of an event a are symmetric notions. In the semantics that we introduce in this paper, history clock $\overleftarrow{a}$ stores the amount of time elapsed after seeing the last a, measuring how far ahead in the future we are w.r.t. the last occurrence of a. Before we see an a for the first time, $\overleftarrow{a}$ is set to $+\infty$. The prophecy clock $\overrightarrow{a}$ stores the negative of the amount of time that needs to be elapsed before seeing the next a. In other words, $-\overrightarrow{a}$ tells us how far behind in the past we are w.r.t. the next occurrence of a. If no more a's are going to be seen, then the prophecy clock of a is set to $-\infty$, i.e., $\overrightarrow{a} = -\infty$. See Figure 1 for a pictorial representation of valuations of event clocks.

Notice that for history (resp. prophecy) clocks, *useful* constraints use non-negative (resp. non-positive) constants. Also, $\overleftarrow{a} < 0$ and $0 < \overrightarrow{a}$ are equivalent to *false* whereas $0 \leq \overleftarrow{a}$, $\overleftarrow{a} \leq \infty$, $\overrightarrow{a} \leq 0$ and $-\infty \leq \overrightarrow{a}$ are equivalent to *true*. A constraint $c \triangleleft \overleftarrow{a}$ does not imply that the history clock $\overleftarrow{a}$ is defined, whereas a constraint $\overleftarrow{a} \triangleleft c$ with $(\triangleleft, c) \neq (\leq, \infty)$ does. The same applies to prophecy clocks where a constraint $c \triangleleft \overrightarrow{a}$ with $(c, \triangleleft) \neq (-\infty, \leq)$ implies that $\overrightarrow{a}$ is defined, whereas $\overrightarrow{a} \triangleleft c$ does not; in fact, $\overrightarrow{a} \leq -\infty$ states that $\overrightarrow{a}$ is undefined.

▶ Remark 2. In the earlier works on ECA [5, 20], prophecy clocks assumed non-negative values and decreased along with time. This allowed to write guards on prophecy clocks with non-negative constants, e.g., $\overrightarrow{a} \leq 5$ means that the next a occurs in at most 5 time units. In our convention, this would be written as $-5 \leq \overrightarrow{a}$. Secondly, an undefined clock (history or prophecy) was assigned a special symbol $\perp$ in earlier works. We have changed this to use $-\infty$ and $+\infty$ for undefined prophecy and history clocks respectively. We adopt these new conventions as they allow to treat both history clocks and prophecy clocks in a symmetric fashion, and a clean integration of undefined values when we describe zones and simulations.

We say that a valuation v satisfies a constraint φ, denoted as $v \models \varphi$, if φ evaluates to *true*, when each variable x in φ is replaced by its value $v(x)$. We write $[\overleftarrow{a}]v$ to denote the valuation v' obtained from v by resetting the history clock $\overleftarrow{a}$ to 0, keeping the value of other clocks unchanged. We denote by $[\overrightarrow{a}]v$ the *set* of valuations v' obtained from v by setting the prophecy clock $\overrightarrow{a}$ non-deterministically to some value in $[-\infty, 0]$, keeping the value of other clocks unchanged. We denote by $v + \delta$ the valuation obtained by increasing the value of all clocks from the valuation v by $\delta \in \mathbb{R}_{\geq 0}$. Not every time elapse may be possible from a valuation, since prophecy clocks need to stay at most 0. For example, if there are two events a, b, then a valuation with $v(\overrightarrow{a}) = -3$ and $v(\overrightarrow{b}) = -2$ can elapse at most 2 time units.

▶ **Definition 3** (Event-clock automata [5]). *An* event-clock automaton (ECA) $\mathcal{A}$ *is given by a tuple* $(Q, \Sigma, X, T, q_0, F)$*, where* Q *is a finite set of states,* Σ *is a finite alphabet of actions,* X *is the set of event clocks for* Σ*,* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of accepting states and* $T \subseteq Q \times \Sigma \times \Phi(X) \times Q$ *is a finite set of transitions.*
The semantics of an ECA $\mathcal{A} = (Q, \Sigma, X, T, q_0, F)$ *is given by a transition system* $S_{\mathcal{A}}$ *whose states are* configurations (q, v) *of* $\mathcal{A}$*, where* $q \in Q$ *and* v *is a valuation. A configuration* (q, v) *is initial if* $q = q_0$*,* $v(x) = \infty$ *for all* $x \in X_H$ *and* $-\infty \leq v(x) \leq 0$ *for all* $x \in X_P$*. A configuration* (q, v) *is accepting if* $q \in F$*, and* $v(x) = -\infty$ *for all* $x \in X_P$ *and* $0 \leq v(x) \leq \infty$ *for all* $x \in X_H$*. Transitions of* $S_{\mathcal{A}}$ *are of two forms:*

- ***Delay transition:*** $(q, v) \xrightarrow{\delta} (q, v + \delta)$*, if* $(v + \delta)(x) \leq 0$ *for all* $x \in X_P$*.*
- ***Action transition:*** $(q, v) \xrightarrow{t} (q', [\overleftarrow{a}]v')$ *if* $t = (q, a, g, q')$ *is a transition in* $\mathcal{A}$*,* $v(\overrightarrow{a}) = 0$*,* $v' \in [\overrightarrow{a}]v$ *and* $v' \models g$*.*
  A transition with action a *can be taken when the value of the prophecy clock* $\overrightarrow{a}$ *is* 0*, then a new value in* $[-\infty, 0]$ *for* $\overrightarrow{a}$ *is non-deterministically guessed so that the resulting valuation* v' *satisfies the guard* g*, and finally, the history clock* $\overleftarrow{a}$ *is reset to* 0*.*

An ECA is called an event recording automaton (ERA) if it only contains history clocks and event predicting automaton (EPA) if it only contains prophecy clocks. A *run* of an event-clock automaton is a finite sequence of transitions from an initial configuration of $S_{\mathcal{A}}$. A run is said to be *accepting* if its last configuration is accepting. We are interested in the *reachability problem* of an event clock automaton. Formally,

▶ **Definition 4** (Reachability problem for ECA). *The reachability problem for an event-clock automaton* $\mathcal{A}$ *is to decide whether* $\mathcal{A}$ *has an accepting run.*

Different solutions based on regions and zones have been proposed in [5, 19, 20]. For ERA, the standard region and zone based algorithms for timed automata work directly. However, for EPA (and ECA), this is not the case. In fact, [19] show that the standard region abstraction is not possible, as there exists no finite bisimulation due to the behavior of prophecy clocks. Also, the standard definition of zones used for timed automata is not sufficient to handle valuations with undefined clocks. The papers [19, 20] make use of special symbols $\perp$ and ? for this purpose. In this work, we use a different formulation of zones by making use of $+\infty$ and $-\infty$. Instead of using $x = \perp$ (resp. $x \neq \perp$) to state that a clock is undefined (resp. defined) as in [19, 20], we write $+\infty \leq x$ or $x \leq -\infty$ or (resp. $x < +\infty$ or $-\infty < x$) depending on whether x is a history clock or a prophecy clock. This distinction between being undefined for history and prophecy clocks plays an important role.

3 Event zones and simulation based reachability

The most widely used approach for checking reachability in a timed automaton is based on reachability in a graph called the *zone graph* of a timed automaton [13]. Roughly, *zones* [7] are sets of valuations that can be represented efficiently using difference constraints between clocks. In this section, we introduce an analogous notion for event-clock automata. We consider *event zones*, which are sets of valuations of event-clock automata.

▶ **Definition 5** (Event zones). *An event zone is a set of valuations satisfying a conjunction of constraints of the form $c \triangleleft x$, $x \triangleleft c$ or $x - y \triangleleft c$, where $x, y \in X$ and $c \in \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$. Constraints of the form $x - y \triangleleft c$ are called diagonal constraints. To evaluate such constraints, we extend addition on real numbers with the convention that $(+\infty) + \alpha = +\infty$ for all $\alpha \in \overline{\mathbb{R}}$ and $(-\infty) + \beta = -\infty$, as long as $\beta \neq +\infty$. We simply write $v(x - y)$ for $v(x) - v(y)$.*

Let W be a set of valuations and q a state. For transition $t := (q, a, g, q_1)$, we write $(q, W) \xrightarrow{t} (q_1, W_1)$ if $W_1 = \{v_1 \mid (q, v) \xrightarrow{t} \xrightarrow{\delta} (q_1, v_1) \text{ for some } \delta \in \mathbb{R}_{\geq 0}\}$. As is usual with timed automata, zones are closed under the time elapse operation. We will show in the next section that starting from an event zone Z, the successors are also event zones: $(q, Z) \xrightarrow{t} (q_1, Z_1)$ implies Z_1 is an event zone too. We use this feature to define an event zone graph.

▶ **Definition 6** (Event zone graph). *Nodes are of the form (q, Z) where q is a state and Z is an event zone. The initial node is (q_0, Z_0) where q_0 is the initial state and Z_0 is given by $\bigwedge_{a \in \Sigma} (\infty \leq \overleftarrow{a}) \wedge (\overrightarrow{a} \leq 0)$. This is the set of all initial valuations, which is already closed under time elapse. For every node (q, Z) and every transition $t := (q, a, g, q_1)$ there is a transition $(q, Z) \xrightarrow{t} (q_1, Z_1)$ in the event zone graph. A node (q, Z) is accepting if $q \in F$ and $Z \cap Z_f$ is non-empty where the final zone Z_f is defined by $\bigwedge_{a \in \Sigma} \overrightarrow{a} \leq -\infty$.*

Two examples of ECA and their event zone graphs are given in Figure 3 and Figure 4 of Appendix A.

Similar to the case of timed automata, the event zone graph can be used to decide reachability. The next lemma follows by a straightforward adaptation of the corresponding proof [13] from timed automata.

▶ **Proposition 7.** *The event zone graph of an ECA is sound and complete for reachability.*

However, as in the case of zone graphs for timed automata, the event zone graph for an ECA is also not guaranteed to be finite. We will now define what a simulation is and see how it can be used to get a finite truncation of the event zone graph, which is still sound and complete for reachability.

▶ **Definition 8** (Simulation). *A simulation relation on the semantics of an ECA 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 two event 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 $(q_1, Z_1), (q_2, Z_2), \ldots$ of reachable *nodes, there exists $j > i$ such that $(q_j, Z_j) \preceq (q_i, Z_i)$.*

The reachability algorithm enumerates the nodes of the event zone graph and uses $\preceq$ to truncate nodes that are smaller with respect to the simulation.

▶ **Definition 9** (Reachability algorithm). *Let $\mathcal{A}$ be an ECA and $\preceq$ a finite simulation for $\mathcal{A}$. Add the initial node of the event zone graph (q_0, Z_0) to a Waiting list. Repeat the following until Waiting list is empty:*

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

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

The correctness of the reachability algorithm follows once again from the correctness of the simulation approach in timed automata [22]. Moreover, termination is guaranteed when the simulation used is finite.

▶ **Theorem 10.** *An ECA has an accepting run iff the reachability algorithm returns Yes.*

We have now presented the framework for the simulation approach in its entirety. However, to make it functional, we will need the following.

1. An efficient representation for event zones and algorithms to compute successors.
2. A concrete simulation relation $\preceq$ for ECA with an efficient simulation test $(q, Z) \preceq (q, Z')$.
3. A proof that $\preceq$ is finite, to guarantee termination of the reachability algorithm.

In the rest of the paper, we show how these can be achieved. To start with, for standard timed automata, zones are represented using Difference-Bound-Matrices (DBMs) [14]. For such a representation to work on event zones, we will need to incorporate the fact that valuations can now take $+\infty$ and $-\infty$. In Section 4, we propose a way to merge $+\infty$ and $-\infty$ seamlessly into the DBM technology. In the subsequent Section 5, we define a simulation for ECA based on $\mathcal{G}$-simulation, develop some technical machinery and present an efficient simulation test. Finally, in Section 6, we deal with the main problem of showing finiteness. For this, we prove some non-trivial invariants on the event zones that are reachable in ECA and use them to show a surprising property regarding prophecy clocks. More precisely, we show that constraints involving prophecy clocks in reachable event zones come from a finite set depending on the maximum constant of the ECA only.

4 Computing with event zones and distance graphs

We now show that event zones can be represented using Difference-Bound-Matrices (DBMs) and the operations required for the reachability algorithm can be implemented using DBMs. Each entry in a DBM encodes a constraint of the form $x - y \triangleleft c$. For timed automata analysis, the entries are $(\triangleleft, c)$ where $c \in \mathbb{R}$ and $\triangleleft \in \{<, \leq\}$, or $(\triangleleft, c) = (<, \infty)$. In our case, we will need to deal with valuations having $+\infty$ or $-\infty$. For this purpose, we first extend weights to include $(\leq, -\infty)$ and $(\leq, \infty)$ and define an arithmetic that admits the new entries in a natural way.

▶ **Definition 11** (Weights). *Let $\mathcal{C} = \{(\leq, -\infty)\} \cup \{(\triangleleft, c) \mid c \in \mathbb{R} \cup \{\infty\}$ and $\triangleleft \in \{\leq, <\}\}$, called the set of weights.*

- ***Order.*** *Define $(\triangleleft_1, c_1) < (\triangleleft_2, c_2)$ when either (1) $c_1 < c_2$, or (2) $c_1 = c_2$ and $\triangleleft_1$ is $<$ while $\triangleleft_2$ is $\leq$. This is a total order, in particular $(\leq, -\infty) < (\triangleleft, c) < (<, \infty) < (\leq, \infty)$ for all $c \in \mathbb{R}$.*
- ***Sum.*** *Let $\alpha, \beta, \gamma, (\triangleleft_1, c_1), (\triangleleft_2, c_2) \in \mathcal{C}$ with $\beta \neq (\leq, \infty)$, $\gamma \notin \{(\leq, -\infty), (\leq, \infty)\}$ and $c_1, c_2 \in \mathbb{R}$. We define the operation of* sum *on weights as follows.*

$$(\leq, \infty) + \alpha = (\leq, \infty) \qquad (\leq, -\infty) + \beta = (\leq, -\infty) \qquad (<, \infty) + \gamma = (<, \infty)$$

$$(\triangleleft_1, c_1) + (\triangleleft_2, c_2) = (\triangleleft, c_1 + c_2) \qquad \text{with } \triangleleft = \leq \text{ if } \triangleleft_1 = \triangleleft_2 = \leq \text{ and } \triangleleft = < \text{ otherwise.}$$

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

▶ **Lemma 12.** *Let v be a valuation, x, y, z be event clocks and $(\triangleleft_1, c_1), (\triangleleft_2, c_2) \in \mathcal{C}$. If $v \models x - y \triangleleft_1 c_1$ and $v \models y - z \triangleleft_2 c_2$, then $v \models x - z \triangleleft c$ where $(\triangleleft, c) = (\triangleleft_1, c_1) + (\triangleleft_2, c_2)$.*

Equipped with the weights and the arithmetic over it, we will work with a graph representation of zones (as so-called distance graphs), instead of matrices (i.e., DBMs), since this makes the analysis more convenient. We wish to highlight that our definition of weights, order and sum have been chosen to ensure that this notion of distance graphs remains identical to the one for usual TA. As a consequence, we are able to adapt many of the well-known properties about distance graphs for ECA.

▶ **Definition 13** (Distance graphs). *A distance graph is a weighted directed graph, with vertices being $X_P \cup X_H \cup \{0\}$ where 0 is a special vertex that plays the role of constant 0. Edges are labeled with weights from $\mathcal{C}$. An edge $x \xrightarrow{\triangleleft c} y$ represents the constraint $y - x \triangleleft c$. For a graph $\mathbb{G}$, we define $[\![\mathbb{G}]\!] := \{v \mid v \models y - x \triangleleft c$ for all edges $x \xrightarrow{\triangleleft c} y$ in $\mathbb{G}\}$. Further,*

- *The weight of a path in a distance graph $\mathbb{G}$ is the sum of the weight of its edges. A cycle in $\mathbb{G}$ is said to be negative if its weight is strictly less than $(\leq, 0)$.*
- *A graph $\mathbb{G}$ is said to be in canonical form if it has no negative cycles and for each pair of vertices x, y, the weight of $x \to y$ is not greater than the weight of any path from x to y.*
- *For two graphs $\mathbb{G}_1$, $\mathbb{G}_2$, we write $\min(\mathbb{G}_1, \mathbb{G}_2)$ for the distance graph obtained by setting the weight of each edge to the minimum of the corresponding weights in $\mathbb{G}_1$ and $\mathbb{G}_2$.*

For an event zone Z, we write $\mathbb{G}(Z)$ for the canonical distance graph that satisfies $[\![\mathbb{G}(Z)]\!] = Z$. We denote by Z_{xy} the weight of the edge $x \to y$ in $\mathbb{G}(Z)$.

We will make use of an important property, which has been shown when weights come from $\mathcal{C} \setminus \{(\leq, +\infty), (\leq, -\infty)\}$, but continues to hold even with the new weights added.

▶ **Lemma 14.** *For every distance graph $\mathbb{G}$, we have $[\![\mathbb{G}]\!] = \emptyset$ iff $\mathbb{G}$ has a negative cycle.*

Successor computation. To implement the computation of transitions $(q, Z) \xrightarrow{t} (q_1, Z_1)$ in an event zone graph, we will make use of some operations on event zones that we define below. Using distance graphs, we show that these operations preserve event zones, that is, starting from an event zone and applying any of the operations leads to an event zone again. Thanks to the algebra over the new weights that we have defined, the arguments are very similar to the case of standard timed automata.

▶ **Definition 15** (Operations on event zones). *Let g be a guard and Z an event zone.*

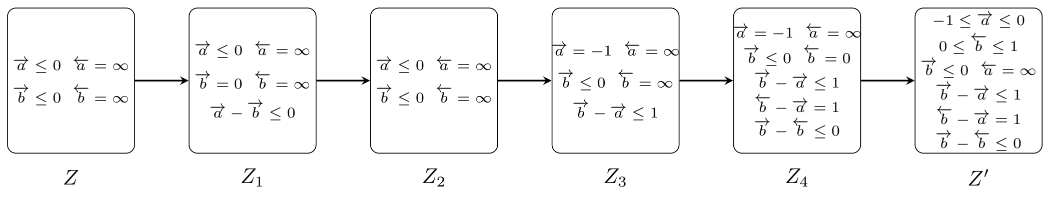
- *Guard intersection: $Z \wedge g := \{v \mid v \in Z$ and $v \models g\}$*
- *Release: $[\overrightarrow{a}]Z = \bigcup_{v \in Z} [\overrightarrow{a}]v$*
- *Reset: $[\overleftarrow{a}]Z = \{[\overleftarrow{a}]v \mid v \in Z\}$*
- *Time elapse: $\overrightarrow{Z} = \{v + \delta \mid v \in Z, \delta \in \mathbb{R}_{\geq 0}$ s.t. $v + \delta \models \bigwedge_{a \in \Sigma} \overrightarrow{a} \leq 0\}$*

A guard g can be seen as yet another event zone and hence guard intersection is just an intersection operation between two event zones. By definition, for a transition $t := (q, a, g, q')$ and a node (q, Z) the successor $(q, Z) \xrightarrow{t} (q', Z')$ can be computed in the following sequence:

$$Z_1 := Z \cap (0 \leq \overrightarrow{a}) \qquad Z_2 := [\overrightarrow{a}]Z_1 \qquad Z_3 := Z_2 \cap g \qquad Z_4 := [\overleftarrow{a}]Z_3 \qquad Z' := \overrightarrow{Z_4}$$

As an example, in Figure 2, suppose an action b with guard $\overrightarrow{a} = -1$ ($\overrightarrow{a} \leq -1 \wedge -1 \leq \overrightarrow{a}$) is fired from Zone Z as depicted, applying the above sequence in order gives Z_1, Z_2, Z_3, Z_4 resulting in the successor zone Z', as depicted in the figure.

We are now ready to state Theorem 16 that says that the operations on event zones translate easily to operations on distance graphs and that the successor of an event zone is an event zone. Except for the release operation $[\overleftarrow{a}]$, the rest of the operations are standard in timed automata, but need to be extended to cope with the new weights $(\leq, +\infty), (\leq, -\infty)$.



$\overrightarrow{a} \leq 0$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} \leq 0$ $\overleftarrow{b} = \infty$
Z
$\overrightarrow{a} \leq 0$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} = 0$ $\overleftarrow{b} = \infty$
$\overrightarrow{a} - \overrightarrow{b} \leq 0$
Z_1
$\overrightarrow{a} \leq 0$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} \leq 0$ $\overleftarrow{b} = \infty$
Z_2
$\overrightarrow{a} = -1$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} \leq 0$ $\overleftarrow{b} = \infty$
$\overrightarrow{b} - \overrightarrow{a} \leq 1$
Z_3
$\overrightarrow{a} = -1$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} \leq 0$ $\overleftarrow{b} = 0$
$\overrightarrow{b} - \overrightarrow{a} \leq 1$
$\overleftarrow{b} - \overrightarrow{a} = 1$
$\overrightarrow{b} - \overleftarrow{b} \leq 0$
Z_4
$-1 \leq \overrightarrow{a} \leq 0$
$0 \leq \overleftarrow{b} \leq 1$
$\overrightarrow{b} \leq 0$ $\overleftarrow{a} = \infty$
$\overrightarrow{b} - \overrightarrow{a} \leq 1$
$\overleftarrow{b} - \overrightarrow{a} = 1$
$\overrightarrow{b} - \overleftarrow{b} \leq 0$
Z'

Figure 2 Successor computation from event zone Z on an action b with guard $\overrightarrow{a} = -1$.

We show that we can perform all these operations in the new algebra with quadratic complexity, as in timed automata without diagonal constraints [32].

▶ **Theorem 16.** *Let Z be an event zone and $\mathbb{G}$ be its canonical distance graph. Let g be a guard. We can compute, in $\mathcal{O}(|X_P \cup X_H|^2)$ time, distance graphs $\mathbb{G}_g$, $[\overrightarrow{a}]\mathbb{G}$, $[\overleftarrow{a}]\mathbb{G}$ and $\overrightarrow{\mathbb{G}}$ in canonical form, such that $Z \wedge g = [\![\mathbb{G}_g]\!]$, $[\overrightarrow{a}]Z = [\![[\overrightarrow{a}]\mathbb{G}]\!]$, $[\overleftarrow{a}]Z = [\![[\overleftarrow{a}]\mathbb{G}]\!]$, and $\overrightarrow{Z} = [\![\overrightarrow{\mathbb{G}}]\!]$.*

Proof (sketch). The distance graphs $\mathbb{G}_g$, $[\overrightarrow{a}]\mathbb{G}$, $[\overleftarrow{a}]\mathbb{G}$ and $\overrightarrow{\mathbb{G}}$ are computed as follows:

- Guard intersection: a distance graph $\mathbb{G}_g$ is obtained from $\mathbb{G}$ as follows,
  - for each atomic constraint $x \triangleleft c$ in g, replace weight of edge $0 \to x$ with the minimum of its weight in $\mathbb{G}$ and $(\triangleleft, c)$,
  - for each atomic constraint $d \triangleleft y$ in g, replace weight of edge $y \to 0$ with the minimum of its weight in $\mathbb{G}$ and $(\triangleleft, -d)$,
  - canonicalize the resulting graph.
- Release: a distance graph $[\overrightarrow{a}]\mathbb{G}$ is obtained from $\mathbb{G}$ by
  - removing all edges involving $\overrightarrow{a}$ and then
  - adding the edges $0 \xrightarrow{(\leq,0)} \overrightarrow{a}$ and $\overrightarrow{a} \xrightarrow{(\leq,\infty)} 0$, and then
  - canonicalizing the resulting graph.
- Reset: a distance graph $[\overleftarrow{a}]\mathbb{G}$ is obtained from $\mathbb{G}$ by
  - removing all edges involving $\overleftarrow{a}$ and then
  - adding the edges $0 \xrightarrow{(\leq,0)} \overleftarrow{a}$ and $\overleftarrow{a} \xrightarrow{(\leq,0)} 0$, and then
  - canonicalizing the resulting graph.
- Time elapse: the distance graph $\overrightarrow{\mathbb{G}}$ is obtained by the following transformation:
  - if $\overleftarrow{x}$ is defined, i.e., the weight of $0 \to \overleftarrow{x}$ is not $(\leq, \infty)$, then replace it with $(<, \infty)$,
  - if $\overrightarrow{x}$ is defined, i.e., the weight of $0 \to \overrightarrow{x}$ is not $(\leq, -\infty)$, then replace it with $(\leq, 0)$,
  - canonicalize the resulting graph.

It is not hard to prove that they correspond to the operations on event zones. Other than canonicalization, it can be easily checked that these operations can be computed in quadratic time. Though canonicalization is cubic time in general, in each of the special cases above, it can be implemented in quadratic time. ◀

5 A concrete simulation relation for ECAs

We fix an event-clock automaton $\mathcal{A} = (Q, \Sigma, X, T, q_0, F)$ for this section. We will define a simulation relation $\preceq_{\mathcal{A}}$ on the configurations of the ECA. We first define a map $\mathcal{G}$ from Q to sets of atomic constraints. The map $\mathcal{G}$ is obtained as the least fixpoint of the set of equations:

$$\mathcal{G}(q) = \{\overrightarrow{b} \leq 0, 0 \leq \overrightarrow{b} \mid b \in \Sigma\} \cup \bigcup_{(q,a,g,q') \in T} \mathsf{split}(g) \cup \mathsf{pre}(a, \mathcal{G}(q'))$$

where $\mathsf{split}(g)$ is the set of atomic constraints occurring in g and, for a set of atomic constraints G, $\mathsf{pre}(a, G)$ is defined as the set of constraints in G except those on $\overrightarrow{a}$ or $\overleftarrow{a}$. Notice that constraints in $\mathcal{G}(q)$ use the constant 0 and constants used in constraints of $\mathcal{A}$.

Let G be a set of atomic constraints. The preorder $\preceq_G$ is defined on valuations by

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

Notice that in the condition above, we *do not* restrict δ to those such that $v+\delta$ is a valuation: we may have $v(\overrightarrow{a}) + \delta > 0$ for some $a \in \Sigma$. This is crucial for the proof of Theorem 17 below. It also allows to get a clean characterizations of the simulation (Lemma 18) which in turn is useful for deriving the simulation test and in showing finiteness. Based on $\preceq_G$ and the $\mathcal{G}(q)$ computation, we can define a preorder $\preceq_{\mathcal{A}}$ between configurations of ECA $\mathcal{A}$ as $(q, v) \preceq_{\mathcal{A}} (q', v')$ if $q = q'$ and $v \preceq_{\mathcal{G}(q)} v'$.

▶ **Theorem 17.** *The relation $\preceq_{\mathcal{A}}$ is a simulation on the transition system $S_{\mathcal{A}}$ of ECA $\mathcal{A}$.*

When $G = \{\varphi\}$ is a singleton, we simply write $\preceq_\varphi$ for $\preceq_{\{\varphi\}}$. The definition of the $\preceq_G$ simulation above in some sense declares what is expected out of the simulation. Below, we give a constructive characterization of the simulation in terms of the constants used and the valuations. For example, if $v(\overleftarrow{a}) = 3$ and $\overleftarrow{a} \leq 5$ is a constraint in G, point 2 below says that all v' with $v'(\overleftarrow{a}) \leq 3$ simulate v. The next lemma is a generalization of Lemma 8 from [18] to our setting containing prophecy clocks and the undefined values $+\infty$ and $-\infty$.

▶ **Lemma 18.** *Let v, v' be valuations and G a set of atomic constraints. We have*
1. *$v \preceq_G v'$ iff $v \preceq_\varphi v'$ for all $\varphi \in G$.*
2. *$v \preceq_{x \triangleleft c} v'$ iff $v(x) \not\triangleleft c$ or $v'(x) \leq v(x)$ or $(\triangleleft, c) = (\leq, \infty)$ or $(\triangleleft, c) = (<, \infty) \wedge v'(x) < \infty$.*
3. *$v \preceq_{c \triangleleft x} v'$ iff $c \triangleleft v'(x)$ or $v(x) \leq v'(x)$ or $(c, \triangleleft) = (\infty, <)$ or $(c, \triangleleft) = (\infty, \leq) \wedge v(x) < \infty$.*

We now state some useful properties that get derived from Lemma 18.

▶ Remark 19. Let v, v' be valuations and G a set of atomic constraints.
1. For all $a \in \Sigma$, if $\{0 \leq \overrightarrow{a}, \overrightarrow{a} \leq 0\} \subseteq G$ and $v \preceq_G v'$ then $v(\overrightarrow{a}) = v'(\overrightarrow{a})$.
2. Let $x \triangleleft_1 c_1$ and $x \triangleleft_2 c_2$ be constraints with $(\triangleleft_1, c_1) \leq (\triangleleft_2, c_2) < (<, \infty)$ (we say that $x \triangleleft_1 c_1$ is subsumed by $x \triangleleft_2 c_2$). If $v \preceq_{x \triangleleft_2 c_2} v'$ then $v \preceq_{x \triangleleft_1 c_1} v'$.
   Indeed, from $(\triangleleft_2, c_2) < (<, \infty)$ and $v \preceq_{x \triangleleft_2 c_2} v'$ we get $v'(x) \leq v(x)$ or $v(x) \not\triangleleft_2 c_2$, which implies $v(x) \not\triangleleft_1 c_1$ since $(\triangleleft_1, c_1) \leq (\triangleleft_2, c_2)$.
3. Let $c_1 \triangleleft_1 x$ and $c_2 \triangleleft_2 x$ be constraints with $(c_1, \triangleleft_1) \leq (c_2, \triangleleft_2) < (\infty, \leq)$ (we say that $c_1 \triangleleft_1 x$ is subsumed by $c_2 \triangleleft_2 x$). If $v \preceq_{c_2 \triangleleft_2 x} v'$ then $v \preceq_{c_1 \triangleleft_1 x} v'$.
   Indeed, from $(c_2, \triangleleft_2) < (\infty, \leq)$ and $v \preceq_{c_2 \triangleleft_2 x} v'$ we get $v(x) \leq v'(x)$ or $c_2 \triangleleft_2 v'(x)$, which implies $c_1 \triangleleft_1 v'(x)$ since $(c_1, \triangleleft_1) \leq (c_2, \triangleleft_2)$.
   The ordering between *lower weights* is defined by $(c_1, \triangleleft_1) < (c_2, \triangleleft_2)$ if $c_1 < c_2$ or $c_1 = c_2$, $\triangleleft_1 = \leq$ and $\triangleleft_2 = <$. We have $(c_1, \triangleleft_1) < (c_2, \triangleleft_2)$ iff $(\triangleleft_2, -c_2) < (\triangleleft_1, -c_1)$.

Before lifting the simulation to event zones, we present a central technical object that will be used from time to time in the next set of results.

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

We assume that G contains $\{0 \leq \overrightarrow{a}, \overrightarrow{a} \leq 0 \mid a \in \Sigma\}$ so that $v \preceq_G v'$ implies $v(\overrightarrow{a}) = v'(\overrightarrow{a})$ for all prophecy clocks $\overrightarrow{a}$ with $a \in \Sigma$. We remove from G constraints equivalent to true, such as $x \leq \infty$, $-3 < \overleftarrow{a}$ or $0 \leq \overleftarrow{a}$, or equivalent to false, such as $\overleftarrow{a} < 0$ or $\infty < x$. Also, by

Remark 19, we may remove from G constraints that are subsumed by other constraints in G, while not changing the simulation relation. Hence, for history clocks, we have at most one upper-bound constraint $\overleftarrow{a} \triangleleft c$ with $(\leq, 0) \leq (\triangleleft, c) < (<, \infty)$, and at most one lower-bound constraint $c \triangleleft \overleftarrow{a}$ with $(0, \leq) < (c, \triangleleft) < (\infty, \leq)$. From now on, we always assume that the sets G of atomic constraints that we consider satisfy the above conditions.

The definition of the distance graph $\mathbb{G}_G(v)$ which defines $\uparrow_G v$ is based on Lemma 18.

- For each prophecy clock $\overrightarrow{a}$, we have the edges $\overrightarrow{a} \xrightarrow{(\leq, -v(\overrightarrow{a}))} 0$ and $0 \xrightarrow{(\leq, v(\overrightarrow{a}))} \overrightarrow{a}$.
- For each history clock $\overleftarrow{a}$, we have the edge $0 \to \overleftarrow{a}$ with weight
  - $(\leq, v(\overleftarrow{a}))$ if $\overleftarrow{a} \triangleleft c \in G$ with $(\triangleleft, c) < (<, \infty)$ and $v(\overleftarrow{a}) \triangleleft c$,
  - $(<, \infty)$ if we are not in the case above and $\overleftarrow{a} < \infty \in G$, $v(\overleftarrow{a}) < \infty$,
  - $(\leq, \infty)$ otherwise.
- For each history clock $\overleftarrow{a}$, we have the edge $\overleftarrow{a} \to 0$ with weight
  - $(\leq, -\infty)$ if $\infty \leq \overleftarrow{a} \in G$ and $v(\overleftarrow{a}) = \infty$, and if we are not in this case:
  - $(\triangleleft, -c)$ if $c \triangleleft \overleftarrow{a} \in G$ with $(c, \triangleleft) < (\infty, \leq)$ and $c \triangleleft v(\overleftarrow{a})$,
  - $(\leq, -v(\overleftarrow{a}))$ if $c \triangleleft \overleftarrow{a} \in G$ with $(c, \triangleleft) < (\infty, \leq)$ and $c \not\triangleleft v(\overleftarrow{a})$,
  - $(\leq, 0)$ otherwise.

With this definition, while $\mathbb{G}_G(v)$ is not in canonical form, it has the desired property:

▶ **Lemma 20.** *We have $v \preceq_G v'$ iff v' satisfies all the constraints of $\mathbb{G}_G(v)$.*

Simulation for event zones and an efficient algorithmic check. Let Z, Z' be two event zones and G be a set of atomic constraints. We say that Z is G-simulated by Z', denoted $Z \preceq_G Z'$, if for all $v \in Z$ there exists $v' \in Z'$ such that $v \preceq_G v'$. Finally, we define $(q, Z) \preceq_{\mathcal{A}} (q', Z')$ if $q = q'$ and $Z \preceq_{\mathcal{G}(q)} Z'$. In the rest of this section, we show how to check this relation efficiently. We let $\downarrow_G Z = \{v \in \mathbb{V} \mid v \preceq_G v' \text{ for some } v' \in Z\}$. Notice that $Z \preceq_G Z'$ iff $Z \subseteq \downarrow_G Z'$ iff $\downarrow_G Z = \downarrow_G Z'$.

▶ **Lemma 21.** *For event zones Z, Z', we have $Z \not\preceq_G Z'$ iff $\exists v \in Z$ with $\uparrow_G v \cap Z' = \emptyset$.*

To check $Z \not\preceq_G Z'$, we require a valuation $v \in Z$ with a witness that $\uparrow_G v \cap Z'$ is empty. In the language of distance graphs, the witness will be a negative cycle in $\min(\uparrow_G v, Z')$. We show that if $\uparrow_G v \cap Z'$ is empty, then there is a small witness, i.e., a negative cycle containing at most three edges, and belonging to one of three specific forms.

▶ **Lemma 22.** *Let v be a valuation, Z' a non-empty reachable event zone with canonical distance graph $\mathbb{G}(Z')$ and G a set of atomic constraints. Then, $\uparrow_G v \cap Z'$ 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 $\mathbb{G}_G(v)$ and $x \to 0$ from $\mathbb{G}(Z')$,*
2. *$0 \to y \to 0$ with $0 \to y$ from $\mathbb{G}(Z')$ and $y \to 0$ from $\mathbb{G}_G(v)$, and*
3. *$0 \to x \to y \to 0$, with weight of $x \to y$ from $\mathbb{G}(Z')$ and the others from $\mathbb{G}_G(v)$. Moreover, this negative cycle has finite weight.*

Proof. Since $Z' \neq \emptyset$, the distance graph $\mathbb{G}(Z')$ has no negative cycle. The same holds for $\mathbb{G}_G(v)$ since $v \in \uparrow_G v \neq \emptyset$. We know that $\uparrow_G v \cap Z' = \emptyset$ iff there is a (simple) negative cycle using edges from $\mathbb{G}_G(v)$ and from $\mathbb{G}(Z')$. Since $\mathbb{G}(Z')$ is in canonical form, we may restrict to negative cycles which do not use two consecutive edges from $\mathbb{G}(Z')$. Now all edges of $\mathbb{G}_G(v)$ are adjacent to node 0. Hence, if a simple cycle uses an edge from $\mathbb{G}(Z')$ which is adjacent to 0, it consists of only two edges $0 \to x \to 0$, one from $\mathbb{G}(Z')$ and one from $\mathbb{G}_G(v)$. Otherwise, the simple cycle is of the form $0 \to x \to y \to 0$ where the edge $x \to y$ is from $\mathbb{G}(Z')$ and the other two edges are from $\mathbb{G}_G(v)$. It remains to show that the two clock negative cycle $0 \to x \to y \to 0$ can be considered to have finite weight, i.e., weight is not $(\leq, -\infty)$.

For the cycle to have weight $(\leq, -\infty)$, one of the edges should have weight $(\leq, -\infty)$ and the others should have a weight different from $(\leq, \infty)$. We will show that for every such combination, there is a smaller negative cycle with a single clock and 0. Hence we can ignore negative cycles of the form $0 \to x \to y \to 0$ with weight $(\leq, -\infty)$.

Suppose $Z'_{xy} = (\leq, -\infty)$. Then, for every valuation in $u \in Z'$, we have $u(y) - u(x) \leq -\infty$, which implies $u(y) = -\infty$ or $u(x) = +\infty$. If $u(x) = +\infty$ for some valuation $u \in Z'$, then the value of x is $+\infty$ for every valuation in Z'. This follows from the successor computation: initially, history clocks are undefined, and then an action a defines $\overleftarrow{a}$, and from that point onwards, $\overleftarrow{a}$ is always $< \infty$. Now, if x is not an undefined history clock in Z', then we need to have $u(y) = -\infty$ for all valuations of Z'. Therefore, either x is a history clock that is undefined in Z' or y is a prophecy clock that is undefined in Z'. In the former case, $Z'_{x0} = (\leq, -\infty)$ and in the latter case $Z'_{0y} = (\leq, -\infty)$. This gives a smaller negative cycle $0 \to x \xrightarrow{Z'_{x0}} 0$ or $0 \xrightarrow{Z'_{0y}} y \to 0$ with the remaining edge $0 \to x$ or $y \to 0$ coming from $\mathbb{G}_G(v)$, since by our hypothesis of a negative cycle, these edges have weight different from $(\leq, \infty)$.

Suppose the weight of $0 \to x$ is $(\leq, -\infty)$. This can happen only when x is a prophecy clock, $v(x) = -\infty$ and weight of $0 \to x$ is $(\leq, v(x))$. Since $Z'_{xy} \neq (\leq, \infty)$, we infer $Z'_{x0} \neq (\leq, \infty)$ by $\dagger_1$ of Lemma 26. Hence $0 \xrightarrow{(\leq, v(x))} x \xrightarrow{Z'_{x0}} 0$ is also a negative cycle.

Suppose $y \to 0$ has weight $(\leq, -\infty)$. This can happen only when y is a history clock and $v(y) = +\infty$. Since $Z'_{xy} \neq (\leq, \infty)$, we obtain $Z'_{0y} \neq (\leq, \infty)$ and hence $0 \xrightarrow{Z'_{0y}} y \xrightarrow{(\leq, -v(y))} 0$ is a negative cycle. ◀

We now have all the results required to state our inclusion test. Using the above lemma, and relying on a careful analysis (as shown in the full version [2]), we obtain the following theorem.

▶ **Theorem 23.** *Let Z, Z' be non-empty reachable zones, and G a set of atomic constraints containing $\overrightarrow{a} \leq 0$ and $0 \leq \overrightarrow{a}$ for every prophecy clock $\overrightarrow{a}$. Then, $Z \not\preceq_G Z'$ iff one of the following conditions holds:*

1. *$Z'_{x0} < Z_{x0}$ for some prophecy clock x, or for some history clock x with*
   - *$(x < \infty) \in G$ and $Z'_{x0} = (\leq, -\infty)$, or*
   - *$(x \lhd_1 c) \in G$ for $c \in \mathbb{N}$ and $(\leq, 0) \leq Z_{x0} + (\lhd_1, c)$.*
2. *$Z'_{0y} < Z_{0y}$ for some prophecy clock y, or for some history clock y with*
   - *$(\infty \leq y) \in G$ and $Z_{0y} = (\leq, \infty)$, or*
   - *$(d \lhd_2 y) \in G$ for $d \in \mathbb{N}$ and $Z'_{0y} + (\lhd_2, -d) < (\leq, 0)$*
3. *$Z'_{xy} < Z_{xy}$ and Z'_{xy} is finite for two distinct (prophecy or history) clocks x, y with $(x \lhd_1 c), (d \lhd_2 y) \in G$ for $c, d \in \mathbb{N}$ and $(\leq, 0) \leq Z_{x0} + (\lhd_1, c)$ and $Z'_{xy} + (\lhd_2, -d) < Z_{x0}$.*

From Theorem 23, we can see that the inclusion test requires iteration over clocks x, y and checking if the conditions are satisfied by the respective weights.

▶ **Corollary 24.** *Checking if $(q, Z) \preceq_{\mathcal{A}} (q', Z')$ can be done in time $\mathcal{O}(|X|^2) = \mathcal{O}(|\Sigma|^2)$.*

6 Finiteness of the simulation relation

In this section, we will show that the simulation relation $\preceq_{\mathcal{A}}$ defined in Section 5 is finite, which implies that the reachability algorithm of Definition 9 terminates. Recall that given an event clock automaton $\mathcal{A}$, we have an associated map $\mathcal{G}$ from states of $\mathcal{A}$ to sets of atomic constraints. Let $M = \max\{|c| \mid c \in \mathbb{Z} \text{ is used in some constraint of } \mathcal{A}\}$, the maximal constant of $\mathcal{A}$. We have $M \in \mathbb{N}$ and constraints in the sets $\mathcal{G}(q)$ use constants in $\{-\infty, \infty\} \cup \{c \in \mathbb{Z} \mid |c| \leq M\}$.

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

1. We prove in Lemma 26 that for any *reachable* node (q, Z) of $\mathsf{EZG}(\mathcal{A})$, the distance graph $\mathbb{G}(Z)$ in canonical form satisfies a set of $(\dagger)$ conditions which depend only on the maximal constant M of $\mathcal{A}$.
2. We introduce an equivalence relation $\sim_M$ of *finite index* on valuations (depending on M only) and show in Lemma 28 that, if G is a set of atomic constraints using constants in $\{c \in \mathbb{Z} \mid |c| \leq M\} \cup \{-\infty, \infty\}$ and if Z is a zone such that its distance graph $\mathbb{G}(Z)$ in canonical form satisfies $(\dagger)$ conditions, then $\downarrow_G Z$ is a union of $\sim_M$ equivalence classes.

We start with a lemma which highlights an important property of *prophecy* clocks in reachable event zones. This property is essential for the proof of the $(\dagger)$ conditions. The proof follows from the observation that the property is true in the initial zone, and is invariant under the zone operations, namely, guard intersection, reset, release and time elapse.

▶ **Lemma 25.** *Let Z be a reachable event zone. For every valuation $v \in Z$, and for every prophecy clock $\overrightarrow{x}$, if $-\infty < v(\overrightarrow{x}) < -M$, then $v[\overrightarrow{x} \mapsto \alpha] \in Z$ for every $-\infty < \alpha < -M$.*

There is no similar version of the above lemma for history clocks. A reset of a history clock makes its value exactly equal to 0 in every valuation and creates non-trivial diagonal constraints with other clocks. Moreover repeated resets can generate arbitrarily large diagonal constraints, for e.g., a loop with guard $x = 1$ and reset x. This is why simulations are particularly needed to control history clocks. Notice that in our simulation $v \preceq_G v'$, we have $v(\overrightarrow{a}) = v'(\overrightarrow{a})$: there is no abstraction of the value of prophecy clocks and the simulation relation by itself does not have any means to show finiteness. However, as we show below, the reachable zones themselves take care of finiteness with respect to prophecy clocks. The challenge is then to combine this observation on prophecy clocks along with the non-trivial simulation happening for history clocks to prove that we still get a finite simulation. This is the purpose of the above mentioned item 2.

Now, we give the $(\dagger)$ conditions and prove that they are satisfied by distance graphs of reachable zones. In particular, the $(\dagger)$ conditions imply that the weight of edges of the form $0 \to \overrightarrow{x}$, $\overrightarrow{x} \to 0$ and $\overrightarrow{x} \to \overrightarrow{y}$ belong to the finite set $\{(\leq, -\infty), (<, \infty), (\leq, \infty)\} \cup \{(\triangleleft, c) \mid c \in \mathbb{Z} \wedge -M \leq c \leq M\}$. For an example, see Figure 4. Thus, we obtain as a corollary that, for EPA, we do not even need simulation to obtain finiteness.

▶ **Lemma 26.** *Let (q, Z) be a* reachable *node in $\mathsf{EZG}(\mathcal{A})$ with $Z \neq \emptyset$. Then, the distance graph $\mathbb{G}(Z)$ in canonical form satisfies the $(\dagger)$ conditions:*

$\dagger_1$ *If $Z_{\overrightarrow{x}0} = (\leq, \infty)$ then $Z_{\overrightarrow{x}y} = (\leq, \infty)$ for all $y \neq \overrightarrow{x}$.*

$\dagger_2$ *If $Z_{\overrightarrow{x}0} = (<, \infty)$ then for all $y \neq \overrightarrow{x}$, either y is a prophecy clock which is undefined in Z and $Z_{\overrightarrow{x}y} = Z_{0y} = (\leq, -\infty)$ or $Z_{\overrightarrow{x}y} \in \{(<, \infty), (\leq, \infty)\}$.*

$\dagger_3$ *If $Z_{\overrightarrow{x}0} < (<, \infty)$ then $(\leq, 0) \leq Z_{\overrightarrow{x}0} \leq (\leq, M)$.*

$\dagger_4$ *If $Z_{\overrightarrow{x}\overleftarrow{y}} < (<, \infty)$ then $(\leq, 0) \leq Z_{\overrightarrow{x}0} \leq (\leq, M)$.*

$\dagger_5$ *Either $Z_{0\overrightarrow{y}} = (\leq, -\infty)$ ($\overrightarrow{y}$ is undefined in Z), or $Z_{x0} + (<, -M) \leq Z_{x\overrightarrow{y}}$ for all $x \neq \overrightarrow{y}$ (including $x = 0$).*

$\dagger_6$ *Either $Z_{0\overrightarrow{x}} = (\leq, -\infty)$ or $(<, -M) \leq Z_{0\overrightarrow{x}} \leq (\leq, 0)$.*

$\dagger_7$ *Either $Z_{\overrightarrow{x}\overrightarrow{y}} \in \{(\leq, -\infty), (<, \infty), (\leq, \infty)\}$ or $(<, -M) \leq Z_{\overrightarrow{x}\overrightarrow{y}} \leq (\leq, M)$.*

Proof sketch. $\dagger_4$ follows immediately from $\dagger_1, \dagger_2, \dagger_3$ and $\dagger_6, \dagger_7$ can be inferred from $\dagger_5$ and the other conditions. So here, we focus on $\dagger_1, \dagger_2$, $\dagger_3$ and partially the case of $\dagger_5$, leaving other details to the full version [2].

For $\dagger_1$, since $Z_{\overrightarrow{x}0} = (\leq, \infty)$, there is a valuation $v \in Z$ with $v(\overrightarrow{x}) = -\infty$. Therefore, for every clock $y \neq \overrightarrow{x}$, we have $v(y - \overrightarrow{x}) = +\infty$. Since $v \in Z$, it satisfies the constraint on $y - \overrightarrow{x}$ given by $Z_{\overrightarrow{x}y}$. This is possible only when $Z_{\overrightarrow{x}y} = (\leq, \infty)$.

For $\dagger_2$, assume that $Z_{\overrightarrow{x}0} = (<, \infty)$ and let $y \neq \overrightarrow{x}$. Consider first the case $Z_{0y} = (\leq, -\infty)$, i.e., y is a prophecy clock which is undefined in Z. Then, since $\mathbb{G}(Z)$ is in canonical form, we have $Z_{\overrightarrow{x}y} \leq Z_{\overrightarrow{x}0} + Z_{0y} = (<, \infty) + (\leq, -\infty) = (\leq -\infty)$. The second case is when $Z_{0y} \neq (\leq, -\infty)$. This implies $Z_{\overrightarrow{x}y} \neq (\leq, -\infty)$ since otherwise we would get $Z_{0y} \leq Z_{0\overrightarrow{x}} + Z_{\overrightarrow{x}y} = (\leq, -\infty)$. We claim that there is a valuation $v \in Z$ with $-\infty < v(y)$ and $-\infty < v(\overrightarrow{x}) < -M$. Consider the distance graph $\mathbb{G}'$ obtained from $\mathbb{G}(Z)$ by setting the weight of edge $y \to 0$ to $\min(Z_{y0}, (<, \infty))$ and of edge $0 \to \overrightarrow{x}$ to $\min(Z_{0\overrightarrow{x}}, (<, -M))$. 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 $(<, -M)$ of $0 \to \overrightarrow{x}$ or the new weight $(<, \infty)$ of $y \to 0$ or both. This gives the cycle $0 \to \overrightarrow{x} \to 0$ with weight $(<, -M) + Z_{\overrightarrow{x}0} = (<, \infty)$ since $Z_{\overrightarrow{x}0} = (<, \infty)$, the cycle $0 \to y \to 0$ with weight $Z_{0y} + (<, \infty)$ which is not negative since $Z_{0y} \neq (\leq, -\infty)$, and the cycle $y \to 0 \to \overrightarrow{x} \to y$ with weight $(<, \infty) + (<, -M) + Z_{\overrightarrow{x}y}$ which is not negative since $Z_{\overrightarrow{x}y} \neq (\leq, -\infty)$. Since $\mathbb{G}'$ has no negative cycle, Lemma 14 implies $[\![\mathbb{G}']\!] \neq \emptyset$. Note that $[\![\mathbb{G}']\!] \subseteq [\![\mathbb{G}(Z)]\!] = Z$. Finally, for all $v \in \mathbb{G}'$, we have $-\infty < v(y)$ and $-\infty < v(\overrightarrow{x}) < -M$, which proves the claim. By Lemma 25, $v_\alpha = v[\overrightarrow{x} \mapsto \alpha] \in Z$ for all $-\infty < \alpha < -M$. Now, $v_\alpha(y - \overrightarrow{x}) = v(y) - \alpha$ satisfies the constraint $Z_{\overrightarrow{x}y}$. We deduce that $Z_{\overrightarrow{x}y}$ is either $(<, \infty)$ or $(\leq, \infty)$.

Next, we turn to $\dagger_3$. Suppose $Z_{\overrightarrow{x}0} = (\triangleleft, c)$ for some integer $c > M$. Then, there exists a valuation $v \in Z$ with $v(\overrightarrow{x}) = -c$ or $v(\overrightarrow{x}) = -c + \frac{1}{2}$ depending on whether $\triangleleft$ is $\leq$ or $<$. Since c, M are integers, we get $-\infty < v(\overrightarrow{x}) < -M$. By Lemma 25, $v[\overrightarrow{x} \mapsto \alpha] \in Z$ for all $-\infty < \alpha < -M$. In particular, $v' = v[\overrightarrow{x} \mapsto -c - 1] \in Z$. For this valuation, we have $v'(\overrightarrow{x}) = -c - 1$. This violates $Z_{\overrightarrow{x}0}$ which says $0 - v'(\overrightarrow{x}) \triangleleft c$, or seen differently, $-c \triangleleft v'(\overrightarrow{x})$.

Finally, for $\dagger_5$, if $Z_{x0} = (\leq, -\infty)$ the condition is trivially true. If $Z_{x0} \in \{(<, \infty), (\leq, \infty)\}$ then x is a prophecy clock and $\dagger_5$ follows from $\dagger_1, \dagger_2$. Therefore, we assume $Z_{x0} = (\triangleleft, c)$ for $c \in \mathbb{Z}$. The left hand side of the condition is $Z_{x0} + (<, -M) = (<, c - M)$, with $c - M \in \mathbb{Z}$. Let $Z_{x\overrightarrow{y}} = (\triangleleft', e)$ with $e \in \mathbb{Z} \cup \{-\infty, +\infty\}$. To show $\dagger_5$ it then suffices to show $c - M \leq e$. This involves more arguments in the same spirit as in $\dagger_2$ case above, and we leave these technical details to the full version [2]. ◀

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

Next, for valuations $v_1, v_2 \in \mathbb{V}$, we define $v_1 \sim_M v_2$ by two conditions: $v_1(x) \sim_M v_2(x)$ and $v_1(x) - v_1(y) \sim_{2M} v_2(x) - v_2(y)$ for all clocks $x, y \in X$. Notice that we use $2M$ for differences of values. Clearly, $\sim_M$ is an equivalence relation of finite index on valuations.

The next result relates the equivalence relation $\sim_M$ and the simulation relation $\preceq_G$ when the finite constants used in the constraints are bounded by M. Recall from Section 5 the definition of the distance graph $\mathbb{G}_G(v)$ for the set of valuations $\uparrow_G v$.

▶ **Lemma 27.** *Let $v_1, v_2 \in \mathbb{V}$ be valuations with $v_1 \sim_M v_2$ and let G be a set of atomic constraints using constants in $\{-\infty, \infty\} \cup \{c \in \mathbb{Z} \mid |c| \leq M\}$. By replacing the weights $(\leq, v_1(x))$ (resp. $(\leq, -v_1(x))$) by $(\leq, v_2(x))$ (resp. $(\leq, -v_2(x))$) in the graph $\mathbb{G}_G(v_1)$ we obtain the graph $\mathbb{G}_G(v_2)$.*

Next we state the central lemma that says that $\downarrow_G Z$ is a union of $\sim_M$ equivalence classes.

▶ **Lemma 28.** *Let $v_1, v_2 \in \mathbb{V}$ be valuations with $v_1 \sim_M v_2$ and let G be a set of atomic constraints using constants in $\{-\infty, \infty\} \cup \{c \in \mathbb{Z} \mid |c| \leq M\}$. Let Z be a zone with a canonical distance graph $\mathbb{G}(Z)$ satisfying (†). Then, $v_1 \in \downarrow_G Z$ iff $v_2 \in \downarrow_G Z$.*

Proof sketch. We will consider two valuations v_1, v_2 such that $v_1 \sim_M v_2$ and $v_1 \in \downarrow_G Z$ and show that the assumption that $v_2 \notin \downarrow_G Z$ leads to a contradiction. Roughly the proof proceeds as follows. Firstly, $v_2 \notin \downarrow_G Z$ implies that $\uparrow_G v_2 \cap Z = \emptyset$. Further, recall from Lemma 22 that if $\uparrow_G v_2 \cap Z = \emptyset$, then we can find a negative cycle C_2 using one edge from $\mathbb{G}(Z)$ and one or two edges from $\mathbb{G}_G(v_2)$. From Lemma 27, there exists a cycle C_1 involving the corresponding edges from $\mathbb{G}(Z)$ and $\mathbb{G}_G(v_1)$. Since $\uparrow_G v_1 \cap Z \neq \emptyset$, we know that C_1 is not negative. We will show that this implies that the C_2 (which was a witness for emptiness of $\uparrow_G v_2 \cap Z$) also cannot be negative, which leads to a contradiction. The central part of the proof involves a careful case analysis of the various forms that the cycle C_2 can take, using different † conditions. We detail two cases here. The remaining eight can be found in the full version [2].

- Cycle $C_2 = 0 \xrightarrow{(\leq, v_2(\overrightarrow{x}))} \overrightarrow{x} \xrightarrow{Z_{\overrightarrow{x}0}} 0$. We have $C_1 = 0 \xrightarrow{(\leq, v_1(\overrightarrow{x}))} \overrightarrow{x} \xrightarrow{Z_{\overrightarrow{x}0}} 0$.
  Let $Z_{\overrightarrow{x}0} = (\triangleleft, c)$. Since C_2 is negative, we deduce that $c \neq \infty$. From (†$_3$), we infer $Z_{\overrightarrow{x}0} \leq (\leq, M)$ and $0 \leq c \leq M$.
  Since C_1 is not negative, we get $(\leq, 0) \leq (\triangleleft, c + v_1(\overrightarrow{x}))$, which is equivalent to $-c \leq v_1(\overrightarrow{x})$. Using $v_1 \sim_M v_2$ and $0 \leq c \leq M$ we deduce that $-c \leq v_2(\overrightarrow{x})$. This is equivalent to $(\leq, 0) \leq (\triangleleft, c + v_2(\overrightarrow{x}))$, a contradiction with C_2 being a negative cycle.
- Cycle $C_2 = 0 \xrightarrow{Z_{0\overrightarrow{x}}} \overrightarrow{x} \xrightarrow{(\leq, -v_2(\overrightarrow{x}))} 0$. We have $C_1 = 0 \xrightarrow{Z_{0\overrightarrow{x}}} \overrightarrow{x} \xrightarrow{(\leq, -v_1(\overrightarrow{x}))} 0$.
  Let $Z_{0\overrightarrow{x}} = (\triangleleft, c)$. Since C_2 is negative, we deduce that $-v_2(\overrightarrow{x}) \neq \infty$. Using $v_1 \sim_M v_2$, we infer $-v_1(\overrightarrow{x}) \neq \infty$. Since C_1 is not negative, we get $Z_{0\overrightarrow{x}} \neq (\leq, -\infty)$. From (†$_6$), we infer $(<, -M) \leq Z_{0\overrightarrow{x}}$ and $-M \leq c \leq 0$.
  Since C_1 is not a negative cycle, we get $(\leq, 0) \leq (\triangleleft, c - v_1(\overrightarrow{x}))$, which is equivalent to $v_1(\overrightarrow{x}) \leq c$. Using $v_1 \sim_M v_2$ and $-M \leq c \leq 0$, we deduce that $v_2(\overrightarrow{x}) \leq c$. This is equivalent to $(\leq, 0) \leq (\triangleleft, c - v_2(\overrightarrow{x}))$, a contradiction with C_2 being a negative cycle. ◀

Finally, from Lemmas 26 and 28, we obtain our main theorem of the section.

▶ **Theorem 29.** *The simulation relation $\preceq_{\mathcal{A}}$ is finite.*

Proof. Let $(q, Z_0), (q, Z_1), (q, Z_2), \ldots$ be an infinite sequence of *reachable* nodes in $\mathsf{EZG}(\mathcal{A})$. By Lemma 26, for all i, the distance graph $\mathbb{G}(Z_i)$ in canonical form satisfies conditions (†).

The atomic constraints in $G = \mathcal{G}(q)$ use constants in $\{-\infty, \infty\} \cup \{c \in \mathbb{Z} \mid |c| \leq M\}$. From Lemma 28 we deduce that for all i, $\downarrow_G Z_i$ is a union of $\sim_M$-classes. Since $\sim_M$ is of finite index, there are only finitely many unions of $\sim_M$-classes. Therefore, we find $i < j$ with $\downarrow_G Z_i = \downarrow_G Z_j$, which implies $Z_j \preceq_G Z_i$. ◀

Note that the number of enumerated zones is bounded by 2^r, where r is the number of regions. This is similar to the exponential blow up that happens in normal timed automata. Indeed, despite this blow up the interest in zone algorithms is that, at least in the timed setting, they work significantly better in practice. We hope the above zone-based approach for ECA will also pave the way for fast implementations for ECA.

7 Conclusion

In this paper, we propose a simulation based approach for reachability in ECAs. The main difficulty and difference from timed automata is the use of prophecy clocks and undefined values. We believe that the crux of our work has been in identifying the new representation for prophecy clocks and undefined values. With this as the starting point, we have been able to adapt the zone graph computation and the $\mathcal{G}$-simulation technique to the ECA setting. This process required us to closely study the mechanics of prophecy clocks in the zone computations and we discovered this surprising property that prophecy clocks by themselves do not create a problem for finiteness.

The final reachability algorithm looks almost identical to the timed automata counterpart and hence provides a mechanism to transfer timed automata technology to the ECA setting. The performance benefits observed for the LU and $\mathcal{G}$-simulation-based reachability procedures for timed automata encourages us to believe that an implementation of our algorithm would also yield good results, thereby providing a way to efficiently check event-clock specifications on timed automata models. We also hope that our framework can be extended to other verification problems, like liveness and to extended models like ECA with diagonal constraints that have been studied in the context of timeline based planning [11, 12].

References

1 S. Akshay, Benedikt Bollig, and Paul Gastin. Event clock message passing automata: a logical characterization and an emptiness checking algorithm. *Formal Methods Syst. Des.*, 42(3):262–300, 2013.

2 S. Akshay, Paul Gastin, R. Govind, and B. Srivathsan. Simulations for event-clock automata. *CoRR*, abs/2207.02633, 2022.

3 S. Akshay, Paul Gastin, and Karthik R. Prakash. Fast zone-based algorithms for reachability in pushdown timed automata. In *CAV (1)*, volume 12759 of *Lecture Notes in Computer Science*, pages 619–642. Springer, 2021.

4 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

5 Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211(1-2):253–273, 1999.

6 Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Hakansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. UPPAAL 4.0. In *QEST*, pages 125–126. IEEE Computer Society, 2006.

7 Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *ACPN 2003*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003.

8 Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods Syst. Des.*, 24(3):281–320, 2004.

9 Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *CAV (1)*, volume 9779 of *Lecture Notes in Computer Science*, pages 513–530. Springer, 2016.

10 Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, 1998.

11 Laura Bozzelli, Angelo Montanari, and Adriano Peron. Taming the complexity of timeline-based planning over dense temporal domains. In *FSTTCS*, volume 150 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

12 Laura Bozzelli, Angelo Montanari, and Adriano Peron. Complexity issues for timeline-based planning over dense time under future and minimal semantics. *Theor. Comput. Sci.*, 901:87–113, 2022.

13 Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In *TACAS*, volume 1384 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 1998.

14 David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

15 Deepak D'Souza and Nicolas Tabareau. On timed automata with input-determined guards. In *FORMATS/FTRTFT*, volume 3253 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2004.

16 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in timed automata with diagonal constraints. In *CONCUR*, volume 118 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

17 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Fast algorithms for handling diagonal constraints in timed automata. In *CAV (1)*, volume 11561 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2019.

18 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability for updatable timed automata made faster and more effective. In *FSTTCS*, volume 182 of *LIPIcs*, pages 47:1–47:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

19 Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. Event clock automata: From theory to practice. In *FORMATS*, volume 6919 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2011.

20 Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. On regions and zones for event-clock automata. *Formal Methods Syst. Des.*, 45(3):330–380, 2014.

21 Frédéric Herbreteau and Gerald Point. TChecker. `https://github.com/fredher/tchecker`, v0.2 – April 2019.

22 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In *LICS*, pages 375–384. IEEE Computer Society, 2012.

23 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Lazy abstractions for timed automata. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 990–1005. Springer, 2013.

24 Gijs Kant, Alfons Laarman, Jeroen Meijer, Jaco van de Pol, Stefan Blom, and Tom van Dijk. LTSmin: High-performance language-independent model checking. In *TACAS*, volume 9035 of *Lecture Notes in Computer Science*, pages 692–707. Springer, 2015.

25 Sebastian Kupferschmid, Martin Wehrle, Bernhard Nebel, and Andreas Podelski. Faster than UPPAAL? In *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 552–555. Springer, 2008.

26 Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *STTT*, 1(1-2):134–152, 1997.

27 Jean-François Raskin and Pierre-Yves Schobbens. The logic of event clocks – decidability, complexity and expressiveness. *J. Autom. Lang. Comb.*, 4(3):247–282, 1999.

28 Victor Roussanaly, Ocan Sankur, and Nicolas Markey. Abstraction refinement algorithms for timed automata. In *CAV (1)*, volume 11561 of *Lecture Notes in Computer Science*, pages 22–40. Springer, 2019.

29 Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. PAT: towards flexible verification under fairness. In *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 709–714. Springer, 2009.

30 Tamás Tóth, Ákos Hajdu, András Vörös, Zoltán Micskei, and István Majzik. Theta: A framework for abstraction refinement-based model checking. In *FMCAD*, pages 176–179. IEEE, 2017.

31 Farn Wang. REDLIB for the formal verification of embedded systems. In *ISoLA*, pages 341–346. IEEE Computer Society, 2006.

32 Jianhua Zhao, Xuandong Li, and Guoliang Zheng. A quadratic-time DBM-based successor algorithm for checking timed automata. *Inf. Process. Lett.*, 96(3):101–105, 2005.

A Appendix for Section 3

In Figure 3, we give the event zone graph of the event-clock automaton $\mathcal{A}_1$ that recognizes the language $\{b^n a \mid n \geq 1\}$ such that there exists some b which occurs exactly one time unit before a.

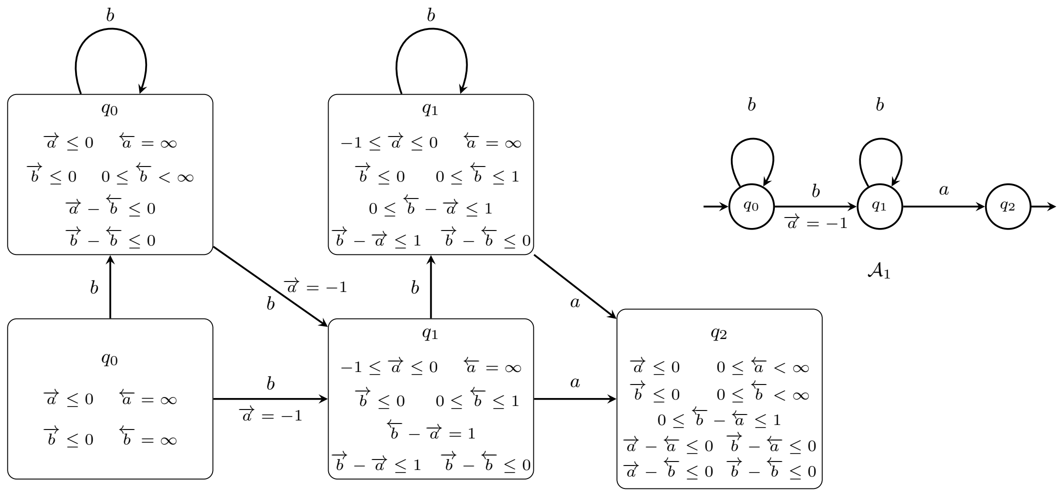


Figure 3 An event-clock automaton and its event zone graph. Missing lower bounds are of the form $-\infty \leq x - y$ and missing upper bounds are of the form $x - y \leq \infty$ (including $y = 0$).

Further, Geeraerts et al. [19, 20] showed that there exists no finite time abstract bisimulation relation for the event predicting automaton (EPA) $\mathcal{A}_2$ given in Figure 4. Figure 4 also depicts the event zone graph of $\mathcal{A}_2$. Note that, since this is an event *predicting* automaton, there are no history clocks. It is easy to see that there are only finitely many distinct constraints involving the prophecy clocks.

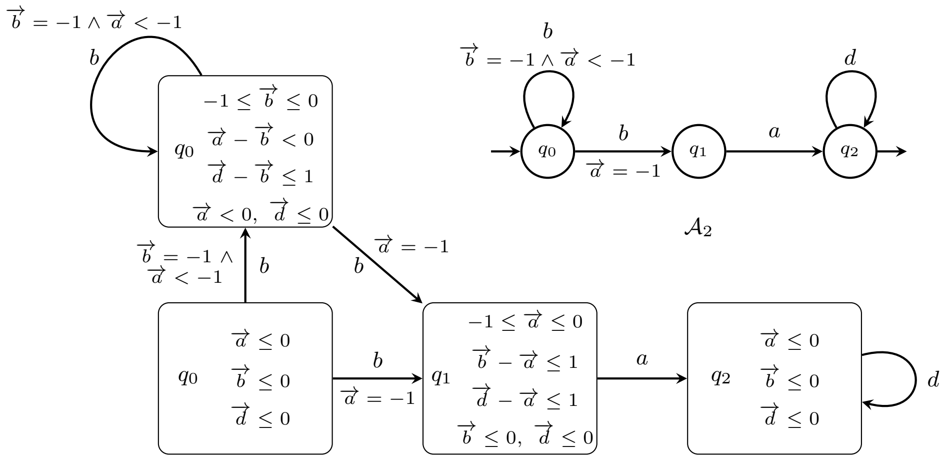


Figure 4 Event *predicting* automaton for which there exists no finite time abstract bisimulation and its event zone graph. Missing lower bounds are of the form $-\infty \leq x - y$ and missing upper bounds are of the form $x - y \leq \infty$ (including $y = 0$).