Form Methods Syst Des (2014) 45:330–380
DOI 10.1007/s10703-014-0212-1

On regions and zones for event-clock automata

Gilles Geeraerts · Jean-François Raskin · Nathalie Sznajder

Published online: 6 August 2014

Abstract *Event clock automata* (ECA) are a model for *timed languages* that has been introduced by Alur, Fix and Henzinger as an alternative to *timed automata*, with better theoretical properties (for instance, ECA are determinizable while timed automata are not). In this paper, we *revisit* and *extend* the theory of ECA. We first prove that *no finite time abstract language equivalence* exists for ECA, thereby disproving a claim in the original work on ECA. This means in particular that regions *do not form a time abstract bisimulation*. Nevertheless, we show that regions can still be used to build a finite automaton recognizing the *untimed language of an* ECA. Then, we extend the classical notions of *zones* and *DBMs* to let them handle event clocks instead of plain clocks (as in timed automata) by introducing *event zones* and *Event DBMs* (EDBMs). We discuss algorithms to handle event zones represented as EDBMs, as well as (semi-) algorithms based on EDBMs to decide language emptiness of ECA.

Keywords Event clock automata · Timed automata · Real-time · Verification · Formal methods · Zones · regions · DBMs

1 Introduction

Timed automata have been introduced by Alur and Dill in the early nineties [3] and are a successful and popular model to reason about *timed behaviors* of computer systems. Where finite automata represent behaviors by finite sequences of *letters* (taken from a finite alphabet,

G. Geeraerts (✉) · J.-F. Raskin
Faculté des Sciences, Département d'Informatique, Université libre de Bruxelles, Brussels, Belgium
e-mail: gigeerae@ulb.ac.be

J.-F. Raskin
e-mail: jraskin@ulb.ac.be

N. Sznajder
Sorbonne Universités, Université Pierre et Marie Curie UMR CNRS 7606, LIP6, Paris, France
e-mail: nathalie.sznajder@lip6.fr

each letter models an action of the system), timed automata define sets of *timed words* (called *timed languages*) that are finite sequences of letters, each paired with a real time stamp. To this end, timed automata extend finite automata with a finite set of real valued clocks, that can be tested and reset with each action of the system. The theory of timed automata is now well developed [1]. Efficient algorithms to analyse timed automata, that rely on peculiar data structures such as zones and DBMs have been proposed [18,24]. These algorithms have been implemented in several tools such as Kronos [12] or UppAal [5]. Those tools have been successfully applied in several industrial case studies (for instance [7,22], see http://www.it.uu.se/research/group/darts/uppaal/examples.shtml for a comprehensive list).

The model of timed automata, however, suffers from certain weaknesses, at least from the theoretical point of view: timed automata are *not determinizable* and *cannot be complemented* in general [3]. Intuitively, this stems from the fact that the reset of the clocks cannot be made deterministic wrt the word being read. Indeed, from a given location, there can be several transitions, labeled by the same letter a, and with compatible guards, but resetting different clocks.

This observation has prompted Alur, Fix and Henzinger to introduce the model of *event clock automata* (ECA for short) [4], as an alternative formalism for representing timed languages. Unlike timed automata, ECA force the clock resets to be strongly linked to the letters that label the transitions. To achieve this, there are, in an ECA, and for each letter a, two clocks $\overleftarrow{x_a}$ and $\overrightarrow{x_a}$. The clock $\overleftarrow{x_a}$ is the *history clock* of a and *always records the time elapsed since the last occurrence of* a. Symmetrically, $\overrightarrow{x_a}$ is the *prophecy clock* for a, and *always predicts the time distance up to the next occurrence of* a. Thus, the value of any history clock *increases* with time elapsing (like clocks in timed automata do), while the values of prophecy clocks *decrease over time*. The main advantage of this definition is that the value of all clocks is uniquely determined at any point in the timed word being read, no matter what path is being followed in the ECA A nice consequence of this definition is that ECA are *determinizable* [4], unlike timed automata. However, this comes at a price, as the expressiveness of ECA is strictly weaker than that of timed automata [3].

Nevertheless, ECA remain an appealing model, and we believe that they could be of practical interest in the modeling and verification of timed systems. While the theory of ECA has witnessed some developments [16,19,20,25,27] since the seminal paper, no tool is available that exploits the full power of event clocks (the only tool we are aware of is TEMPO [26] and it is restricted to *event-recording automata*, i.e. ECA with history clocks only).

In this work, we revisit and extend the theory of ECA, with the hope to make it more practical and amenable to implementation. A widespread belief [4] about ECA is that they are similar enough to timed automata that the classical techniques (such as regions, zones or DBMs) developed for them can readily be applied to ECA. The present research, however, highlights *fundamental discrepancies* between timed automata and ECA:

(1) First, we show that *there is no finite time abstract language equivalence* on the valuations of *event clocks*, whereas the region equivalence [3] *is* a finite time abstract language equivalence for timed automata. This implies, in particular, that *regions do not form a finite time-abstract bisimulation for* ECA, thereby contradicting a claim found in the original paper on ECA[4].

(2) With timed automata, checking language emptiness can be done by building the so-called region automaton [3] which recognizes $\mathsf{Untime}(\mathsf{L}(A))$, the untimed version of A's timed language. A consequence of the surprising result of point 1 is that, for some ECAA, the *region automaton recognizes a strict subset of* $\mathsf{Untime}(\mathsf{L}(A))$. Thus, the

region automaton (as defined in [3]) *is not a sound construction for checking language emptiness of* ECA.

(3) To recover a finite automaton recognising $\mathsf{Untime}(\mathsf{L}(A))$, we introduce a *novel semantics for* ECA, the so-called *weak semantics* which is obtained by slightly relaxing the definition of the elapsing of time on prophecy clocks that are above the maximal constant *cmax* appearing in the automaton. We show that this does not impact the recognised language of the automaton, i.e. the *weak semantics recognises the same language as the classical one*, but with the benefit that *regions are finite time-abstract bisimulation* on the weak semantics. Equipped with this theoretical tool, we show that the *existential region automaton*, which is a slight modification of the original definition of region automaton (on the classical semantics) allows to recover $\mathsf{Untime}(\mathsf{L}(A))$.

(4) Efficient algorithms to analyze timed automata are best implemented using *zones* [1], that are in turn represented by *DBMs* [18]. Unfortunately, zones and DBMs cannot be directly applied to ECA. Indeed, a zone is, roughly speaking, a conjunction of constraints of the form $x - y \prec c$, where x, y are clocks, $\prec$ is either $<$ or $\leq$ and c is an integer. This makes sense in the case of timed automata, since the difference of two clock values is an invariant with time elapsing. This is not the case when we consider event clocks, as *prophecy and history clocks evolve in opposite directions with time elapsing*. Thus, we introduce the notions of event-zones and Event DBMs that can handle constraints of the form $x + y \prec c$, when x and y are of different types.

(5) In the case of timed automata two basic, zone-based algorithms for solving language emptiness have been studied: the *forward analysis algorithm* that iteratively computes all the states reachable from the initial state, and the *backward analysis algorithm* that computes all the states that can reach a target state. While the former might not terminate in general, the latter is guaranteed to terminate [1]. We show that this is not the case anymore with ECA: *both algorithms might not terminate* again because of event clocks evolving in opposite directions.

(6) Still in the case of timed automata, *widening operators* have been proposed that overcome this issue and guarantee the termination of the *forward algorithm*. The most popular widening operators is certainly the so-called 'k-approximation' [14]. This operator applies to zones, and consists, roughly speaking, in replacing every constraint of the form $x - y \prec c$ by $x - y \prec +\infty$, and every constraint of the form $c \prec x - y$ by $k \prec x - y$, when $c > k$. Usually, this operators is used by setting $k = cmax$, i.e., the largest constant appearing in the guards of the automaton. Intuitively, this widening seems to make sense, as the automaton cannot distinguish between two clock values that are $> cmax$. Nevertheless, the correctness of this operator has sparkled much debate in the community of real-time systems recently. Bouyer has finally settled the question by showing, that, *cmax*-approximation is *correct* when the timed automaton has no *diagonal constraints* in the guards, but that it is an *over-approximation* when diagonal constraints are allowed [9]. This means that there are some timed automata with diagonal constraints on which the *forward algorithm*, together with the *cmax*-approximation widening operator, will compute a *strict over-approximation* of the reachable states.

In the present work, we show that, contrary to timed automata case, this operator is *not correct* for ECA, *even when no diagonal constraints are allowed*. More precisely, applying either the forward or the backward algorithm to ECA, together with the *cmax*-approximation widening operator (as defined for timed automata), may yield a strict over-approximation of the set of reachable states. This observation thus accounts for an additional difference between timed automata and ECA, and show once again that the analysis techniques for timed automata cannot be straightforwardly applied to ECA.

Nevertheless, we present an alternative version of the k-approximation operator (actually, a slight relaxation of the original operator), and prove that it is *correct* in the case of ECA, using proof techniques similar to Bouyer's [9]. To the best of our knowledge, this yields the first forward and backward *zone-based* algorithms for ECA that are *both sound and complete*.

These observations highlight the structure of the paper: after some preliminaries in Sect. 2, where we recall the model of ECA and other technical matters, we present several equivalence relations for event-clocks valuations (Sect. 3). Then, we discuss regions and region automata in the setting of event-clocks, in Sect. 5. In Sect. 6, we introduce event zones and event-DBMs, which are adaptations of classical clock zones and DBMs to event-clocks. In Sect. 7, we discuss the classical forward and backward algorithms, when applied to ECA, as well as the associated widening operators.

This work is an extended and revised version of a conference paper appeared in FORMATS 2011 [21].

The authors would like to express their gratitude towards the anonymous reviewers. Their numerous comments and suggestions have significantly helped to improve the present paper.

2 Preliminaries

Let us start by introducing the basic notions that will be used in the present work. We first discuss them informally, using a running example.

Assume we want to model a log-in procedure. To log in to the system, the user must first enter his user name (action u of the system). Then, he must type his password (action p) and gets logged in (action l) if the password is correct, or has to retype his password (action r) if the previous attempt failed. This can be repeated several times. In order to avoid dictionary attacks on this log in procedure, additional rules are enforced. First, in case of a wrong password, the user has to wait at least 1 t.u. before being allowed to try a new password. Second, the whole log in procedure has to be completed within 5 t.u. after the user has typed his user name. If the user has not been able to provide a matching password for the user name within 5 t.u., the log in procedure is reset and starts anew (action t, for 'time out').

Obviously, a faithful model of an execution of this procedure has to encode the (exact) time stamps of the events, in addition to the sequence of actions performed during the execution. As an example, a possible execution of the protocol is that the user types his user name at time 1.3, and types a wrong password at time 3.4. Then, the system waits one time unit and offers the user to retype his password at time 4.4. The user types again a wrong password at time 6.2, and the log in procedure is reset at time 6.3, i.e., exactly 5 time units after the user has provided his user name. Then, the user completes the log in procedure, by typing his user name at time 7.5, his password at time 8.3, and gets logged in at time 8.4. This execution can be represented by a *timed word*, which is a finite sequence of pairs (action, time stamp):

$$(\mathtt{u}, 1.3)\ (\mathtt{p}, 3.4)\ (\mathtt{r}, 4.4)\ (\mathtt{p}, 6.2)\ (\mathtt{t}, 6.3)\ (\mathtt{u}, 7.5)\ (\mathtt{p}, 8.3)\ (\mathtt{l}, 8.4)$$

Then, the set of *all possible executions* of the procedure can be modeled as a (possibly infinite) set of timed words, i.e. a *timed language*. To formally define timed languages in a finite way, one usually relies on some kind of automaton model. A popular automaton model for timed languages is that of *timed automata* [1], which extends finite automata by means of *clocks*, i.e. real-valued variables whose values evolve with time elapsing, and that can be tested and reset at will during the execution of the automaton.

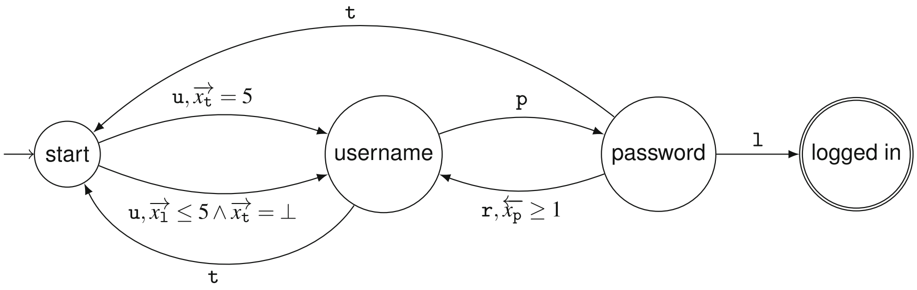

start
username
password
logged in
t
$\mathtt{u}, \overrightarrow{x_{\mathtt{t}}} = 5$
p
l
$\mathtt{u}, \overrightarrow{x_{\mathtt{l}}} \leq 5 \wedge \overrightarrow{x_{\mathtt{t}}} = \bot$
$\mathtt{r}, \overleftarrow{x_{\mathtt{p}}} \geq 1$
t

Fig. 1 A login protocol modeled as an event-clock automaton on the alphabet $\Sigma = \{\mathtt{l}, \mathtt{p}, \mathtt{r}, \mathtt{t}, \mathtt{u}\}$

Instead, this work considers the model of *event-clock automata* (ECA for short), introduced in [4]. The most salient difference between timed automata and ECA is that clocks in ECA are called *event clocks*, because their values are tightly linked to the events that occur (i.e., the letters making up the timed word). In and ECA, there are at most two event clocks associated with each each letter a: the *history clocks* $\overleftarrow{x_a}$ that records the time elapsed since the last occurrence of a (or contains $\bot$ if no such event ever occurred), and the *prophecy clock* $\overrightarrow{x_a}$ which predicts the time up to the next occurrence of a (or contains $\bot$ if no more a will occur). In our example execution, at time 2.1 (after the beginning of the execution), the value of $\overleftarrow{x_{\mathtt{u}}}$ is thus 0.8 since, at that time, the last u has occurred at time stamp 1.3; $\overrightarrow{x_{\mathtt{r}}}$ has value 2.3; and $\overleftarrow{x_{\mathtt{p}}}$ has value $\bot$; for instance. Thus, in an ECA, the value of the clocks is uniquely determined by the (complete) word being read, contrary to timed automata, where the same prefix can be read by following two different paths in the automata that have different effects on the clocks.

Then, our log in procedure can be modeled by the ECA given in Fig. 1. As can be seen on the figure, an ECA is a *finite automaton* whose transitions can be labeled by a *condition* or *guard* on the *event clocks*, in addition to the action corresponding to the transition. In Fig. 1, the ECA has four states, labeled 'start', 'username', 'password' and 'logged in', that correspond respectively to the beginning of the login protocol, the fact that the user has typed his username, the fact that the user has typed his password, and the completion of the login procedure. Each execution of the log in procedure corresponds to a *run* of the automaton. Such a run is a path in the automaton that starts in an initial location[1] (in our case, the 'start' location), ends in an accepting location[2] (in our case, the 'logged in' location), and where all guards on transitions taken along the path are satisfied. *Guards* on transitions are conditions on the valuations of the clocks.

For instance, a run of the ECA in Fig. 1, that corresponds to the timed word given above, is as follows. First, the run starts in the initial state 'start'. Then it moves to 'user name' after 1.3 time units, taking the upper transition on the figure. This transition imposes, thanks to its guard $\overrightarrow{x_{\mathtt{t}}} = 5$ that the next t action will occur exactly 5 t.u. after its firing, i.e., at time stamp $5 + 1.3 = 6.3$. Then, it takes, after 2.1 t.u. the p-labeled transition to 'password' and the r-labeled transition back to 'user name', after an additional 1 t.u. Remark that the delay between the p- and the r-labeled transition had to be ≥ 1 because of the $\overleftarrow{x_{\mathtt{p}}} \geq 1$ guard on the r-labeled transition. After the time-out (t-labeled transition), the run reaches the 'start' location again, and the u-labeled transition bearing guard $\overrightarrow{x_{\mathtt{l}}} \leq 5 \wedge \overrightarrow{x_{\mathtt{t}}} = \bot$ is fired to reach location 'username'. Here, the constraint $\overrightarrow{x_{\mathtt{t}}} = \bot$ reads 'the t action will

[1] Initial locations are denoted by a small arrow pointing to the node.

[2] Accepting locations are denoted by a doubled border.

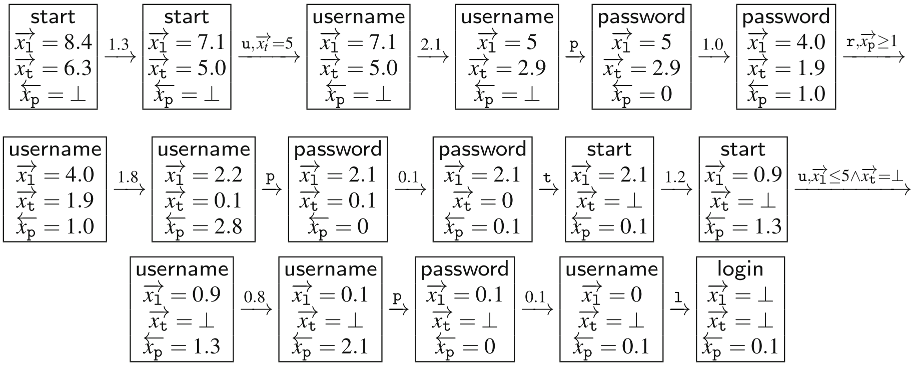

start
$\overrightarrow{x_1}=8.4$
$\overrightarrow{x_t}=6.3$
$\overleftarrow{x_p}=\bot$
1.3
start
$\overrightarrow{x_1}=7.1$
$\overrightarrow{x_t}=5.0$
$\overleftarrow{x_p}=\bot$
u,$\overrightarrow{x_l}=5$
username
$\overrightarrow{x_1}=7.1$
$\overrightarrow{x_t}=5.0$
$\overleftarrow{x_p}=\bot$
2.1
username
$\overrightarrow{x_1}=5$
$\overrightarrow{x_t}=2.9$
$\overleftarrow{x_p}=\bot$
p
password
$\overrightarrow{x_1}=5$
$\overrightarrow{x_t}=2.9$
$\overleftarrow{x_p}=0$
1.0
password
$\overrightarrow{x_1}=4.0$
$\overrightarrow{x_t}=1.9$
$\overleftarrow{x_p}=1.0$
r,$\overleftarrow{x_p}\geq 1$
username
$\overrightarrow{x_1}=4.0$
$\overrightarrow{x_t}=1.9$
$\overleftarrow{x_p}=1.0$
1.8
username
$\overrightarrow{x_1}=2.2$
$\overrightarrow{x_t}=0.1$
$\overleftarrow{x_p}=2.8$
p
password
$\overrightarrow{x_1}=2.1$
$\overrightarrow{x_t}=0.1$
$\overleftarrow{x_p}=0$
0.1
password
$\overrightarrow{x_1}=2.1$
$\overrightarrow{x_t}=0$
$\overleftarrow{x_p}=0.1$
t
start
$\overrightarrow{x_1}=2.1$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=0.1$
1.2
start
$\overrightarrow{x_1}=0.9$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=1.3$
u,$\overrightarrow{x_1}\leq 5\wedge\overrightarrow{x_t}=\bot$
username
$\overrightarrow{x_1}=0.9$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=1.3$
0.8
username
$\overrightarrow{x_1}=0.1$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=2.1$
p
password
$\overrightarrow{x_1}=0.1$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=0$
0.1
username
$\overrightarrow{x_1}=0$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=0.1$
l
login
$\overrightarrow{x_1}=\bot$
$\overrightarrow{x_t}=\bot$
$\overleftarrow{x_p}=0.1$

Fig. 2 A run of the ECA in Fig. 1 on $(\mathtt{u}, 1.3)(\mathtt{p}, 3.4)(\mathtt{r}, 4.4)(\mathtt{p}, 6.2)(\mathtt{t}, 6.3)(\mathtt{u}, 7.5)(\mathtt{p}, 8.3)(\mathtt{l}, 8.4)$

never happen again'. The run ends by taking the p labeled transition after 0.8 t.u., then the l labeled transition after 0.1 t.u., to reach the 'logged in' location. This run is depicted in Fig. 2, where each box represents an *extended state* of the ECA, i.e., a pair (q, v), where q is a location (displayed on top of the box), and v is a valuation of the clocks. An arrow labeled by a real number represents the elapsing of time, and the other arrows indicate the firing of the corresponding transitions. Hence, in a run of an ECA on a given timed word, the transition taken when reading some letter might depend on the rest of the word.

As our example shows, ECA are expressive enough to model interesting and realistic computer systems with real-time constraints. Other examples can be found in [4] and [26] for instance. From a theoretical point of view, the class of languages that can recognised by ECA is a strict subclass of the languages recognised by timed automata, and contains strictly all languages recognised by *deterministic* timed automata [4].

2.1 Basic notions

Words and timed words An alphabet Σ is a finite set of symbols. A (finite) *word* is a finite sequence $w = w_0 w_1 \cdots w_n$ of elements of Σ. We denote the length of w by $|w|$. We denote by Σ^* the set of finite words over Σ. A (finite) *timed word* over Σ is a pair $\theta = (\tau, w)$ such that w is a word over Σ and $\tau = \tau_0\tau_1\cdots\tau_{|w|-1}$ is a word over $\mathbb{R}^{\geq 0}$ with $\tau_{i-1} \leq \tau_i$ for all $1 \leq i \leq |w| - 1$. We denote by $\mathsf{T}\Sigma^*$ the set of finite timed words over Σ. A timed *language* is a subset of $\mathsf{T}\Sigma^*$. For a timed word $\theta = (\tau, w)$, we let $\mathsf{Untime}(\theta) = w$. For a timed language L, we let $\mathsf{Untime}(L) = \{\mathsf{Untime}(\theta) \mid \theta \in L\}$.

Event clocks Given an alphabet Σ, we define the set of associated *event clocks* $\mathbb{C}_\Sigma = \mathbb{H}_\Sigma \cup \mathbb{P}_\Sigma$, where $\mathbb{H}_\Sigma = \{\overleftarrow{x_a} \mid a \in \Sigma\}$ is the set of *history clocks*, and $\mathbb{P}_\Sigma = \{\overrightarrow{x_a} \mid a \in \Sigma\}$ is the set of *prophecy clocks*. Let $C \subseteq \mathbb{C}_\Sigma$ be a set of event clocks. A *valuation* of the clocks in C is a function $v : C \to \mathbb{R}^{\geq 0} \cup \{\bot\}$, where $\bot$ means that the clock value is undefined. We denote by $\mathscr{V}(C)$ the set of all valuations of the clocks in C. For a valuation $v \in \mathscr{V}(C)$, for all $x \in C$ such that $v(x) \neq \bot$, if $x \in \mathbb{H}_\Sigma$, we let $\langle v(x)\rangle = \lceil v(x)\rceil - v(x)$ and if $x \in \mathbb{P}_\Sigma$, we let $\langle v(x)\rangle = v(x) - \lfloor v(x)\rfloor$, where $\lfloor v(x)\rfloor$ and $\lceil v(x)\rceil$ denote respectively the floor and ceiling of $v(x)$. Intuitively, for all clocks x, $\langle v(x)\rangle$ is the remaining time before x crosses an integer value (provided it is not reset). We also denote by $v^\pm$ the valuation s.t. $v^\pm(x) = v(x)$ for all $x \in \mathbb{H}_\Sigma$, and $v^\pm(x) = -v(x)$ for all $x \in \mathbb{P}_\Sigma$.

For all valuations $v \in \mathscr{V}(C)$ and all $d \in \mathbb{R}^{\geq 0}$ such that $v(x) \geq d$ for all $x \in \mathbb{P}_\Sigma \cap C$, we define the valuation $v + d$ obtained from v by letting d time units elapse: for all $x \in \mathbb{H}_\Sigma \cap C$, $(v + d)(x) = v(x) + d$ and for all $x \in \mathbb{P}_\Sigma \cap C$, $(v + d)(x) = v(x) - d$, with the convention that $\bot + d = \bot - d = \bot$. A valuation is *initial* iff $v(x) = \bot$ for all $x \in \mathbb{H}_\Sigma$, and *final* iff $v(x) = \bot$ for all $x \in \mathbb{P}_\Sigma$. We note $v[x := c]$ the valuation that matches v on all its clocks except for $v(x)$ that equals c. We extend this notation to set of clocks X.

An *atomic constraint* over a set of variables X is either true or of the form $x \sim c$, where $x \in X$, $c \in \mathbb{N}$ and $\sim \in \{<, >, =\}$. An *atomic event clock constraint* on a set of event clocks C is either an atomic constraint on C, or a constraint of the form $x = \bot$, for some $x \in C$. A *constraint* over X is a Boolean combination of atomic constraints. An *event clock constraint* over a set of event clocks C is a Boolean combination of atomic event clock constraints.[3] We denote $\mathsf{Constr}(X)$ the set of all possible constraints over the set of variables X, and by $\mathsf{ECConstr}(C)$ the set of all possible *event clock* constraints over the set of event clocks C. A valuation $v : X \mapsto \mathbb{R}^{\geq 0} \cup \{\bot\}$ satisfies a constraint $\psi \in \mathsf{ECConstr}(X)$, denoted $v \models \psi$ according to the following rules: $v \models \mathsf{true}$, $v \models x \sim c$ iff $v(x) \neq \bot$ and $v(x) \sim c$, $v \models x = \bot$ iff $v(x) = \bot$, $v \models \neg\psi$ iff $v \not\models \psi$, and $v \models \psi_1 \wedge \psi_2$ iff $v \models \psi_1$ and $v \models \psi_2$.

2.2 Event-clock automata: syntax and semantics

We are now ready to recall the definition of event-clock automata. We start with the syntax, then give two semantic interpretations of ECA, the former in terms of finite word language, the latter in term of infinite words. Remark that, throughout this paper, we will focus mainly on *finite words languages* of ECA because our results can easily be adapted to the *infinite words* case.

Definition 1 ([4]) An *event-clock automaton* $A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$ (ECA for short) is a tuple, where:

(1) Q is a finite set of locations,
(2) $q_i \in Q$ is the initial location,
(3) Σ is an alphabet,
(4) $C \subseteq \mathbb{C}_\Sigma$ is a set of event clocks on Σ,
(5) $\delta \subseteq Q \times \Sigma \times \mathsf{ECConstr}(C) \times Q$ of edges,
(6) $\alpha \subseteq Q$ is the set of accepting locations

We also require that, for each $q \in Q$, $\sigma \in \Sigma$, δ is defined for a finite number of $\psi \in \mathsf{ECConstr}(C)$.

Observe that this standard definition of event-clock automata disallows silent transitions (i.e. ε-labeled transitions). Intuitively, this can be explained by the fact that such transitions would not modify the value of clocks anyway. Then, using a result by Diekert, Gastin and Petit on timed automata [17], one can easily show that ε-labeled transition can be removed in ECA too, without modifying the accepted language.

We distinguish two syntactic subclasses of ECA, corresponding to the cases where only history clocks, or only prophecy clocks are present. Formally, an ECA$A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$ is:

(1) an *event recording automaton* (ERA for short) iff $C \subseteq \mathbb{H}_\Sigma$;
(2) an *event predicting automaton* (EPA for short) iff $C \subseteq \mathbb{P}_\Sigma$.

[3] In the rest of the paper, we often use $x \geq c$ and $x \leq c$ as shorthands for $x > c \vee x = c$ and $x < c \vee x = c$ respectively.

Runs and accepted language The semantics of ECA is best described in terms of a timed transition system, i.e. an infinite transition system where the elapsing of time is made explicit.[4] An *extended state* (or simply state) of an $\mathsf{ECA}\, A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$ is a pair (q, v) where $q \in Q$ is a location, and $v \in \mathscr{V}(C)$ is a valuation. Then, the infinite (timed) transition system associated to A is defined as follows:

Definition 2 Let $A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$ be an ECA. Its *classical semantics* is the infinite timed transition system $\mathsf{TS}_A = \left\langle Q^A, Q_i^A, \rightarrow, \alpha^A \right\rangle$, where:

(1) $Q^A = Q \times \mathscr{V}(C)$ is the set of extended states of A,
(2) $Q_i^A = \{(q_i, v) \mid v \text{ is initial}\}$,
(3) $\alpha^A = \{(q, v) \mid q \in \alpha \text{ and } v \text{ is final}\}$, and
(4) the transition relation $\rightarrow \subseteq \left(Q^A \times \mathbb{R}^{\geq 0} \times Q^A\right) \cup \left(Q^A \times \Sigma \times Q^A\right)$ is s.t.:

(a) $\big((q, v), t, (q, v')\big) \in \rightarrow$ iff $v' = v + t$ (in particular, this implies that $v(x) \geq t$ for all prophecy clocks x), and

(b) $\big((q, v), a, (q', v')\big) \in \rightarrow$ iff there are $(q, a, \psi, q') \in \delta$ and $\overline{v} \in \mathscr{V}(C)$ s.t. $\overline{v}[\overrightarrow{x_a} := 0] = v$, $\overline{v}[\overleftarrow{x_a} := 0] = v'$ and $\overline{v} \models \psi$. Intuitively, $\overline{v}$ is the clock valuation obtained after letting time elapse, and where: (i) $\overleftarrow{x_a}$ still has its *old value*, in the sense that it contains the time elapsed since the last a (recall that $\overleftarrow{x_a}$ will be equal to 0 after the transition has fired) and; (ii) $\overrightarrow{x_a}$ already has its *new value*, i.e., the value predicting the time to the next a, that it will contain after the firing of the transition.

We use the notations $(q, v) \xrightarrow{t} (q, v')$ whenever $\big((q, v), t, (q, v')\big) \in \rightarrow$, $(q, v) \xrightarrow{a} (q', v')$ whenever $\big((q, v), a, (q', v')\big) \in \rightarrow$ and $(q, v) \xrightarrow{t,a} (q', v')$ whenever there is (q'', v'') s.t. $(q, v) \xrightarrow{t} (q'', v'') \xrightarrow{a} (q', v'')$. Intuitively, this semantics means that a history clock $\overleftarrow{x_a}$ always records the time elapsed since the last occurrence of the corresponding a event, and that a prophecy clock $\overrightarrow{x_a}$ always predicts the delay up to the next occurrence of a. Thus, when firing an a-labeled transition, the guard must be tested against $\overline{v}$ (as defined above) because it correctly predicts the next occurrence of a and correctly records its last occurrence (unlike v and v', as $v(\overrightarrow{x_a}) = 0$ and $v'(\overleftarrow{x_a}) = 0$).

Thanks to this definition of the transition relation, we are ready to define the notion of *run*:

Definition 3 Let $\mathsf{TS}_A = \left\langle Q^A, Q_i^A, \rightarrow, \alpha^A \right\rangle$ be the timed transitions system of some $\mathsf{ECA}A$. *A* (q, v)*-run of* TS_A *on the timed word* $\theta = (\tau, w)$ is a finite sequence $\rho = (q_0, v_0)(t_0, w_0)(q_1, v_1)(t_1, w_1)(q_2, v_2) \cdots (q_n, v_n)$ s.t.:

(1) $(q_0, v_0) = (q, v)$, $t_0 = \tau_0$,
(2) for any $1 \leq i \leq |w| - 1$: $t_i = \tau_i - \tau_{i-1}$, and
(3) for any $0 \leq i \leq |w| - 1$: $(q_i, v_i) \xrightarrow{t_i, w_i} (q_{i+1}, v_{i+1})$.

A (q, v)-run is *initialized* iff $(q, v) \in Q_i^A$ (in this case, we simply call it a run). Let us now explain how to interpret the set of accepting states of the ECA:

Definition 4 Let $\mathsf{TS}_A = \left\langle Q^A, Q_i^A, \rightarrow, \alpha^A \right\rangle$ be the timed transitions system of some $\mathsf{ECA}A$. A (q, v)-run, ending in (q_n, v_n) is *accepting* iff $(q_n, v_n) \in \alpha^A$.

[4] Remark that the term 'timed transition system' has been used with other meanings. In particular, in [4], timed transitions systems are actually a *model* that can be seen as a variant of timed automata.

Whenever ρ is an accepting run on θ, we say that ρ *accepts* θ. Thanks to these notions, we can now define the *language accepted* by a timed transition system $\mathsf{TS}_A = \langle Q^A, Q_i^A, \rightarrow, \alpha^A \rangle$, for some $\mathsf{ECA}\,A$.

Definition 5 $\mathsf{TS}_A = \langle Q^A, Q_i^A, \rightarrow, \alpha^A \rangle$ be the timed transitions system of some $\mathsf{ECA}\,A$. Then, the *language accepted by* TS_A *from* (q, v) is the set $\mathsf{L}(\mathsf{TS}_A, (q, v))$ of all timed words that are accepted by a (q, v)-run of TS_A. The *language accepted by* TS_A is the set $\mathsf{L}(\mathsf{TS}_A)$ of all timed words accepted by an initialized run of TS_A.

By abuse of notation, we often use the *language accepted by* A or *language accepted by* A *from* (q, v) to refer respectively to the language of TS_A and the language accepted by TS_A from (q, v). We often denote them by $\mathsf{L}(A)$ and $\mathsf{L}(A, (q, v))$ respectively. Recall that in Sect. 4 we will define an alternative semantics for ECA.

2.3 Reachability problem

In this paper, we focus mainly on the *reachability* problem that consists in determining whether a given set of target locations Q_{target} is reachable:

Problem 6 (*Reachability problem*) Given an $\mathsf{ECA}\,A$ and set of locations Q_{target} of A, the *reachability problem* asks whether there exists a finite (initialized) run $(q_0, v_0)(t_0, w_0) \cdots (q_n, v_n)$ of A with $q_n \in Q_{\text{target}}$.

We say that Q_{target} is *reachable* in A iff the answer to the reachability problem is 'yes' on that instance.

Two related problems are the *language emptiness problem* (does $\mathsf{L}(A) = \varnothing$?) and the *language inclusion problem* (does $\mathsf{L}(A) \subseteq \mathsf{L}(B)$?). The latter allows, for instance, to check that all executions of a system modeled as an $\mathsf{ECA}\,A$ are included in a set of *correct executions*, defined as an $\mathsf{ECA}\,B$. It is well-known that these problems can be reduced to the reachability problem. In particular, language inclusion boils down to asking whether $\mathsf{L}(A) \cap \overline{\mathsf{L}(B)} = \varnothing$. Since ECA are closed under intersection and complement [4] (due to the determinization procedure), one can compute an $\mathsf{ECA}\,C$ s.t. $\mathsf{L}(C) = \mathsf{L}(A) \cap \overline{\mathsf{L}(B)}$, and test for its language emptiness.

As stated in the introduction, ECA have been introduced as an alternative to timed automata, for the specification of timed languages. The original work on ECA [4] contains a thorough comparisons of the expressiveness of these two models. In particular, it is shown that each ECA can be turned into a *non-deterministic timed automaton* (TA for short, see Appendix 1 for a formal definition) that accepts the same language. For the sake of completeness, we recall the details of the construction in Appendix 1. It allows us to characterise precisely[5] the size of the timed automaton, a result that will be important later in the paper, to motivate our new version of the region automaton for ECA:

Theorem 7 ([4]) *For all* $\mathsf{ECA}\,A = \langle Q^A, q_i^A, \Sigma, C, \delta^A, \alpha^A \rangle$, *one can build a* TA $B = \langle Q^B, Q_i^B, \Sigma, X^B, \delta^B, \alpha^B \rangle$ *s.t.:*

- *both automata accept the same timed language:* $\mathsf{L}(A) = \mathsf{L}(B)$,
- $|Q^B| \leq |Q^A| \times 2^{(4 \times cmax + 6) \times |\Sigma|}$,
- $|X^B| \leq (4 \times cmax + 6) \times |\Sigma|$,
- *the maximal constant cmax is the same in both A and B.*

[5] Remark that, although the construction and Lemma 51 are given in the case where the ECA contains no punctual guards, Theorem 7 is still valid in the general case, as removing punctuality from the guards does not change the number of states nor the number of clocks of the ECA.

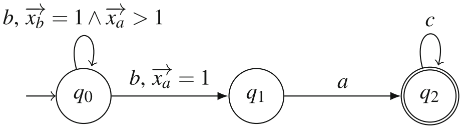

$b, \overrightarrow{x_b} = 1 \wedge \overrightarrow{x_a} > 1$
$b, \overrightarrow{x_a} = 1$
a
c
q_0
q_1
q_2

Fig. 3 The event-predicting automaton A_{inf}

3 Time-abstract equivalence relations for event-clocks

A classical technique to analyze timed transition systems is to define *time abstract equivalence relations* on the set of states of their underlying timed transition system, and to reason on the resulting *quotient* transition system. In the case of *timed automata*, a fundamental concept is the *region equivalence* [3], which is a *finite time-abstract* bisimulation, and allows to decide properties of TA such as reachability. Contrary to a widespread belief [4], we show that the classical semantics of ECA does not enjoy these properties. To prove that ECA admit no finite time-abstract bisimulation in general, we prove a stronger result: **the subclass of EPA (under the classical semantics) admits no finite time-abstract language equivalence**.

Let $\mathscr{C}$ be a class of timed transition systems on the alphabet Σ. Let us first recall the three classical equivalence notions on clock valuations:

- $\approx_L \subseteq \mathscr{V}(\mathbb{C}_\Sigma) \times \mathscr{V}(\mathbb{C}_\Sigma)$ is a *time abstract language equivalence* for the class $\mathscr{C}$ iff for all $\mathscr{T} \in \mathscr{C}$, for all pairs of states (q, v_1) and (q, v_2) of $\mathscr{T}$, s.t. $(v_1, v_2) \in \approx_L$: $\mathsf{Untime}(\mathsf{L}(q, v_1)) = \mathsf{Untime}(\mathsf{L}(q, v_2))$.
- $\lesssim \subseteq \mathscr{V}(\mathbb{C}_\Sigma) \times \mathscr{V}(\mathbb{C}_\Sigma)$ is a *time abstract simulation relation* for the class $\mathscr{C}$ iff, for all $\mathscr{T} \in \mathscr{C}$, for all pairs of states (q, v_1) and (q, v_2) of $\mathscr{T}$, s.t. $(v_1, v_2) \in \lesssim$, for all $t_1 \in \mathbb{R}^{\geq 0}$, for all $a \in \Sigma$: $(q, v_1) \xrightarrow{t_1,a} (q', v'_1)$ implies that there exists $t_2 \in \mathbb{R}^{\geq 0}$ s.t. $(q, v_2) \xrightarrow{t_2,a} (q', v'_2)$ and $v'_1 \lesssim v'_2$. In this case, we say that v_2 *simulates* v_1. Finally, $\simeq \subseteq \mathscr{V}(\mathbb{C}_\Sigma) \times \mathscr{V}(\mathbb{C}_\Sigma)$ is a *time abstract simulation equivalence* iff there exist time abstract simulation relations $\lesssim_1$ and $\lesssim_2$ s.t. $v_1 \simeq v_2$ iff $v_1 \lesssim_1 v_2$ and $v_2 \lesssim_2 v_1$.
- $\sim$ is a *time abstract bisimulation equivalence* for the class $\mathscr{C}$ iff it is a *symmetric* time abstract simulation for the class $\mathscr{C}$.

We say that an equivalence relation is *finite* iff it is of finite index. The relationship between those different equivalence notions is easy to establish, following the definition:

Lemma 8 *Any time abstract bisimulation is a time abstract simulation equivalence, and any time abstract simulation equivalence is a time abstract language equivalence*

Let us first prove the absence of *finite* time abstract language equivalence for the class of timed transition systems generated by EPA, according to the classical semantics (see Definition 2). We establish this results thanks to A_{inf} depicted in Fig. 3, with set of clocks $\mathbb{P}_{\{a,b,c\}}$ (it is thus well an EPA):

Proposition 9 *There is no finite time abstract language equivalence for* EPA*, using the classical semantics.*

Proof Let us assume that $\approx_L$ is a time abstract language equivalence on the class of timed transition systems generated by EPA (using the semantics of Definition 2). We will show, thanks to A_{inf}, that $\approx_L$ has necessarily *infinitely* many equivalence classes.

For any $n \in \mathbb{N}$, let v^n denote the *initial* valuation of $\mathbb{P}_{\{a,b,c\}}$ s.t. $v^n(\overrightarrow{x_a}) = n$, $v^n(\overrightarrow{x_b}) = 0$ and $v^n(\overrightarrow{x_c}) = \perp$, and let θ^n denote the timed word $(b, 0)(b, 1)(b, 2) \cdots (b, n-1)(a, n)$. Observe that, for any $n \geq 0$, there is only one finite run of A_{inf} starting in (q_0, v^n) and this run accepts the finite word θ^n. Hence, for any $n \geq 0$: $\mathsf{Untime}(\mathsf{L}(A, (q_0, v^n))) = \mathsf{Untime}(\{\theta^n\}) = b^n a$.

Let $V^{\mathbb{N}} = \{v^n \mid n \in \mathbb{N}\}$. Clearly, for all $v^{n_1}, v^{n_2} \in V^{\mathbb{N}}$ with $n_1 \neq n_2$, we have $v^{n_1} \not\approx_L v^{n_2}$, and so $\approx_L$ has necessarily an infinite number of equivalence classes. Thus, there is no *finite* time abstract language equivalence on the class of EPA. □

Summing up Proposition 9, and Lemma 8, we obtain:

Corollary 10 *With the classical semantics, there is no* finite *time abstract language equivalence, no* finite *time abstract simulation equivalence and no* finite *time abstract bisimulation for* EPA *and for* ECA.

Observe however, that in the case of ERA, there *is* a finite time-abstract bisimulation, which is the region equivalence [3], that we discuss in Sect. 5. Before that, we introduce an alternative semantics for ECA, that admits a finite time abstract bisimulation.

4 An alternative semantics for ECA

In this section, we define an alternative semantics for ECA, that we call the *weak semantics*. The benefits of this new definition are twofold. First, the weak semantics preserves the untimed language of the ECA. Second, the classical region equivalences (as defined in the next section) do form a finite time-abstract bisimulation on the weak semantics (unlike the classical one). Hence, the weak semantics allows to build a region automaton that accepts exactly $\mathsf{Untime}(\mathsf{L}(A))$ for all ECA A.

Weak time successors We start with the definition of the set of *weak time successors* of some valuation v by t time units:

$$v +_{\mathrm{w}} t = \left\{ v' \in \mathscr{V}(\mathbb{C}_\Sigma) \,\middle|\, \forall x : \begin{array}{c} \big(x \in \mathbb{P}_\Sigma \text{ and } v(x) > cmax\big) \text{ implies } v'(x) > cmax - t \\ \text{and} \\ \big(x \notin \mathbb{P}_\Sigma \text{ or } v(x) \leq cmax \text{ or } v(x) = \bot\big) \text{ implies } v'(x) = (v+t)(x) \end{array} \right\}$$

As can be seen, weak time successors introduce non-determinism on prophecy clocks that are larger than *cmax*. So, $v +_{\mathrm{w}} t$ is a *set* of valuations. Observe that for all v, t: $(v+t) \in v +_{\mathrm{w}} t$.

Weak transition system We can now define the weak semantics of ECA:

Definition 11 Let $A = \langle Q, q_i, \Sigma, C, \delta, \alpha\rangle$ be an ECA. Its *weak semantics* is the infinite timed transition system $\mathsf{TS}^w_A = (Q^A, Q^A_i, \rightarrow_w, \alpha^A)$ where:

(1) $Q^A = Q \times \mathscr{V}(C)$ is the set of states;
(2) $Q^A_i = \{(q_i, v) \mid v \text{ is initial}\}$ is the set of initial states;
(3) $\alpha^A = \{(q, v) \mid q \in \alpha \text{ and } v \text{ is final }\}$ is the set of final sates;
(4) the weak transition relation $\rightarrow_w \subseteq (Q^A \times \mathbb{R}^{\geq 0} \times Q^A) \cup (Q^A \times \Sigma \times Q^A)$ is such that:

   (a) $\big((q, v), a, (q', v')\big) \in \rightarrow_w$ iff there are $(q, a, \psi, q') \in \delta$ and $\overline{v} \in \mathscr{V}(C)$ s.t. $\overline{v}[\overrightarrow{x_a} := 0] = v$, $\overline{v}[\overleftarrow{x_a} := 0] = v'$ and $\overline{v} \models \psi$, and
   (b) $\big((q, v), t, (q, v')\big) \in \rightarrow_w$ iff $v' \in v +_{\mathrm{w}} t$.

Observe that TS^w_A and TS_A differ only in the way the elapsing of time is handled. In the sequel, we denote by $\mathsf{wL}(A, q)$ the set of words accepted by some run of TS^w_A starting in q, and by $\mathsf{wL}(A)$ the language accepted by some run of TS^w_A starting in an initial state. Runs of

TS_A^w are called *weak runs* of A. Observe that each run of A (under the classical semantics) is also a weak run of A. It is thus immediate that $\mathsf{L}(A) \subseteq \mathsf{wL}(A)$. However, the converse also holds. Indeed, the non-determinism in the elapsing of time (in weak runs) occurs only for prophecy clocks which are larger than *cmax*. Since the ECA cannot distinguish between those valuations, we can, from any weak run, build a run in the sense of the classical semantics, by adapting the values of the prophecy clocks larger than *cmax*, when need be. This is the intuition behind the proof of the next proposition:

Proposition 12 *For all* ECAA*:* $\mathsf{L}(A) = \mathsf{wL}(A)$.

Proof We have already established that $\mathsf{L}(A) \subseteq \mathsf{wL}(A)$. Let us show that $\mathsf{L}(A) \supseteq \mathsf{wL}(A)$. Let $\theta = (\tau_0, w_0) \cdots (\tau_n, w_n)$ be a timed word in $\mathsf{wL}(A)$, and let $(q_0, v_0) \xrightarrow{t_0}_w (q_0, v_0') \xrightarrow{w_0} (q_1, v_1) \cdots (q_{n+1}, v_{n+1})$ be the corresponding accepting weak run of A. We can also build an accepting run of A on θ by setting the valuations of the prophecy clocks greater than *cmax* to a value compatible with real time-elapsing. For any $0 \le i \le n$, we build $\overline{v}_i$ as follows. For all clock x, s.t. $x \in \mathbb{P}_\Sigma$ and $v_i'(x) > cmax$, we let $k_x > i$ denote the least position s.t. $v_{k_x}'(x) \le cmax$. Remark that such a position always exists in an *accepting* run, because all prophecy clocks will eventually be equal to $\bot$. Then define:

$$\overline{v}_i(x) = \begin{cases} v_{k_x}'(x) + \sum_{j=i+1}^{k_x} t_j & \text{if } x \in \mathbb{P}_\Sigma \text{ and } v_i'(x) > cmax \\ v_i'(x) & \text{otherwise} \end{cases}$$

Remark that v_i' and $\overline{v}_i$ differ only on prophecy clocks larger than *cmax*, and that $v_i'(x) > cmax$ iff $\overline{v}_i(x) > cmax$ for any i and x. Moreover, the sequence of $\overline{v}_i$ clearly form a sequence of time successors. We further define $\tilde{v}_i$ for all i as follows: $\tilde{v}_i(x) = t_i + \overline{v}_i(x)$ for all $x \in \mathbb{P}_\Sigma$ s.t. $v_i(x) > cmax$ and $\tilde{v}_i(x) = v_i(x)$ otherwise. Hence, it can be checked that for all $0 \le i \le n$, $(q_i, \tilde{v}_i) \xrightarrow{t_i} (q_i, \overline{v}_i) \xrightarrow{w_i} (q_{i+1}, \tilde{v}_{i+1})$, and so that $(q_0, \tilde{v}_0) \xrightarrow{t_0} (q_0, \overline{v}_0) \xrightarrow{w_0} (q_1, \tilde{v}_1) \xrightarrow{t_1} (q_1, \overline{v}_1) \cdots (q_{n+1}, \tilde{v}_{n+1})$. Moreover, we let $\tilde{v}_{n+1}(x) = v_{n+1}(x) = \bot$ for all $x \in \mathbb{P}_\Sigma$. Thus, $\theta \in \mathsf{L}(A)$ and thus, $\mathsf{L}(A) \supseteq \mathsf{wL}(A)$. □

5 Regions and event clocks

In this section, we recall and discuss the classical notion of *region* that is known to form *finite time-abstract bisimulation* for the class of TA. While Corollary 10 tells us that regions are not a time-abstract bisimulation for ECA, when considering the classical semantics (contrary to what was claimed in [4]), we show that they are a (finite) time-abstract bisimulation when considering the *weak semantics* that we have introduced in the previous section (see Definition 11). We close the section by discussing several notions of *region automaton*, and show which ones allow to recognise the untimed language of the ECA.

5.1 Regions

Let us fix a set of clocks $C \subseteq \mathbb{C}_\Sigma$ and a constant $cmax \in \mathbb{N}$. Let us first recall the notion of *region equivalence* for ECA[4] (a straightforward adaptation of the Alur-Dill region equivalence for TA [3]). This equivalence is denoted by $\approx_{cmax}$: for all $v_1, v_2 \in \mathcal{V}(C)$: $v_1 \approx_{cmax} v_2$ iff:

(C1) for all $x \in C$, $v_1(x) = \bot$ iff $v_2(x) = \bot$,

(C2) for all $x \in C$ s.t. $v_1(x) \neq \bot$: either $v_1(x) > cmax$ and $v_2(x) > cmax$, or $\lceil v_1(x) \rceil = \lceil v_2(x) \rceil$ and $\lfloor v_1(x) \rfloor = \lfloor v_2(x) \rfloor$,

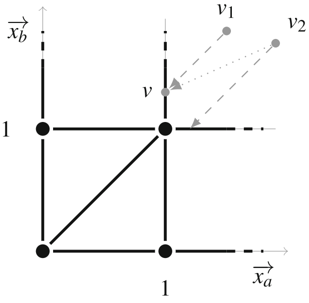


Fig. 4 The set of regions $\mathsf{Reg}\left(\mathbb{P}_{\{a,b\}}, 1\right)$. *Dashed arrows* show the trajectories followed by the valuations with time elapsing in the classical semantics. The dotted arrow indicates a potential *weak time successor* of v_2

(C3) for all $x_1, x_2 \in C$ s.t. $v_1(x_1) \neq \bot$, $v_1(x_2) \neq \bot$, $v_1(x_1) \leq cmax$ and $v_1(x_2) \leq cmax$: $\langle v_1(x_1)\rangle \leq \langle v_1(x_2)\rangle$ if and only if $\langle v_2(x_1)\rangle \leq \langle v_2(x_2)\rangle$.

Equivalence classes of $\approx_{cmax}$ are called *regions* and the set of those classes is denoted by $\mathsf{Reg}\,(C, cmax)$.

Example 13 Fig. 4, illustrates $\mathsf{Reg}\left(\{\overrightarrow{x_a}, \overrightarrow{x_b}\}, 1\right)$. The *dashed arrows* show the respective trajectories of valuations v_1 and v_2 with time elapsing (under the classical notion of time successor). The dotted arrow shows one potential *weak* time successor (v) of v_2

Observe that, for any *cmax*, and for any finite set of clocks C, $\mathsf{Reg}\,(C, cmax)$ is a *finite* set. A region r on set of clocks C is *initial* (resp. *final*) iff it contains only initial (final) valuations.

Regions are not a language equivalence for the classical semantics Since the regions defined above are of finite index, Corollary 10 implies that, when considering the classical semantics, they cannot form a language equivalence for ECA. This implies that regions are not a *time abstract bisimulation*, contrary to what was claimed in the seminal paper on ECA[4]. Let us explain intuitively why it is not the case. Consider $\mathsf{Reg}\left(\mathbb{P}_{\{a,b\}}, 1\right)$ and the two valuations v_1 and v_2 in Fig. 4. Clearly, v_1 can reach the region where $\overrightarrow{x_a} = 1$ and $\overrightarrow{x_b} > 1$, while v_2 cannot. Conversely, v_2 can reach $\overrightarrow{x_a} > 1$ and $\overrightarrow{x_b} = 1$ but v_2 cannot. It is easy to build an ECA with $cmax = 1$ that distinguishes between those two cases and accepts different words.

When considering the depiction of regions in Fig 4 and the reason why they are not a language equivalence, it is tempting to try and refine regions as in Fig. 5. Remark that this refinement corresponds to the definition of regions introduced by Bouyer in [9]. Clearly, in this case, v_1 and v_2 are in different regions. Yet, one can be convinced that this refinement fails to be a time abstract language equivalence (as implied by Corollary 10) by considering now valuations v_3 and v_4 in Fig. 5. It is easy to see that, for A_{inf} in Fig. 3: $\mathsf{Untime}(\mathsf{L}(A_{\inf}, (q_0, v_3))) = \{bbba\} \neq \{bbbba\} = \mathsf{Untime}(\mathsf{L}(A_{\mathsf{inf}}, (q_0, v_4)))$, although v_3 and v_4 belong to the same region. Indeed, from v_3, the (q_0, q_0) edge can be taken 3 times before we reach $\overrightarrow{x_a} = 1$ and the (q_0, q_1) edge can be fired. However, the (q_0, q_0) edge has to be taken 4 times from v^4 before we reach $\overrightarrow{x_a} = 1$ and the (q_0, q_1) edge can be taken. The problem can already be observed when the first b-labeled transition is fired in A_{inf}. Letting 1 time unit elapse from v_3, then firing the transition and letting $\overrightarrow{x_b}$ take value 1 after the transition reaches region r_1 (dashed gray line from v_3 on the figure). On the other hand, performing the same actions from v_4 (dashed gray line from v_4 on the figure) reaches region r_2.

These two examples illustrate the issue with *prophecy clocks* and regions. Roughly speaking, to keep the set of regions finite, valuations where the clocks are *too large* (for instance, $> cmax$ in the case of $\mathsf{Reg}\,(C, cmax)$) belong to the same region. This is not a problem for

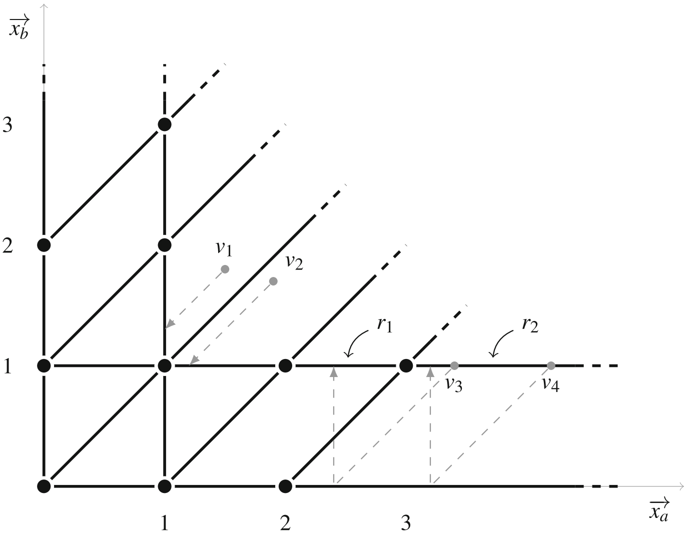


Fig. 5 A possible refinement of $\mathsf{Reg}\left(\mathbb{P}_{\{a,b\}}, 1\right)$. *Curved arrows* are used to refer to selected regions

history clocks as a history clock larger than *cmax* remains over *cmax* with time elapsing, until it is reset. This is not the case for prophecy clocks whose values *decrease with time elapsing*: eventually, those clocks reach a value $\leq$ *cmax*, but the region equivalence is too coarse to allow to predict the region they reach.

However, note that, when considering the *weak semantics* as introduced in the previous section, valuation v in Fig. 4 *is* a potential *weak time successor* of v_2. Actually, we will prove in Theorem 16 hereunder that *regions are a time abstract bisimulation for the weak semantics*.

Moreover, under the *classical semantics*, and when we restrict to the class of *event recording automata*, regions form a finite time-abstract bisimulation too. This is not surprising as an ERA is essentially a special case of timed automaton. More precisely, each ERA can be turned into a timed automaton as follows: for each clock $\overleftarrow{x_a}$ in the ERA, introduce a clock x_a in the TA which is reset every time an a-labeled transition is fired [4]. As this construction preserves determinism, and since regions form a finite time abstract bisimulation for TA[3], so do they for ERA:

Theorem 14 *For all cmax* $\in \mathbb{N}$, $\approx_{cmax}$ *is a finite time abstract bisimulation for the class of (timed transitions systems generated by)* ERA *with maximal constant cmax.*

Regions are a time-abstract bisimulation for the weak semantic As already explained, the main benefit of the weak semantics introduced in Sect. 4 is that regions are a time-abstract bisimulation in this case. To prove this result we rely on the following property which is reminiscent of time abstract bisimulation:

Lemma 15 *Let C be a set of clocks and let cmax* $\in \mathbb{N}$. *For all* $v_1, v_2 \in \mathcal{V}(C)$ *s.t.* $v_1 \approx_{cmax} v_2$, *for all* $t_1 \in \mathbb{R}^{\geq 0}$, *there exist* t_2 *and* $v' \in (v_2 +_{\mathrm{w}} t_2)$ *s.t.* $v_1 + t_1 \approx_{cmax} v'$.

Proof The cases where $v_1 \approx_{cmax} v_1 + t_1$ are trivial. We first restrict ourselves to the case where v_1 and $v_1 + t_1$ belong to adjacent regions, that is,

$$\forall 0 \leq t' \leq t_1, \begin{pmatrix} \text{either } v_1 + t' \approx_{cmax} v_1, \\ \text{or } v_1 + t' \approx_{cmax} v_1 + t_1 \end{pmatrix} \tag{1}$$

Let us now show how to chose t_2. Let C_v^0 denote the set of clocks x s.t. $\langle v(x) \rangle = 0$. Under the hypothesis (1), we have to consider two cases:

(1) Either $C_{v_1}^0 = \varnothing$ and $C_{v_1+t_1}^0 \neq \varnothing$. In that case, let x be a clock in $C_{v_1+t_1}^0$. We let $t_2 = \langle v_2(x) \rangle$

(2) Or $C_{v_1}^0 \neq \varnothing$ and $C_{v_1+t_1}^0 = \varnothing$. In that case, we need to consider two sub-cases. If there is x s.t. $\langle v_2(x) \rangle \neq 0$, we let t_2 be a value s.t. $0 < t_2 < \min\{\langle v_2(x) \rangle \mid \langle v_2(x) \rangle \neq 0\}$. Otherwise, all the clocks in v_2 have a null fractional part, and we can take any delay < 1 for t_2: we let $t_2 = 0.1$.

Now, let us show that there exists $v \in v_2 +_{\mathrm{w}} t_2$ s.t. $v \approx_{cmax} v_1 + t_1$. For that purpose, we first build a valuation v_3 as follows. For any history clock x, we let $v_3(x) = v_2(x)$. For all prophecy clocks x s.t. $v_2(x) \leq cmax$, or $v_2(x) = \bot$, we let $v_3(x) = v_2(x)$ too. For all prophecy clocks x s.t. $v_2(x) > cmax$ (and thus $v_1(x) > cmax$ since $v_1 \approx_{cmax} v_2$), we consider two cases. Either $(v_1 + t_1)(x) > cmax$. In that case we let $v_3(x) = cmax + t_2 + 1$. Or $(v_1 + t_1)(x) = cmax$. In that case we let $v_3(x) = cmax + t_2$. Remark that the case $(v_1 + t_1)(x) < cmax$ is not possible since we have assumed that $v_1(x) > cmax$ and that v_1 and $v_1 + t_1$ are in adjacent regions.

We now let $v' = v_3 + t_2$. It is easy to check that $v' \approx_{cmax} (v_1 + t_1)$. Moreover, $v' \in (v_2 +_{\mathrm{w}} t_2)$, since v_3 has been obtained from v_2 by replacing values larger than *cmax* by other values larger than *cmax*.

To conclude, observe that if $v_3 \in (v_2 +_{\mathrm{w}} t_2)$ and $v_2 \in (v_1 +_{\mathrm{w}} t_1)$, then $v_3 \in (v_1 +_{\mathrm{w}} (t_1 + t_2))$. This allows to handle the case where v_1 and $v_1 + t_1$ are not in adjacent regions: by decomposing t_1 into a sequence $t_1', t_2', \ldots, t_n'$ s.t. $t_1 = t_1' + t_2' + \cdots + t_n'$, and for all $1 \leq i < n$, $v_1 + \sum_{j=1}^{i} t_j'$ and $v_1 + \sum_{j=1}^{i+1} t_j'$ are in adjacent regions. Then, applying the reasoning above, we get a sequence $t_1'', \ldots, t_n''$ of time delays and a sequence $v_0', v_1', \ldots, v_n'$ of valuations s.t. $v_0' = v_2$, for all $0 \leq i < n$, $v_{i+1}' \in v_i' +_{\mathrm{w}} t_i''$ and $v_{i+1}' \approx_{cmax} v_1 + \sum_{j=1}^{i+1} t_j'$. Thus, $v_n' \in v_2 +_{\mathrm{w}} \sum_{j=1}^{n} t_j''$ and $v_n' \approx_{cmax} v_1 + \sum_{j=1}^{n} t_j' = v_1 + t_1$. □

We can now prove that regions are a timed abstract bisimulation for the weak semantics:

Theorem 16 Under the weak semantics, $\approx_{cmax}$ *is a time abstract bisimulation for the class of* ECA *with maximal constant cmax.*

Proof Let v_1 and v_2 be two valuations s.t. $v_1 \approx_{cmax} v_2$, and let q be an ECA location.

We consider separately the discrete and continuous transitions that can occur in the transitions systems. Let $a \in \Sigma$ be such that $(q, v_1) \xrightarrow{a} (q', v_1')$. Thus, there are $(q, a, \psi, q') \in \delta$ and $\overline{v} \in \mathscr{V}(C)$ such that $\overline{v}[\overrightarrow{x_a} := 0] = v_1$, $\overline{v}[\overleftarrow{x_a} := 0] = v_1'$ and $\overline{v} \models \psi$. Let v_2' be the valuation defined by

$$v_2'(x) = \begin{cases} v_2(x) & \text{if } x \in C \setminus \{\overrightarrow{x_a}, \overleftarrow{x_a}\} \\ 0 & \text{if } x = \overleftarrow{x_a} \\ \overline{v}(x) & \text{if } x = \overrightarrow{x_a} \end{cases}$$

and $\overline{\overline{v}}$ be the valuation defined by $\overline{\overline{v}}[\overrightarrow{x_a} := 0] = v_2$ and $\overline{\overline{v}}[\overleftarrow{x_a} := 0] = v_2'$. It is immediate that that $\overline{v} \approx_{cmax} \overline{\overline{v}}$ and thus that $\overline{\overline{v}} \models \psi$. Hence, $(q, v_2) \xrightarrow{a} (q, v_2')$ with $v_1' \approx_{cmax} v_2'$.

Now let us assume a timed transition $(q, v_1) \xrightarrow{t_1} (q, v_1')$ with $v_1' \in v_1 +_{\mathrm{w}} t_1$.

- If $v_1' = v_1 + t_1$, by Lemma 15, there exist $t_2 \in \mathbb{R}^{\geq 0}$ and $v_2' \in \mathscr{V}(C)$ such that $v_2' \in v_2 +_{\mathrm{w}} t_2$ and $v_1' \approx_{cmax} v_2'$.
- Otherwise, let $C_{v_1}^{cmax}$ be the set of prophecy clocks such that $v_1(x) > cmax$. We let $\overline{v_1}$ be the valuation such that $v_1' = \overline{v_1} + t_1$. Observe that, for all $x \notin C_{v_1}^{cmax}$, $\overline{v_1}(x) = v_1(x)$, and for all $x \in C_{v_1}^{cmax}$, since $v_1'(x) > cmax - t_1$ (by definition), we deduce that $\overline{v_1}(x) > cmax$. Hence $\overline{v_1} \approx_{cmax} v_1 \approx_{cmax} v_2$, and, by Lemma 15, there exist $t_2 \in \mathbb{R}^{\geq 0}$ and $v_2' \in \mathscr{V}(C)$ such that $v_2' \in v_2 +_{\mathrm{w}} t_2$ and $v_2' \approx_{cmax} v_1'$. □

Discussion This result might seem in contradiction with the facts that (i) both the classical and the weak semantics accept the same language ($\mathsf{L}(A) = \mathsf{wL}(A)$, Proposition 12), which implies that $\mathsf{Untime}(\mathsf{L}(A)) = \mathsf{Untime}(\mathsf{wL}(A))$ and (ii) ECA admit no finite time-abstract language equivalence (Corollary 10). One might thus wonder why the region equivalence, which is a time-abstract language equivalence for the weak semantics, cannot be used to define a finite time-abstract language equivalence for the classical semantics. To explain why those results are coherent, we consider again the EPA A_{inf} (Fig. 3). Let v be the valuation s.t. $v(\overrightarrow{x_a}) = 42$, $v(\overrightarrow{x_b}) = 0$, and $v(\overrightarrow{x_c}) = \bot$. Then, it is easy to check that $\mathsf{Untime}(\mathsf{L}(A, (q_0, v)) = \{b^{42}a\})$ but that $\mathsf{Untime}(\mathsf{wL}(A, (q_0, v))) = \{b^i a \mid i \geq 1\}$. That is, more words can be accepted from (q_0, v) in the weak semantics than in the classical semantics. However, these extra words do not modify the accepted language of the whole ECA A, since they can be accepted, in the classical semantics, from *other initial configurations*, as we have just shown. Now consider v' s.t. $v'(\overrightarrow{x_a}) = 2$, $v'(\overrightarrow{x_b}) = 0$, and $v'(\overrightarrow{x_c}) = \bot$. Similarly, $\mathsf{Untime}(\mathsf{L}(A, (q_0, v'))) = \{b^2 a\}$ and $\mathsf{Untime}(\mathsf{wL}(A, (q_0, v'))) = \{b^i a \mid i \geq 1\}$. It is then clear that v and v' are time-abstract language equivalent in the weak semantics (and indeed $v \approx_1 v'$), but not in the classical one.

5.2 Region automata

Let us now consider the consequences of Corollary 10 and Theorem 16 on the notion of region automaton. We first define two variants of the region automaton:

Definition 17 Let $\mathscr{T} = \langle Q^A, Q_i^A, \rightarrow_A, \alpha^A \rangle$ be a timed transition system which is either TS_A or TS_A^w for some ECA $A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$. Let $\mathscr{R}$ be a set of regions on $\mathscr{V}(C)$. Then, the **existential** (resp. **universal**) *$\mathscr{R}$-region automaton of $\mathscr{T}$* is the finite transition system $RA(\exists, \mathscr{R}, \mathscr{T})$ (resp. $RA(\forall, \mathscr{R}, \mathscr{T})$) defined by $\langle Q^R, Q_i^R, \Sigma, \delta^R, \alpha^R \rangle$ s.t.:

(1) $Q^R = Q \times \mathscr{R}$
(2) $Q_i^R = \{(q_i, r) \mid r \text{ is an initial region}\}$
(3) $\delta^R \subseteq Q^R \times \Sigma \times Q^R$ is s.t. $\big((q_1, r_1), a, (q_2, r_2)\big) \in \delta$ iff **there exists a valuation** (resp. **for all valuations**) $v_1 \in r_1$, there exists a time delay $t \in \mathbb{R}^{\geq 0}$ and a valuation $v_2 \in r_2$ s.t. $(q_1, v_1) \xrightarrow{t,a}_A (q_2, v_2)$.
(4) $\alpha^R = \{(q, r) \mid q \in \alpha \text{ and } r \text{ is a final region}\}$

Let $R = \langle Q^R, Q_i^R, \Sigma, \delta^R, \alpha^R \rangle$ be a region automaton and w be an (untimed) word over Σ. A *run* of R on $w = w_0 w_1 \dots w_n$ is a finite sequence $(q_0, r_0)(q_1, r_1) \dots (q_{n+1}, r_{n+1})$ of states of R such that: $(q_0, r_0) \in Q_i^R$ and such that: for all $0 \leq i \leq n$: $\big((q_i, r_i), w_i, (q_{i+1}, r_{i+1})\big) \in \delta^R$. Such a run is *accepting* iff $(q_{n+1}, r_{n+1}) \in \alpha^R$ (in that case, we say that w is accepted by R). The language $\mathsf{L}(R)$ of R is the set of all untimed words accepted by R.

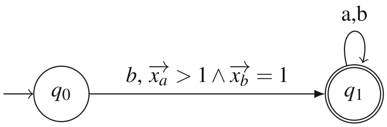
Let A be an ECA with alphabet Σ and maximal constant $cmax$. If we adapt and apply the notion of region automaton, as defined for TA [3], to A, considering the classical semantics, we

Table 1 The different region automata we consider

	Classical semantics	Weak semantics
Univ.	$\mathsf{RegAut}_{\forall}(A) = RA(\forall, \mathsf{Reg}(\mathbb{C}_\Sigma, cmax), \mathsf{TS}_A)$	$\mathsf{wRegAut}_{\forall}(A) = RA(\forall, \mathsf{Reg}(\mathbb{C}_\Sigma, cmax), \mathsf{TS}_A^w)$
Exist.	$\mathsf{RegAut}_{\exists}(A) = RA(\exists, \mathsf{Reg}(\mathbb{C}_\Sigma, cmax), \mathsf{TS}_A)$	$\mathsf{wRegAut}_{\exists}(A) = RA(\exists, \mathsf{Reg}(\mathbb{C}_\Sigma, cmax), \mathsf{TS}_A^w)$

Gray cells indicate the automata that recognise $\mathsf{Untime}(\mathsf{L}(A))$

Fig. 6 An $\mathsf{ECA}\, A$ such that $\mathsf{Untime}(\mathsf{L}(A)) \neq \varnothing$ and $\mathsf{L}(\mathsf{RegAut}_{\forall}(A)) = \varnothing$


a,b
$b, \overrightarrow{x_a} > 1 \wedge \overrightarrow{x_b} = 1$
q_0
q_1

obtain $RA(\forall, \mathsf{Reg}(\mathbb{C}_\Sigma, cmax), \mathsf{TS}_A)$. To alleviate notations, we denote it by $\mathsf{RegAut}_{\forall}(A)$. In the rest of the paper, we also consider several other variants that we denote by the shortcuts given in Table 1.

Observe that in the case of TA, the distinction between universal and existential region automata has no influence, because regions form a time-abstract bisimulation, and thus existential and universal region automata coincide. Let us see how these results adapt (or not) to ECA.

Recognized language of $\mathsf{RegAut}_{\forall}(A)$ Let us show that, in general, when the *classical semantics* is considered, the *universal* region automaton *does not recognize the untimed language of the* ECA.

Lemma 18 *There is an* $\mathsf{ECA}\, A$ *such that* $\mathsf{L}(\mathsf{RegAut}_{\forall}(A)) \subsetneq \mathsf{Untime}(\mathsf{L}(A))$.

Proof Consider the automaton A_{inf} in Fig. 3, with $cmax = 1$. Assume there is an edge of the form $\big((q_0, r), b, (q_0, r')\big)$ in $\mathsf{RegAut}_{\forall}(A_{\mathsf{inf}})$, where r is initial. By the guard of the (q_0, q_0) loop, r' is a region s.t. for all $v \in r'$: $v(\overrightarrow{x_b}) = 1$ and $v(\overrightarrow{x_a}) > 1$. To fire the (q_0, q_0) loop again, we need to let time elapse up to the point where $\overrightarrow{x_b} = 0$. Then consider two valuations v and v' s.t. $v(\overrightarrow{x_b}) = v'(\overrightarrow{x_b}) = 1$, $v(\overrightarrow{x_a}) = 1.1$ and $v'(\overrightarrow{x_a}) = 2.1$. Clearly, $\{v, v'\} \subseteq r'$. However, firing the (q_0, q_0) loop from (q_0, v) leads to (q_0, v''), with $v''(\overrightarrow{x_a}) = 0.1$, and firing the same (q_0, q_0) loop from (q_0, v') leads to (q_0, v''') with $v'''(\overrightarrow{x_a}) = 1.1$. Thus, v'' and v''' do not belong to the same region. Since we are considering a *universal* automaton, we conclude that there is no edge of the form $\big((q_0, r'), b, (q_0, r'')\big)$. Hence, $\mathsf{RegAut}_{\forall}(A_{\mathsf{inf}})$ cannot recognize an arbitrary number of b's from any of its initial states (actually, one can easily check that the accepted language of $\mathsf{RegAut}_{\forall}(A_{\mathsf{inf}})$ is abc^*). Thus, $\mathsf{L}(\mathsf{RegAut}_{\forall}(A_{\mathsf{inf}})) \subsetneq \mathsf{Untime}(\mathsf{L}(A_{\mathsf{inf}}))$. □

Remark 19 Note that there also exists an $\mathsf{ECA}\, A$ such that $\mathsf{L}(\mathsf{RegAut}_{\forall}(A)) = \varnothing$ although $\mathsf{Untime}(\mathsf{L}(A)) \neq \varnothing$, hence the universal region automaton cannot be used to solve the emptiness problem of an ECA. Consider for instance the ECA pictured in Fig. 6. In this case, $\mathsf{Untime}(\mathsf{L}(A)) = bbb^*a(\{a, b\})^*$. In $\mathsf{RegAut}_{\forall}(A)$, there is an edge of the form $((q_0, r), b, (q_1, r'))$ where r is an initial region. Moreover, the guard on the transition from q_0 to q_1 in A implies that r' is such that : for all valuation $v \in r'$, $v(\overrightarrow{x_a}) > 1$ and $v(\overrightarrow{x_b}) = 1$. As in the proof of Lemma 18, it is easy to show that one can find two valuations v and v' in r' such that, after letting 1 time unit elapse, $v + 1$ and $v' + 1$ will not belong to the same region, hence (q_1, r') has no successor in the universal region automaton and $\mathsf{L}(\mathsf{RegAut}_{\forall}(A)) = \varnothing$.

Recognized language of region automata for the weak semantics Thanks to the time abstract bisimulation property enjoyed by the regions on the weak semantics (Theorem 16), and thanks to the fact that $\mathsf{L}(A) = \mathsf{wL}(A)$ for all $\mathsf{ECA}\, A$ (Proposition 12) we can show that both the existential and the universal region automata defined on the weak semantics recognise the untimed language of A. To obtain this result, we rely on the following stronger lemma, stating that, in the weak semantics, existential and universal automata are the same objects:

Lemma 20 *For all* $\mathsf{ECA}\, A$*:* $\mathsf{wRegAut}_\forall (A) = \mathsf{wRegAut}_\exists (A)$.

Proof By definition, $\mathsf{wRegAut}_\forall (A)$ and $\mathsf{wRegAut}_\exists (A)$ have the same sets of states, final states and initial states. Still by definition, $\rightarrow^\forall \subseteq \rightarrow^\exists$, where $\rightarrow^\forall$ and $\rightarrow^\exists$ denote the transition relations of $\mathsf{wRegAut}_\forall (A)$ and $\mathsf{wRegAut}_\exists (A)$ respectively. However, thanks to the time-abstract bisimulation property (Theorem 16), we also have $\rightarrow^\forall \supseteq \rightarrow^\exists$. Hence the Lemma. □

Then, we can show that both region automata for the weak semantics recognise the untimed language of the ECA:

Theorem 21 *For all* $\mathsf{ECA}\, A$*:* $\mathsf{L}(\mathsf{wRegAut}_\forall (A)) = \mathsf{L}(\mathsf{wRegAut}_\exists (A)) = \mathsf{Untime}(\mathsf{L}(A))$

Proof Since $\approx_{cmax}$ is a time-abstract bisimulation on TS^w_A, by Theorem 16, we can adapt the classical proof on timed automata [3] to show that: $\mathsf{L}(\mathsf{wRegAut}_\forall (A)) = \mathsf{Untime}(\mathsf{wL}(A))$ Then, thanks to Lemma 20, $\mathsf{L}(\mathsf{wRegAut}_\exists (A)) = \mathsf{L}(\mathsf{wRegAut}_\forall (A))$ for all $\mathsf{ECA}\, A$. Hence, we conclude that $\mathsf{wRegAut}_\forall (A)$ and $\mathsf{wRegAut}_\exists (A)$ both recognise $\mathsf{Untime}(\mathsf{wL}(A))$. Since, by Proposition 12, $\mathsf{wL}(A) = \mathsf{L}(A)$, we obtain the Theorem. □

Recognized language of $\mathsf{RegAut}_\exists (A)$ We complete the picture by showing that it is possible to define a region automaton that is based on the classical semantics of ECA and still recognises $\mathsf{Untime}(\mathsf{L}(A))$. It is the existential region automaton $\mathsf{RegAut}_\exists (A)$.

Lemma 22 *For all* $\mathsf{ECA}\, A$*:* $\mathsf{wRegAut}_\exists (A) = \mathsf{RegAut}_\exists (A)$.

Proof Let us assume that $\mathsf{wRegAut}_\exists (A) = \left\langle \tilde{Q}^R, \tilde{Q_i}^R, \tilde{\Sigma}, \tilde{\delta}^R, \tilde{\alpha}^R \right\rangle$ and that $\mathsf{RegAut}_\exists (A) = \langle Q^R, Q^R_i, \Sigma, \delta^R, \alpha^R \rangle$

First, recall that $\mathsf{wRegAut}_\exists (A)$ and $\mathsf{RegAut}_\exists (A)$ differ only on the transition relation, i.e. $\tilde{Q}^R = Q^R$, $\tilde{Q}_i = Q_i$, $\tilde{\Sigma} = \Sigma$ and $\tilde{\alpha}^R = \alpha^R$. The transitions relations $\tilde{\delta}^R$ and δ^R are based on the weak semantics and on the classical semantics respectively. Moreover, recall that these two semantics differ only on the elapsing of time and that, for all valuation v, and all time delay t, $(v + t) \in (v +_\mathrm{w} t)$. Thus, $\delta^R \subseteq \tilde{\delta}^R$.

Let $\big((q_1, r_1), a, (q_2, r_2)\big)$ be a transition in $\tilde{\delta}^R$, and let us show that it belongs to δ^R too. Since $\big((q_1, r_1), a, (q_2, r_2)\big) \in \tilde{\delta}^R$, there are $v_1 \in r_1$, a time delay $t \in \mathbb{R}^{\geq 0}$ and a valuation $v_2 \in r_2$ s.t. $(q_1, v_1) \xrightarrow{t,a}_w (q_2, v_2)$ (where $\rightarrow_w$ denotes the weak semantics). Thus, there is a valuation v s.t. $v \in v_1 +_\mathrm{w} t$, and $(q_1, v) \xrightarrow{a}_w (q_2, v_2)$. Let r be the region containing v. Let us build a valuation $v_1' \in r_1$ s.t. $v_1' + t = v$ as follows. For all clocks x:

$$v_1'(x) = \begin{cases} v(x) + t & \text{if } x \in \mathbb{P}_\Sigma \text{ and } v_1(x) > cmax \\ v_1(x) & \text{otherwise} \end{cases}$$

Let us show that $v_1' \in r_1$. First, we observe that v_1' and v_1 coincide on all clocks s.t. $v_1(x) \leq cmax$, and on all history clocks y s.t. $v_1(y) > cmax$. Thus to prove that $v_1' \in r_1$, it remains to show that, on all clocks $x \in \mathbb{P}_\Sigma$ s.t. $v_1(x) > cmax$, $v'_1(x) > cmax$ too. Since $v \in v_1 +_{\mathrm{w}} t$, and since $v_1(x) > cmax$, the definition of $+_{\mathrm{w}}$ ensures that $v(x) > cmax - t$. Thus, $v'_1(x) = v(x) + t > cmax - t + t = cmax$. Let us now show that $v'_1 + t = v$. The following holds for all clocks x. In the case where $x \in \mathbb{P}_\Sigma$ and $v_1(x) > cmax$, we have $(v'_1 + t)(x) = v'_1(x) - t = v(x) + t - t = v(x)$. Otherwise, $(v'_1 + t)(x) = (v_1 + t)(x)$ by definition of v'_1, and $(v_1 + t)(x) = v(x)$, by definition of the weak time successors, and since $v_1(x) \leq cmax$.

Thus, we have $(q_1, v'_1) \xrightarrow{t} (q_1, v)$ (where $\rightarrow$ denotes the classical semantics of ECA). Since both semantics coincide on discrete transitions, we have $(q_1, v'_1) \xrightarrow{t} (q_1, v) \xrightarrow{a} (q_2, v_2)$. We conclude that the transition $\big((q_1, r_1), a, (q_2, r_2)\big)$ is present in δ^R too. □

Thanks to the former lemma, and to Theorem 21, we can conclude that the existential region automaton (based on the classical semantics) accepts $\mathsf{Untime}(\mathsf{L}(A))$ (unlike its universal counterpart):

Theorem 23 *For all* $\mathsf{ECA}\,A$*:* $\mathsf{L}(\mathsf{RegAut}_\exists(A)) = \mathsf{Untime}(\mathsf{L}(A))$.

Size of the existential region automaton We have just introduced three constructions to obtain, from any $\mathsf{ECA}\,A$, a *finite automaton* recognising $\mathsf{Untime}(\mathsf{L}(A))$. This construction, however, is not the first to achieve this: in the original paper [4] on ECA, the following construction is proposed. First transform A into a non-deterministic timed automaton B s.t. $\mathsf{L}(B) = \mathsf{L}(A)$, using the technique recalled in Sect. 1, then compute the region automaton of B. Unfortunately, as stated by Theorem 7, building B can, in the worst case, incur a blow up in the number of clocks and locations. More precisely, it is well known [3] that the number of Alur-Dill regions on n clocks and with maximal constant $cmax$ is at most $R(n, cmax) = n! \times 2^n \times (2 \times cmax + 2)^n$. Thus, applying the classical region automaton construction [3] to the $\mathsf{TA}\,B$, obtained from the $\mathsf{ECA}\,A$ with alphabet Σ, m locations and maximal constant $cmax$, yields a region automaton with a number of states equal, in the worst case, to:

$$m \times 2^{(4cmax+6)\times|\Sigma|} \times R\big((4cmax + 6) \times |\Sigma|, cmax\big)$$

In the case of event clocks, we cannot directly reuse the value of $R(n, cmax)$ given above, as we have to take into account the $\perp$ value as a supplementary value for the clocks. Hence:

Lemma 24 *For all set of event clocks C, and all natural constant cmax:* $|\mathsf{Reg}\,(C, cmax)| \leq R(|C|, cmax + 1)$.

Thus, the size of $\mathsf{RegAut}_\exists(A)$ is bounded as follows:

Proposition 25 *For all* ECA *A with m locations and n clocks,* $\mathsf{RegAut}_\exists(A)$ *has at most* $m \times R(n, cmax + 1)$ *locations.*

Since $n \leq 2 \times |\Sigma|$ by definition of ECA, and by Theorem 7, we conclude that, in the worst case, $\mathsf{RegAut}_\exists(A)$ is smaller than the region automaton obtained by the technique of [4] given in Sect. 1 (and can be directly constructed from A).

5.3 Discussion

We close this section by a brief discussion recalling the results we have obtained. We have considered, both for the classical and the weak semantics, two possible definitions for the region automaton depending on the definition of the transition relation (either existential or universal). The following corollary summarises the main results of this section regarding the accepted languages:

Corollary 26 *For all* $\mathsf{ECA}\, A$*:*

$$\mathsf{L}(\mathsf{RegAut}_{\exists}(A)) = \mathsf{L}(\mathsf{wRegAut}_{\forall}(A)) = \mathsf{L}(\mathsf{wRegAut}_{\exists}(A)) \\ = \\ \mathsf{Untime}(\mathsf{L}(A))$$

Moreover, there exists an EPA *with three clocks s.t.:*

$$\mathsf{L}(\mathsf{RegAut}_{\forall}(A)) \subsetneq \mathsf{Untime}(\mathsf{L}(A))$$

Thus, $\mathsf{RegAut}_{\exists}(A)$ seems the most adequate tool so far to reason about the untimed language of A, as its definition relies on the classical semantics of ECA, and on the classical Alur-Dill definition of regions. Finally, Proposition 25 shows that the definition of $\mathsf{RegAut}_{\exists}(A)$ induces a potentially smaller automaton than the one that can be obtained by first translating the ECA into a TA, as was originally suggested in [4].

Yet, for practical purposes, it is well-known that zones are a more efficient data structure than regions to implement algorithms analysing real-time models such as timed automata. The purpose of the next section is to introduce an ECA version of the classical zone data structure.

6 Zones and event-clocks

In the setting of TA, the *zone data structure* [18] has been introduced as an effective way to improve the running time and memory consumption of on-the-fly algorithms for checking emptiness. In this section, we *adapt* this notion to the framework of ECA, and discuss forward and backward analysis algorithms. Roughly speaking, a *zone* is a symbolic representation for a set of clock valuations that are defined by constraints of the form $x - y \prec c$, where x, y are clocks, $\prec$ is either $<$ or $\leq$, and c is an integer constant. Keeping the difference between clock values makes sense in the setting of timed automata as all the clocks have always real values and the difference between two clock values is an invariant over the elapsing of time. To adapt the notion of zone to ECA, we need to overcome two difficulties. First, as already pointed out when discussing the notion of region, prophecy and history clocks evolve in different directions with time elapsing. Hence, it is not always the case that if $v(x) - v(y) = c$ then $(v + t)(x) - (v + t)(y) = c$ for all t (for instance if x is a prophecy clock and y a history clock). However, the *sum* of clocks of different types is now an invariant, so event clock zones must be definable, either by constraints of the form $x - y \prec c$, if x and y are both history or both prophecy clocks, or by constraints of the form $x + y \prec c$ otherwise, yielding the use of some kind of octagons. Formally, we introduce the notion of event-zone as follows.

Definition 27 For a set C of clocks over an alphabet Σ, an *event-zone* is a subset of $\mathcal{V}(C)$ that is defined by a conjunction of constraints of the form $x = \bot$; $x \sim c$; $x_1 - x_2 \sim c$ if $x_1, x_2 \in \mathbb{H}_{\Sigma}$ or $x_1, x_2 \in \mathbb{P}_{\Sigma}$; and $x_1 + x_2 \sim c$ if either $x_1 \in \mathbb{H}_{\Sigma}$ and $x_2 \in \mathbb{P}_{\Sigma}$ or $x_1 \in \mathbb{P}_{\Sigma}$ and $x_2 \in \mathbb{H}_{\Sigma}$, with $x, x_1, x_2 \in C$, $\sim \in \{\leq, \geq, <, >\}$ and $c \in \mathbb{Z}$.

6.1 Event-clock Difference Bound Matrices

In the context of TA, Difference Bound Matrices (DBMs for short) have been introduced to represent and manipulate zones [6,18]. It has been shown that DBMs stay an efficient encoding even for octagons [23], a class our event-zones fall into. However, in this particular case, we are interested in a sub-case of octagons, in which the type of constraints considered (sum or difference) depend only on the type of variables involved: when the clocks are of the same type, we are interested in the difference of their values, otherwise we take into account their sum. Hence, to adapt DBMs to event clocks, we need to be able to (i) encode constraints of the form $x + y \prec c$ and of the form $x' - y' \prec c$, depending on the types of x, y, x' and y', (ii) encode constraints of the form $x = \bot$, and (iii) encode the fact that a variable is not constrained by the zone. Indeed, in a DBM, this is encoded by the pair of constraints $x \geq 0$ and $x < +\infty$. This is not sound in our case since $0 \leq x < +\infty$ implies that $x \neq \bot$. Thus, we introduce a special symbol ? to denote the absence of constraint.

Formally, an EDBM M, on the set of clocks $C = \{x_1, \ldots, x_n\}$, is a $(n + 1)$ square matrix of elements from $\big(\mathbb{Z} \times \{<, \leq\}\big) \cup \{(\infty, <), (\bot, =), (?, =)\}$ s.t. for all $0 \leq i, j, \leq n$: $M_{i,j} = (\bot, =)$ implies $i = 0$ or $j = 0$ (i.e., $\bot$ can only appear in the first position of a row or column). As in the case of DBMs, we assume that the extra clock x_0 is always equal to zero. Thus, in particular, a constraint of the form $x_i = \bot$ will be encoded with either $M_{i,0} = (\bot, =)$ or $M_{0,i} = (\bot, =)$.

Moreover, since prophecy clocks decrease with time evolving, while history clocks increase, prophecy clocks are encoded by their *opposite value* in the matrix. As we will see, this allows the EDBM to naturally encode *differences* of pairs of variables, when the two clocks are of the same type, and *sums* of variables when the two clocks are of different types.

An EDBM represents an event-zone as follows. Let

$$x^{\pm} = \begin{cases} x & \text{if } x \in \mathbb{H}_{\Sigma} \\ -x & \text{otherwise} \end{cases}$$

Then, each element $(m_{i,j}, \prec_{i,j})$ (with $m_{i,j} \notin \{?, \bot\}$) of an EDBM represents the constraint $x_i^{\pm} - x_j^{\pm} \prec_{i,j} m_{i,j}$. Thus, intuitively, the constraint ranges over the *value* of the history clocks and the *opposite value* of the prophecy clocks, which is coherent with the fact that prophecy clocks decrease with time elapsing. Finally, the special symbol ? encodes the absence of constraint. In particular, for the elements $M_{i,0}$, $M_{0,i}$, it indicates that the clock x_i can take any real value, *or* the $\bot$ value.

Formally, let v be a valuation on the set of clocks $C = \{x_1, \ldots, x_n\}$, let M be an EDBM on C. Then for all $0 \leq i, j \leq n$, we say that v satisfies $M_{i,j} = (m_{i,j}, \prec_{i,j})$ (denoted $v \models M_{i,j}$) iff:

(1) either $m_{i,j} = ?$
(2) or $i = 0$ and $m_{i,j} = v(x_j) = \bot$
(3) or $j = 0$ and $m_{i,j} = v(x_i) = \bot$
(4) or $m_{i,j} \notin \{?, \bot\}$ and $v^{\pm}(x_i) - v^{\pm}(x_j) \prec_{i,j} m_{i,j}$ (where we let $\bot + c = c + \bot = \bot - c = c - \bot = \bot$ for all c).

Then, assuming that $v(x_0) = 0$ for any valuation v, we define the set of valuations represented by M as:

$$[\![M]\!] = \{v \mid \forall 0 \leq i, j \leq n : v \models M_{i,j}\}$$

When $[\![M]\!] = \varnothing$, we say that M is *empty*.

Example 28 As an example, consider the three EDBMs in Fig. 7, They all represent $x_1 = \bot \wedge 0 < x_3 - x_4 < 1 \wedge x_2 + x_4 \leq 2$ (where x_1, x_2 are prophecy clocks, and x_3, x_4 are history clocks).

Fig. 7 Three EDBMs representing the event-zone $x_1 = \bot \wedge 0 < x_3 - x_4 < 1 \wedge x_2 + x_4 \leq 2$ (where x_1, x_2 are prophecy clocks, and x_3, x_4 are history clocks). M^2 is in normal form, but not canonical. M^3 is canonical. Differences between two successive EDBMs are highlighted. Variables names have been displayed to enhance readability

$$M^1 = \begin{array}{cccccc} & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & (0,\leq) & (?,=) & (?,=) & (?,=) & (?,=) \\ x_1 & (\bot,=) & (?,=) & (?,=) & (?,=) & (?,=) \\ x_2 & (0,\leq) & (?,=) & (0,\leq) & (?,=) & (?,=) \\ x_3 & (?,=) & (?,=) & (?,=) & (0,\leq) & (1,<) \\ x_4 & (?,=) & (?,=) & (2,\leq) & (0,<) & (0,\leq) \end{array}$$

$$M^2 = \begin{array}{cccccc} & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & (0,\leq) & \boldsymbol{(\bot,=)} & \boldsymbol{(\infty,<)} & \boldsymbol{(0,\leq)} & \boldsymbol{(0,\leq)} \\ x_1 & (\bot,=) & (?,=) & (?,=) & (?,=) & (?,=) \\ x_2 & (0,\leq) & (?,=) & (0,\leq) & \boldsymbol{(0,\leq)} & \boldsymbol{(0,\leq)} \\ x_3 & \boldsymbol{(\infty,<)} & (?,=) & \boldsymbol{(\infty,<)} & (0,\leq) & (1,<) \\ x_4 & \boldsymbol{(\infty,<)} & (?,=) & (2,\leq) & (0,<) & (0,\leq) \end{array}$$

$$M^3 = \begin{array}{cccccc} & x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & (0,\leq) & (\bot,=) & (\infty,<) & (0,\leq) & (0,\leq) \\ x_1 & (\bot,=) & (?,=) & (?,=) & (?,=) & (?,=) \\ x_2 & (0,\leq) & (?,=) & (0,\leq) & (0,\leq) & (0,\leq) \\ x_3 & (\infty,<) & (?,=) & \boldsymbol{(3,<)} & (0,\leq) & (1,<) \\ x_4 & (\infty,<) & (?,=) & (2,\leq) & (0,<) & (0,\leq) \end{array}$$

Canonical and normal form EDBMs As in the case of DBMs, we need to rely on a *canonical* notion of EDBM, in which the constraints represented in the matrix are as tight as possible. To show how to effectively turn a given EDBM M into a canonical EDBM M' such that $[\![M]\!] = [\![M']\!]$, we first introduce the notion of *normal form* EDBM. An EDBM is in normal form if the constraints on the clocks that are not required to have a value in $\mathbb{R}$ are as tight as possible.

Definition 29 (*normal form EDBM*) An EDBM M is in *normal form* if

(i) $M_{i,0} = (\bot, =)$ **iff** $M_{0,i} = (\bot, =)$ for all $1 \leq i \leq n$
(ii) $M_{i,0} = (?, =)$ **iff** $M_{0,i} = (?, =)$ for all $1 \leq i \leq n$
(iii) **if** $M_{i,0} \in \{(\bot, =), (?, =)\}$ for some $1 \leq i \leq n$, **then** $M_{i,j} = M_{j,i} = (?, =)$, for all $1 \leq j \leq n$.

Given a normal form EDBM M, it is then easy to extract the set of clocks constrained by M to have their value in $\mathbb{R}$: we let $C_M(\mathbb{R}) = \{x_i \in C \mid \text{for all } v \in [\![M]\!], v(x_i) \in \mathbb{R}\}$ be the set of such clocks. When M is in normal form, $C_M(\mathbb{R})$ has the following property: for all $x_i \in C$, $x_i \in C_M(\mathbb{R})$ if and only if $M_{i,0} \notin \{(\bot, =), (?, =)\}$. We further let $I_M(\mathbb{R}) = \{0\} \cup \{1 \leq i \leq n \mid x_i \in C_M(\mathbb{R})\}$ be the set of indices of the clocks that are in $C_M(\mathbb{R})$, augmented with 0. Then, we denote by $M[C_M(\mathbb{R})] = (M_{i,j})_{i,j \in I_M(\mathbb{R})}$ the EDBM M restricted to the clocks constrained to have their value in $\mathbb{R}$. Actually, $M[C_M(\mathbb{R})]$ is a classical DBM.

Definition 30 (*canonical EDBM*) An EDBM M is *canonical* if it is in normal form and if the DBM $M[C_M(\mathbb{R})]$ is canonical [18], i.e., if the constraints represented in the matrix are as tight as possible.

To canonically represent the empty zone, we select a particular EDBM $M_\varnothing$ s.t. $[\![M_\varnothing]\!] = \varnothing$.

Example 31 For example, consider again the three EDBMs in Fig. 7. M^1 is not in normal form (and thus neither canonical), because, for instance, $M^1_{1,0} = (\bot, =)$, but $M^1_{0,1} = (?, =)$. M^2 is in normal form, but still not canonical. Indeed, in $M^2[C_M(\mathbb{R})]$, we find the constraints $x_4 + x_2 \leq 2$ and $x_3 - x_4 < 1$. This implies that $x_3 + x_2 < 3$, hence the constraint in $M^2_{3,2}$ can be strengthened. Doing so yields M^3, which is now canonical.

Then, given an EDBM M, Algorithm 1 allows to compute a canonical EDBM M' s.t. $[\![M]\!] = [\![M']\!]$. This algorithm relies on the function `DBMCanonical(`M,S`)`, where M is an $(\ell + 1) \times (\ell + 1)$ EDBM, and $S \subseteq \{0, \ldots, \ell\}$. `DBMCanonical(`$M$,$S$`)` applies the classical algorithm to obtain canonical DBMs [18] on the DBM obtained by projecting away from M all the lines and columns $i \notin S$. Algorithm 1 proceeds in three steps. In the first loop, we look for lines (resp. columns) i s.t. $M_{i,0}$ (resp. $M_{0,i}$) is $(\bot, =)$, meaning that there is a constraint imposing that $x_i = \bot$. In this case, the corresponding $M_{0,i}$ (resp. $M_{i,0}$) must be equal to $(\bot, =)$ too, and all the other elements in the ith line and column must contain $(?, =)$. If we find a j s.t. $M_{i,j} \neq (?, =)$ or $M_{j,i} \neq (?, =)$, then the zone is empty, and we return $M_\varnothing$. Then, in the second loop, the algorithm looks for lines (resp. columns) i with the first element equal to $(?, =)$ but containing a constraint of the form $(c, \prec)$, which imposes that the variable i must be different from $\bot$. We record this information by replacing the $(?, =)$ in $M_{i,0}$ (resp. $M_{0,i}$) by the weakest possible constraint that forces x_i to have a value different from $\bot$. This is either $(0, \leq)$ or $(\infty, <)$, depending on the type of x_i and is taken care by the `SetCst()` function. At this point, the EDBM M is in normal form, and the set S contains the indices of all variables that are constrained to be real, and hence is exactly $C_M(\mathbb{R})$. The algorithm finishes by calling the algorithm to obtain canonical DBMs. Remark, in particular, that the algorithm returns $M_\varnothing$ iff M is empty which also provides us with a test for EDBM emptiness (see next section).

```
EDBMCanonical($M$) begin
    Let $S = \{0\}$ ;
    foreach $1 \leq i \leq n$ s.t. $M_{i,0} = (\bot, =)$ or $M_{0,i} = (\bot, =)$ do
        if $M_{0,i} \notin \{(?, =), (\bot, =)\}$ or $M_{i,0} \notin \{(?, =), (\bot, =)\}$ or $\exists 1 \leq j \leq n$ s.t. $M_{i,j} \neq (?, =)$ or
        $M_{j,i} \neq (?, =)$ then
            return $M_\varnothing$;
        $M_{i,0} \leftarrow (\bot, =)$ ; $M_{0,i} \leftarrow (\bot, =)$ ;
    foreach $0 \leq i, j \leq n$ s.t. $M_{i,j} \notin \{(?, =), (\bot, =)\}$ do
        $S \leftarrow S \cup \{i, j\}$ ;
    foreach $i, j \in S$ do
        SetCst($M_{i,j}$) ;
    $M' \leftarrow$ DBMCanonical($M$,$S$) ;
    if $M' =$ Empty then return $M_\varnothing$ ;
    return $M'$ ;
end

SetCst($M_{i,j}$) begin
    if $M_{i,j} = (?, =)$ then
        if $(x_i \in \mathbb{P}_\Sigma$ or $x_i = x_0)$ and $(x_j \in \mathbb{H}_\Sigma$ or $x_j = x_0)$ then $M_{i,j} \leftarrow (0, \leq)$ ;
        else $M_{i,j} \leftarrow (\infty, <)$ ;
end
```

Algorithm 1: An algorithm to turn EDBMs into canonical form

Proposition 32 *For all EDBM M,* `EDBMCanonical(`M`)` *returns a canonical EDBM M' s.t. $[\![M']\!] = [\![M]\!]$.*

Proof Let M be an EDBM. We first consider the case where $M' = M_\varnothing$ has been returned in line 1. In this case, the conditional of the loop in line 1 guarantees that either $M_{i,0} = (\bot, =)$, or that $M_{0,i} = (\bot, =)$. Then, the conditional in line 1 guarantees that we are in one of the four following cases: 1. $M_{i,0} = (\bot, =)$ and $M_{0,i} \notin \{(?, =), (\bot, =)\}$; or 2. $M_{0,i} = (\bot, =)$ and $M_{i,0} \notin \{(?, =), (\bot, =)\}$; or 3. $M_{i,0} = (\bot, =)$ and there is $1 \le j \le n$ s.t. $M_{i,j} \neq (?, =)$; or 4. $M_{0,i} = (\bot, =)$ and there is $1 \le j \le n$ s.t. $M_{j,i} \neq (?, =)$. In the two first cases, the constraints $v(x_i) = \bot$ and $0 \le v(x_i) < \infty$ must hold for all $v \in [\![M]\!]$. In the third case, either $M_{i,j} = (\bot, =)$, and no valuation v can satisfy either of the four cases of the definition of $v \models M$; or $M_{i,j} = (m, \prec) \notin \{(\bot, =), (?, =)\}$ and all valuations v must satisfy $v^{\pm}(x_i) - v^{\pm}(x_j) \prec m$ with $v(x_i) = \bot$. The fourth case is symmetric to the third case with the roles of i and j swapped. In all cases, we conclude that no valuation v is s.t. $v \models M$, hence $[\![M]\!] = \varnothing$. Thus, $[\![M']\!] = [\![M_\varnothing]\!] = [\![M]\!]$ and $M_\varnothing$ is a canonical EDBM by definition.

Otherwise, let M^7 and M^{11} be the modified matrix M before respectively line 1 and line 1 and M' be the EDBM returned by `EDBMCanonical(`M`)`. The matrix M^7 is such that

($\star$) for all $1 \le i \le n$, $M^7_{i,0} = (\bot, =)$ if and only if $M^7_{0,i} = (\bot, =)$
($\star\star$) if $M^7_{i,0} = (\bot, =)$ for some $1 \le i \le n$, then $M^7_{i,j} = M^7_{j,i} = (?, =)$ for all $1 \le j \le n$.

Moreover, it is easy to see that the **for** loop of line 1 does not modify the semantics of the EDBM, i.e. $[\![M]\!] = [\![M^7]\!]$.

The for loop in starting in line 7 builds a set S of indices, s.t.:

$$S = \{0 \le i \le n \mid \exists 0 \le j \le n, M^7_{i,j} \notin \{(?, =), (\bot, =)\}\} \vee M^7_{j,i} \notin \{(?, =), (\bot, =)\}$$

Thus, before executing line 1, $S = I_M(\mathbb{R})$. Then, for all $i, j \in S$, if $M^7_{i,j} = (?, =)$ or $M^7_{j,i} = (?, =)$, then M^7 can safely be replaced by $(<, \infty)$ which is the weakest constraint that forces $v^{\pm}(x_i) - v^{\pm}(x_j)$ to be a real value. This is what the `SetCst(`$M_{i,j}$`)` procedure does, and thus $[\![M^{11}]\!] = [\![M^7]\!] = [\![M]\!]$.

Then, the algorithm calls `DBMCanonical` in line 1, which is the classical normalisation operator on DBM. In the case where `DBMCanonical` finds a set of constraints that are not satisfiable, it returns '`Empty`', we have that $[\![M^{11}]\!] = \varnothing$, and `EDBMCanonical(`$M$`)` returns $M_\varnothing$. Otherwise, `DBMCanonical` returns an EDBM M' s.t. $[\![M']\!] = [\![M^{11}]\!] = [\![M]\!]$ and s.t. $M[C_M(\mathbb{R})]$ is a canonical DBM. Thus, by definition of *canonical EDBM*, we only need to show that M^{11} is in normal form to establish that M' is canonical and conclude the proof.

To show that M^{11} is in normal form, we first observe that, for all $i \notin S$, for all $0 \le j \le n$, $M^{11}_{i,j} = M^7_{i,j}$. Then,

- Let $1 \le i \le n$ and assume that $M^{11}_{0,i} = (\bot, =)$. Then $i \notin S$ and $M^{11}_{0,i} = M^7_{0,i}$. By ($\star$), $M^7_{i,0} = (\bot, =) = M^{11}_{i,0}$. Symmetrically, if $M^{11}_{i,0} = (\bot, =)$ then $M^{11}_{0,i} = (\bot, =)$.
- Let $1 \le i \le n$, and assume that $M^{11}_{0,i} = (?, =)$. Again, $i \notin S$, and we have $M^{11}_{i,0} = M^7_{i,0} \in \{(\bot, =), (?, =)\}$. By ($\star$), we deduce that $M^7_{i,0} = (?, =)$. Again, by symmetry, if $M^{11}_{i,0} = (?, =)$ then $M^{11}_{0,i} = (?, =)$.
- Let $1 \le i \le n$ such that $M^{11}_{i,0} = (\bot, =)$. Then $i \notin S$ and for all $1 \le j \le n$, $M^{11}_{i,j} = M^6_{i,j}$ and $M^{11}_{j,i} = M^7_{j,i}$. By ($\star\star$), $M^{11}_{i,j} = M^{11}_{j,i} = (?, =)$.
- Let $1 \le i \le n$ such that $M^{11}_{i,0} = (?, =)$. Then $i \notin S$, and, by definition of S, and by definition of an EDBM, for all $1 \le j \le n$, $M^{11}_{i,j} = M^{11}_{j,i} = (?, =)$. □

6.2 Symbolic operations on zones

In this section, we discuss *symbolic operations on zones*, that is, we show how basic operations on zones can be directly computed on their EDBM representations. Most of the operations described here are extensions of the classical operations on DBMs, in order to cope with the $(?, =)$ and $(\bot, =)$ elements.

Intuitively, the six basic operations we need to perform on event-zones are: to compute the *future* and the *past* of an event-zone ; to compute the *intersection* of two event-zone s; to *project away* a given event-clock from the set of valuations of an event-zone (we call the operation the *release* of the event clock); to test inclusion of an event-zone into another; and, finally, to test whether an event-zone is *empty*. These operations will form the basis of the symbolic algorithms for testing language emptiness of ECA, that we discuss in Sect. 7.

Let us now formalise those operations. For that purpose, we define an ordering $\leq$ on EDBM elements. We let $(m, \prec) \leq (m', \prec')$ iff one of the following holds: either (i) $m' = ?$; or (ii) $m, m' \in \mathbb{Z} \cup \{\infty\}$ and $m < m'$; or (iii) $m = m'$ and either $\prec=\prec'$ or $\prec'=\leq$. We also extend the $+$ operator to EDBM elements as follows. Let $(m, \prec)$ and $(m', \prec')$ be two EDBM elements s.t. $m, m' \notin \{?, \bot, \infty\}$. Then, $(m, \prec) + (m', \prec') = (m + m', \prec'')$, where $\prec''=\leq$ iff $\prec=\prec'=\leq$, and $\infty + c = c + \infty = \infty$ for all $c \in \mathbb{Z} \cup \{\infty\}$. In the case where either $m \in \{?, \bot\}$ or $m' \in \{?, \bot\}$, the sum is not defined.

Future For an event-zone Z, we let

$$\overrightarrow{Z} = \{v \in \mathscr{V}(\mathbb{C}_\Sigma) \mid \exists v' \in Z, t \in \mathbb{R}^{\geq 0} : v = v' + t\}$$

This operation is computed symbolically as follows. Let M be a canonical EDBM on n clocks. If $M = M_\varnothing$, we let $\overrightarrow{M} = M_\varnothing$. Otherwise, we let $\overrightarrow{M}$ be s.t.:

$$\overrightarrow{M}_{i,j} = \begin{cases} (0, \leq) & \text{if } M_{i,j} \notin \{(\bot, =), (?, =)\},\ j = 0 \text{ and } x_i \in \mathbb{P}_\Sigma \\ (\infty, <) & \text{if } M_{i,j} \notin \{(\bot, =), (?, =)\},\ j = 0 \text{ and } x_i \in \mathbb{H}_\Sigma \\ M_{i,j} & \text{otherwise} \end{cases}$$

Proposition 33 *Let M be a canonical EDBM. Then, $\overrightarrow{[\![M]\!]} = [\![\overrightarrow{M}]\!]$.*

Proof In the case where $M = M_\varnothing$ the proof is trivial. Otherwise, M is non-empty, since it is canonical. We assume that M is an EDBM on set of clocks $C = \{x_1, \ldots, x_n\}$, that for all $0 \leq i, j \leq n$: $M_{i,j} = (m_{i,j}, \prec_{i,j})$ and that $\overrightarrow{M}_{i,j} = (m'_{i,j}, \prec'_{i,j})$. It is easy to see that any $v \in \overrightarrow{[\![M]\!]}$ satisfies the constraints of $[\![\overrightarrow{M}]\!]$. Thus, $\overrightarrow{[\![M]\!]} \subseteq [\![\overrightarrow{M}]\!]$.

Consider now a valuation $v \in [\![\overrightarrow{M}]\!]$. We need to find a delay $t \in \mathbb{R}^{\geq 0}$ such that there exists $v_M \in [\![M]\!]$ such that $v_M + t = v$. This amounts to solving the following system of inequalities:

$$\begin{cases} -m_{i0} - v(x_i) \prec_{i0} t \prec_{0i} m_{0i} - v(x_i) & \text{for all } x_i \in \mathbb{P}_\Sigma \cap C \text{ such that } m_{0i} \notin \{\bot, ?\} \\ v(x_i) - m_{i0} \prec_{i0} t \prec_{0i} v(x_i) + m_{0i} & \text{for all } x_i \in \mathbb{H}_\Sigma \cap C \text{ such that } m_{0i} \notin \{\bot, ?\} \\ 0 \leq t \end{cases}$$

with the convention that $\infty + c = \infty - c = \infty$ and that $-\infty + c = -\infty - c = -\infty$ for all $c \in \mathbb{N}$. We show that the set of solutions is not empty, i.e. that all inequalities are pairwise coherent.

Since for all $x_i \in \mathbb{P}_\Sigma \cap C$, $(m'_{0i}, \prec'_{0i}) = (m_{0i}, \prec_{0i})$, we know that $v(x_i) \prec_{0i} m_{0i}$ and since for all $x_i \in \mathbb{H}_\Sigma \cap C$ $(m'_{0i}, \prec'_{0i}) = (m_{0i}, \prec_{0i})$, we also know that $-m_{0i} \prec_{0i} v(x_i)$. Then, none of the inequalities forces t to be negative.

Let now x_i, x_j be two prophecy clocks s.t. $m_{0,i} \notin \{\bot, ?\}$ and $m_{0,j} \notin \{\bot, ?\}$. For all $v_M \in [\![M]\!]$, $-m_{i0} \prec_{i0} v_M(x_i) \prec_{0i} m_{0i}$, and $-m_{j0} \prec_{j0} v_M(x_j) \prec_{0j} m_{0j}$, then $-m_{i0} - m_{0j} \prec_1 v_M(x_i) - v_M(x_j) \prec_2 m_{0i} + m_{j0}$, where $\prec_1 = \leq$ iff ($\prec_{i0} = \leq$ and $\prec_{0j} = \leq$), and $\prec_2 = \leq$ iff ($\prec_{0i} = \leq$ and $\prec_{j0} = \leq$). Since M is canonical, $(m_{ji}, \prec_{ji}) \leq (m_{0i} + m_{j0}, \prec_2)$ and $(m_{ij}, \prec_{ij}) \leq (m_{i0} + m_{0j}, \prec_1)$. Since $(m'_{ij}, \prec'_{ij}) = (m_{ij}, \prec_{ij})$ and $(m'_{ji}, \prec'_{ji}) = (m_{ji}, \prec_{ji})$, we deduce that $-m_{i0} - m_{0j} \prec_1 v(x_i) - v(x_j) \prec_2 m_{0i} + m_{j0}$. Hence, $-m_{i0} - v(x_i) \prec_1 m_{0j} - v(x_j)$ and $-m_{j0} - v(x_j) \prec_2 m_{0i} - v(x_i)$. Then the constraints on t deduced from x_i and x_j are coherent. With the same arguments, we obtain that the constraints on t deduced from $x_i, x_j \in \mathbb{H}_\Sigma \cap C$ are coherent too.

Consider now $x_i \in \mathbb{P}_\Sigma \cap C$ and $x_j \in \mathbb{H}_\Sigma \cap C$. Then again, since any valuation v_M in $[\![M]\!]$ satisfies $-m_{i0} - m_{0j} \prec_1 v_M(x_i) + v_M(x_j) \prec_2 m_{0i} + m_{j0}$, so does v, and one can deduce that $-m_{i0} - v(x_i) \prec_1 v(x_j) + m_{0j}$ and $v(x_j) - m_{j0} \prec_2 m_{0i} - v(x_i)$ and hence that the constraints on t derived from $x_i \in \mathbb{P}_\Sigma \cap C$ and $x_j \in \mathbb{H}_\Sigma \cap C$ are coherent.

Then, the set of solutions of the inequalities is not empty. Let t be such a solution and v_M be the valuation s.t. $v_M(x) = v(x) + t$ for any $x \in \mathbb{P}_\Sigma \cap C$ and $v_M(x) = v(x) - t$ for all $x \in \mathbb{H}_\Sigma \cap C$. Such a valuation exists, and is in $[\![M]\!]$ by construction. Then, since $v = v_M + t$ with $v_M \in [\![M]\!]$ and some $t \in \mathbb{R}^{\geq 0}$ we deduce that $v \in \overrightarrow{[\![M]\!]}$ and $[\![\overrightarrow{M}]\!] \subseteq \overrightarrow{[\![M]\!]}$. □

Past For an event-zone Z, we let

$$\overleftarrow{Z} = \{v \in \mathscr{V}(\mathbb{C}_\Sigma) \mid \exists t \in \mathbb{R}^{\geq 0} : v + t \in Z\}$$

This operation is symmetrical to the *future* operation. Let M be a canonical EDBM on n clocks. If $M = M_\varnothing$, we let $\overleftarrow{M} = M_\varnothing$. Otherwise, we let $\overleftarrow{M}$ be s.t. for all i, j:

$$\overleftarrow{M}_{i,j} = \begin{cases} (\infty, <) & \text{if } M_{i,j} \notin \{(\bot, =), (?, =)\}, i = 0 \text{ and } x_j \in \mathbb{P}_\Sigma \\ (0, \leq) & \text{if } M_{i,j} \notin \{(\bot, =), (?, =)\}, i = 0 \text{ and } x_j \in \mathbb{H}_\Sigma \\ M_{i,j} & \text{otherwise} \end{cases}$$

Proposition 34 *Let M be a canonical EDBM. Then, $\overleftarrow{[\![M]\!]} = [\![\overleftarrow{M}]\!]$.*

Proof As prophecy and history clocks evolve in opposite directions, the arguments of the proof for $\overrightarrow{M}$ can be adapted. □

Intersection Let M_1 and M_2 be two *normal form* EDBMs on n clocks (remark that, for the intersection *we do not require* the EDBMs to be canonical). We consider several cases. If $M^1 = M_\varnothing$ or $M^2 = M_\varnothing$, we let $M^1 \cap M^2 = M_\varnothing$. If there are $0 \leq i, j \leq n$ s.t. $M^1_{i,j} \npreceq M^2_{i,j}$ and $M^2_{i,j} \npreceq M^1_{i,j}$, we let $M^1 \cap M^2 = M_\varnothing$ too. Otherwise, we let $M^1 \cap M^2$ be the EDBM M' s.t for all i, j: $M'_{i,j} = min(M^1_{i,j}, M^2_{i,j})$.

Proposition 35 *Let M^1 and M^2 be two EDBMs in normal form, and on the same set of clocks. Then, $M^1 \cap M^2$ is a normal form EDBM s.t. $[\![M^1 \cap M^2]\!] = [\![M^1]\!] \cap [\![M^2]\!]$.*

Proof In the case where $M^1 = M_\varnothing$ or $M^2 = M_\varnothing$ the proof is trivial. Otherwise, we first consider the case where there are $0 \leq i, j \leq n$ s.t. $M^1_{i,j} \npreceq M^2_{i,j}$ and $M^2_{i,j} \npreceq M^1_{i,j}$. By definition of $\preceq$, this implies that either $M^1_{i,j}$ or $M^2_{i,j}$ is equal to $(\bot, =)$, and that the other constraint is of the form $(c, \prec)$, with $c \in \mathbb{R}^{\geq 0} \cup \{\infty\}$. Then, clearly $[\![M^1]\!] \cap [\![M^2]\!] = \varnothing$ and thus $[\![M^1]\!] \cap [\![M^2]\!] = [\![M_\varnothing]\!] = [\![M^1 \cap M^2]\!]$.

Thus, let us assume that for all $0 \leq i, j \leq n$, $\min\{M^1_{i,j}, M^2_{i,j}\} = M'_{i,j}$ is defined. We first show that M' is a normal form EDBM:

- for $1 \leq i \leq n$, $M'_{i,0} = (\bot, =)$ iff $M^1_{i,0} = (\bot, =)$ or $M^2_{i,0} = (\bot, =)$ iff, since M^1 and M^2 are in normal form, $M^1_{0,i} = (\bot, =)$ or $M^2_{0,i} = (\bot, =)$ iff, since $M'_{0,i}$ is defined, $M'_{0,i} = (\bot, =)$.
- for $1 \leq i \leq n$, $M'_{i,0} = (?, =)$ iff $M^1_{i,0} = (?, =)$ and $M^2_{i,0} = (?, =)$, iff $M^1_{0,i} = (?, =)$ and $M^2_{0,i} = (?, =)$ iff $M'_{0,i} = (?, =)$.
- let $1 \leq i \leq n$ such that $M'_{i,0} \in \{(\bot, =), (?, =)\}$. Then, $M^1_{i,0}, M^2_{i,0} \in \{(\bot, =), (?, =)\}$ and, since M^1 and M^2 are in normal form, $M^1_{i,j} = M^1_{j,i} = M^2_{i,j} = M^2_{j,i} = (?, =)$ for all $1 \leq j \leq n$. Then $M'_{i,j} = M'_{j,i} = (?, =)$ for all $1 \leq j \leq n$.

Let v be a valuation. Observe that, by definition of the ordering $\leq$ on EDBM constraints:

$$\big(v \models (m_1, \prec_1) \text{ and } v \models (m_2, \prec_2)\big) \quad \textbf{iff} \quad v \models \min\big\{(m_1, \prec_1), (m_2, \prec_2)\big\}.$$

By definition of $M^1 \cap M^2$, we conclude that $[\![M^1]\!] \cap [\![M^2]\!] = [\![M^1 \cap M^2]\!]$. □

Release Let Z be an event-zone , and let x be a clock of Z. Then, we let:

$$\mathsf{rel}_x(Z) = \{v[x := d] \mid v \in Z, d \in \mathbb{R}^{\geq 0} \cup \{\bot\}\}$$

Let M be a canonical EDBM on n clocks. and let x be one of those event clocks. In the case where $M = M_\varnothing$, we let $\mathsf{rel}_x(M) = M_\varnothing$. Otherwise, we let $\mathsf{rel}_x(M)$ be the EDBM s.t. for all i, j:

$$\mathsf{rel}_x(M)_{i,j} = \begin{cases} M_{i,j} & \text{if } x_i \neq x \text{ and } x_j \neq x \\ (?, =) & \text{otherwise} \end{cases}$$

Proposition 36 *Let M be a canonical EDBM on set of clocks C, and let $x \in C$. Then,* $\mathsf{rel}_x([\![M]\!]) = [\![\mathsf{rel}_x(M)]\!]$

Proof In the case where $M = M_\varnothing$ the proof is trivial. Otherwise, M being canonical, $[\![M]\!] \neq \varnothing$. Let us assume that x is the clock of index k in C. We first examine the case where $M_{k,0} = (?, =)$, then $\mathsf{rel}_x(M) = M$ since M is canonical. Since x is already unconstrained in M, we have $\mathsf{rel}_x([\![M]\!]) = [\![M]\!]$. Hence $\mathsf{rel}_x([\![M]\!]) = [\![M]\!] = [\![\mathsf{rel}_x(M)]\!]$.

Otherwise, let us assume that $C = \{x_1, \ldots, x_n\}$, that for all $0 \leq i, j \leq n$: $M_{i,j} = (m_{ij}, \prec_{ij})$ and consider some $v \in \mathsf{rel}_x([\![M]\!])$. By definition, there is some $v' \in [\![M]\!]$, such that $v'(y) = v(y)$, for all clock $y \neq x$ in C. Since $v' \models M_{i,j}$ for all $0 \leq i, j \leq n$, we deduce that $v \models M_{i,j}$ for all $i, j \neq k$, and, since $v \models (?, =)$, $v \models \mathsf{rel}_x(M)$. Thus, $\mathsf{rel}_x([\![M]\!]) \subseteq [\![\mathsf{rel}_x(M)]\!]$.

Conversely, let $v \in [\![\mathsf{rel}_x(M)]\!]$. Since we have ruled out the case where $M_{k,0} = (?, =)$, either $M_{k,0} = (\bot, =)$ or $M_{k,0} = (m, \prec)$, with $m \in \mathbb{R}^{\geq 0} \cup \{\infty\}$. In the first case, let v' be the valuation s.t. $v'(x) = \bot$ and for all $y \neq x$: $v'(y) = v(y)$. Clearly $v' \in [\![M]\!]$ and thus, $v \in \mathsf{rel}_x[\![M]\!]$. In the second case, let v' be a valuation that is a solution of the following set of inequalities if x is a history clock:

$$\begin{array}{rcll} v'(y) & = & v(y) & \text{for all } y \neq x \\ -m_{0k} \prec_{0k} & v'(x) & \prec_{k0}\, m_{k0} & \\ -m_{jk} \prec_{jk} & v'(x) - v'(x_j) & \prec_{kj}\, m_{kj} & \text{for all } x_j \in (\mathbb{H}_\Sigma \cap C) \setminus \{x\} \\ -m_{jk} \prec_{jk} & v'(x) + v'(x_j) & \prec_{kj}\, m_{kj} & \text{for all } x_j \in (\mathbb{P}_\Sigma \cap C) \setminus \{x\} \end{array}$$

or a solution of the following set of inequalities if x is a prophecy clock:

$$\begin{array}{rcll} v'(y) & = & v(y) & \text{for all } y \neq x \\ -m_{k0} \prec_{k0} & v'(x) & \prec_{0k} m_{0k} & \\ -m_{kj} \prec_{kj} & v'(x) - v'(x_j) & \prec_{jk} m_{jk} & \text{for all } x_j \in (\mathbb{P}_\Sigma \cap C) \setminus \{x\} \\ -m_{jk} \prec_{jk} & v'(x) + v'(x_j) & \prec_{kj} m_{kj} & \text{for all } x_j \in (\mathbb{H}_\Sigma \cap C) \setminus \{x\} \end{array}$$

assuming as usual that $\bot + c = c + \bot = \bot - c = c - \bot = \bot$.

Since M is canonical, such a v' exists (the arguments are similar than the ones used in the proof of Proposition 33), and it is in $[\![M]\!]$ by construction. Hence v is in $\mathsf{rel}_x([\![M]\!])$. We conclude that $[\![\mathsf{rel}_x(M)]\!] \subseteq \mathsf{rel}_x([\![M]\!])$. □

Inclusion Let M_1 and M_2 be two canonical EDBMs on n clocks. Then, we let $M^1 \subseteq M^2$ iff $M^1_{i,j} \leq M^2_{i,j}$ for all $0 \leq i, j \leq n$.

Proposition 37 *Let M^1 and M^2 be two canonical EDBMs on the same set of clocks. Then, $[\![M^1]\!] \subseteq [\![M^2]\!]$ iff $M^1 \subseteq M^2$.*

Proof The proof stems from the fact that $[\![M^1]\!] \subseteq [\![M^2]\!]$ **iff** $[\![M^1]\!] \cap [\![M^2]\!] = [\![M^1]\!]$ **iff** $[\![M^1 \cap M^2]\!] = [\![M^1]\!]$ **iff**, $\min(M^1_{i,j}, M^2_{i,j}) = M_{i,j}$ for all $0 \leq i, j \leq n$ (by Proposition 35). □

Emptiness For the sake of completeness, we recall here that Algorithm 1, that turns an EDBM M into an equivalent *canonical* EDBM, can be used to test whether M is empty. Indeed, by Proposition 32, $[\![M]\!] = \varnothing$ iff `EDBMCanonical`(M) $= M_\varnothing$. Moreover, by further inspecting the execution of `EDBMCanonical`(M) when M is *in normal form*, we remark that $[\![M]\!] = \varnothing$ iff the DBM obtained from M by projecting away the clocks that are not constrained to be real is empty, as formalized by the next proposition. Then:

Proposition 38 *Let $M = (m_{ij}, \prec_{ij})$ be a* normal form *EDBM on n clocks. Then: $[\![M]\!] = \varnothing$ iff $[\![M[C_M(\mathbb{R})]]\!] = \varnothing$ iff there is a* negative cycle *in M, i.e. a sequence $i_1, i_2, \ldots, i_m$ of indices s.t.*

(1) for all $1 \leq k \leq m$: $0 \leq i_k \leq n$ and
(2) for all $1 \leq k < m$: $(m_{i_k i_{k+1}}, \prec_{i_k i_{k+1}}) \notin \{(?, =), (\bot, =)\}$ and
(3) for all $1 \leq k < \ell < m$: $i_k \neq i_\ell$ and
(4) $i_1 = i_m$ and
(5) $(m_{i_1 i_2}, \prec_{i_1 i_2}) + (m_{i_2 i_3}, \prec_{i_2 i_3}) + \cdots + (m_{i_{m-1} i_m}, \prec_{i_{m-1} i_m}) < (0, \leq)$.

Proof The proof relies on Proposition 32 establishing the correctness of Algorithm 1. By Proposition 32, $[\![M]\!]$ is empty iff `EDBMCanonical`(M) returns $M_\varnothing$. In Algorithm 1, this can only happen either (i) because there are i and j s.t. $M_{i,0} = (\bot, =) \vee M_{0,i} = (\bot, =)$ and $M_{i,j} \neq (?, =) \vee M_{j,i} \neq (?, =)$, or (ii) because $M[C_M(\mathbb{R})]$ is empty. The first case is not possible when M is in *normal form*, hence $[\![M]\!] = \varnothing$ iff $[\![M[C_M(\mathbb{R})]]\!] = \varnothing$. By classical results on DBMS [18], the latter holds iff $M[C_M(\mathbb{R})]$ contains a negative cycle. As $M[C_M(\mathbb{R})]$ is a DBM (and thus does not contain elements in $\{(?, =), (\bot, =)\}$), obtained from M by projecting away some lines and columns, the cycle in $M[C_M(\mathbb{R})]$ respects points 1 through 5 and is present in M too. Hence the proposition. □

6.3 Representing regions by EDBM

Since any region is also a zone, by definition, each region r must admit an EDBM representation M^r. Let us show how M^r can be computed from the definition of r. This definition of M^r will be useful, in particular in Sect. 7, when we will reason on different *widening operators*.

Let Σ be a finite alphabet, *cmax* be a natural constant, and let r be a region from $\mathsf{Reg}(\mathbb{C}_\Sigma, cmax)$. To each clock $x \in \mathbb{C}_\Sigma$, we associate we associate $r(x)$, which is either $\bot$ when $v(x) = \bot$ for all valuation v in r, or the interval containing all values $v(x)$ s.t. $v \in r$. Formally:

$$r(x) = \begin{cases} \bot & \text{if } v(x) = \bot \text{ for all } v \in r \\ ]c; c+1[& \text{if } 0 \le c < cmax \text{ and } v(x) \in]c, c+1[\text{ forall } v \in r \\ [c; c] & \text{if } 0 \le c \le cmax \text{ and } v(x) = c \text{ for all } v \in r \\ ]cmax; \infty[& \text{if } v(x) > cmax \text{ forall } v \in r \end{cases}$$

where $c \in \mathbb{N}$.

By the definition of regions, $r(x)$ is well-defined for all r and x. Moreover, to alleviate notations, we rely on the $\sqsubseteq_r$ ordering that compares the fractional parts of the clocks different from $\bot$. We recall that given a valuation v, for all $x \in \mathbb{H}_\Sigma$, $\langle v(x) \rangle = \lceil v(x) \rceil - v(x)$, and for all $x \in \mathbb{P}_\Sigma$, $\langle v(x) \rangle = v(x) - \lfloor v(x) \rfloor$, where $\lfloor v(x) \rfloor$ and $\lceil v(x) \rceil$ denote respectively the floor and ceiling of $v(x)$. Formally, for a region r, and for all pairs of clocks x_i, x_j s.t. $r(x_i) \neq \bot$ and $r(x_j) \neq \bot$, we let $x_i \sqsubseteq_r x_j$ iff $\langle v(x_i) \rangle \le \langle v(x_j) \rangle$ for all $v \in r$. Again, by the definition of regions $\sqsubseteq_r$ is well-defined. We further let $x_i \sqsubset_r x_j$ iff $x_i \sqsubseteq_r x_j$ but $x_j \not\sqsubseteq_r x_j$, and $x_i \equiv_r x_j$ iff $x_i \sqsubseteq_r x_j$ and $x_j \sqsubseteq_r x_i$.

With these notations, we can now define M^r from r. We assume $\mathbb{C}_\Sigma = \{x_1, \ldots, x_n\}$ and let $M^r = (M^r_{ij})_{0 \le i,j \le n}$. We first define the elements in the first line and column of M^r. For all $1 \le i \le n$:

(1) If $x_i \in \mathbb{H}_\Sigma$, then:

$$M^r_{i0} = \begin{cases} (\bot, =) & \text{if } r(x_i) = \bot \\ (c+1, <) & \text{if } r(x_i) =]c; c+1[\\ (c, \le) & \text{if } r(x_i) = [c; c] \\ (\infty, <) & \text{if } r(x_i) =]cmax; \infty[\end{cases}$$

$$M^r_{0i} = \begin{cases} (\bot, =) & \text{if } r(x_i) = \bot \\ (-c, <) & \text{if } r(x_i) =]c; c+1[\\ (-c, \le) & \text{if } r(x_i) = [c; c] \\ (-cmax, <) & \text{if } r(x_i) =]cmax; \infty[\end{cases}$$

(2) Otherwise, $x_i \in \mathbb{P}_\Sigma$. Then:

$$M^r_{i0} = \begin{cases} (\bot, =) & \text{if } r(x_i) = \bot \\ (-c, <) & \text{if } r(x_i) =]c; c+1[\\ (-c, \le) & \text{if } r(x_i) = [c; c] \\ (-cmax, <) & \text{if } r(x_i) =]cmax; \infty[\end{cases}$$

$$M^r_{0i} = \begin{cases} (\bot, =) & \text{if } r(x_i) = \bot \\ (c+1, <) & \text{if } r(x_i) =]c; c+1[\\ (c, \le) & \text{if } r(x_i) = [c; c] \\ (\infty, <) & \text{if } r(x_i) =]cmax; \infty[. \end{cases}$$

Next, we define the elements of the form $(m^r_{ij}; \prec^r_{ij})$ for $1 \leq i, j \leq n$. We consider several cases:

(1) If $r(x_i) = \bot$ or $r(x_j) = \bot$, then:

$$M^r_{ij} = (?, =)$$

(2) If $\{x_i, x_j\} \subseteq \mathbb{H}_\Sigma$, $r(x_i) =]c, c+1[$ and $r(x_j) =]d, d+1[$, then:

$$M^r_{ij} = \begin{cases} (c - d, <) & \text{if } x_j \sqsubset_r x_i \\ (c - d + 1, <) & \text{if } x_i \sqsubset_r x_j \\ (c - d, \leq) & \text{if } x_i \equiv_r x_j \end{cases}$$

(3) If $\{x_i, x_j\} \subseteq \mathbb{P}_\Sigma$, $r(x_i) =]c, c+1[$ and $r(x_j) =]d, d+1[$, then:

$$M^r_{ij} = \begin{cases} (d - c, <) & \text{if } x_j \sqsubset_r x_i \\ (d - c + 1, <) & \text{if } x_i \sqsubset_r x_j \\ (d - c, \leq) & \text{if } x_i \equiv_r x_j \end{cases}$$

(4) If $x_i \in \mathbb{H}_\Sigma$, $x_j \in \mathbb{P}_\Sigma$, $r(x_i) =]c, c+1[$ and $r(x_j) =]d, d+1[$, then:

$$M^r_{ij} = \begin{cases} (c + d + 2, <) & \text{if } x_i \sqsubset_r x_j \\ (c + d + 1, <) & \text{if } x_j \sqsubset_r x_i \\ (c + d + 1, \leq) & \text{if } x_i \equiv_r x_j \end{cases}$$

(5) If $x_i \in \mathbb{P}_\Sigma$, $x_j \in \mathbb{H}_\Sigma$, $r(x_i) =]c, c+1[$ and $r(x_j) =]d, d+1[$, then:

$$M^r_{ij} = \begin{cases} (-c - d, <) & \text{if } x_i \sqsubset_r x_j \\ (-c - d - 1, <) & \text{if } x_j \sqsubset_r x_i \\ (-c - d - 1, \leq) & \text{if } x_i \equiv_r x_j \end{cases}$$

(6) If $\{x_i, x_j\} \subseteq \mathbb{H}_\Sigma$, then

$$M^r_{ij} = \begin{cases} (c - d, <) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) =]d; d+1[\\ (c - d + 1, <) & \text{if } r(x_i) =]c; c+1[\text{ and } r(x_j) = [d; d] \\ (c - d, \leq) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) = [d; d] \end{cases}$$

(7) If $\{x_i, x_j\} \subseteq \mathbb{P}_\Sigma$, then

$$M^r_{ij} = \begin{cases} (d - c + 1, <) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) =]d; d+1[\\ (d - c, <) & \text{if } r(x_i) =]c; c+1[\text{ and } r(x_j) = [d; d] \\ (c - d, \leq) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) = [d; d] \end{cases}$$

(8) If $x_i \in \mathbb{H}_\Sigma$ and $x_j \in \mathbb{P}_\Sigma$, then

$$M^r_{ij} = \begin{cases} (c + d + 1, <) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) =]d; d+1[\\ & \quad \text{or } r(x_i) =]c; c+1[\text{ and } r(x_j) = [d; d] \\ (c + d, \leq) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) = [d; d] \end{cases}$$

(9) If $x_i, \in \mathbb{P}_\Sigma$ and $x_j \in \mathbb{H}_\Sigma$, then

$$M^r_{ij} = \begin{cases} (-c-d, <) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) =]d; d+1[\\ & \quad \text{or } r(x_i) =]c; c+1[\text{ and } r(x_j) = [d; d] \\ (-c-d, \leq) & \text{if } r(x_i) = [c; c] \text{ and } r(x_j) = [d; d] \end{cases}$$

(10) In all other cases:

$$M^r_{ij} = (\infty, <)$$

Let us briefly justify the values given at points 4 and 5. Assume $x_i \in \mathbb{H}_\Sigma$, $x_j \in \mathbb{P}_\Sigma$, $r(x_i) =]c; c+1[$ and $r(x_j) =]d; d+1[$. From those constraints, we already deduce that, for any $v \in r$: $v(x_i) + v(x_j) \in]c+d; c+d+2[$. Hence, with these only constraints, we would set $(m^r_{ij}, \prec^r_{ij})$ to $(c+d+2, <)$ and $(m^r_{ji}, \prec^r_{ij})$ to $(-c-d, <)$. However, with an additional constraint on the fractional parts of the clocks, we are able to further strengthen those constraints. For instance, assume that $x_j \sqsubset_r x_i$. This means that for all $v \in r$: $\langle v(x_j) \rangle < \langle v(x_i) \rangle$. However, since $x_i \in \mathbb{H}_\Sigma$ and $r(x_i) =]c; c+1[$, for all $v \in r$: $\langle v(x_i) \rangle = c+1-v(x_i)$. Similarly, for all $v \in r$: $\langle v(x_j) \rangle = v(x_j) - d$. Hence:

$$\begin{aligned} x_j \sqsubset_r x_i & \text{ iff } \langle v(x_j) \rangle < \langle v(x_i) \rangle \\ & \text{ iff } v(x_j) - d < c + 1 - v(x_i) \\ & \text{ iff } v(x_i) + v(x_j) < c + d + 1 \end{aligned}$$

Thus, the fact that $x_j \sqsubset_r x_i$ allows us to strengthen the upper bound on $x_i + x_j$, from $c+d+2$ to $c+d+1$. Thus, we let $m^r_{ij} = (c+d+1, <)$ (and still $m^r_{ji} = (-c-d, <)$, as this has no influence on the lower bound). The other cases follow similarly. With these observations in mind, the next proposition follows immediately:

Proposition 39 *For all region $r \in \mathsf{Reg}(\mathbb{C}_\Sigma, cmax)$: $[\![M^r]\!] = r$.*

Proof By straightforward inspection of the definition of M^r. □

Remark that, by definition, M^r is a *normal form EDBM*, but it might not be *canonical*. In particular, when one of the clock is greater than *cmax* in the region r, any related element of M^r is set to $(\infty, <)$. However, Algorithm 1 can be used to turn M^r into a canonical EDBM.

7 Forward and backward zone-based analysis

Now that we have at our disposal the event-zone data structure, which has the potential to compactly represent large sets of valuations, let us introduce two basic algorithms that rely on event-zones . Those algorithms are meant to decide the *language emptiness* problem, and can easily be adapted to other instances of the reachability problem.

The two basic solutions to solve this problem are known as the *forward* and *backward* algorithms. The *forward* algorithm consists in computing iteratively all the configurations that are reachable in a given $\mathsf{ECA}A$ from its set of initial configurations. Then, $\mathsf{L}(A) \neq \varnothing$ iff the computed set contains an accepting configuration. Symmetrically, the *backward algorithm* computes iteratively all the configurations that can reach an accepting configuration. Then, $\mathsf{L}(A) \neq \varnothing$ iff the computed set contains an initial configuration. Since those sets are potentially infinite, both algorithms use the event-zone data structure to store and manipulate the computed states.

When applied to *timed automata*, it is well-known that the *backward algorithm* always terminates, but that the *forward algorithm* might not terminate on certain instances. To ensure termination of the forward algorithm, a *widening operator* [15], often denoted Approx_k, and working directly on zones, has been proposed (and later implemented in tools such as UppAal or Kronos [5,12]). The correctness of Approx_k has been established later by Bouyer, only for TA that do not contain diagonal constraints [9,10]. In the setting of ECA, however, we show hereunder that neither algorithm terminates in general. Intuitively, the non-termination of the backward algorithm stems from the fact that *prophecy clocks* act somehow like regular TA clocks that count backwards. Hence, the phenomena preventing termination of the forward algorithm can be recreated in an execution of the backward algorithm, using prophecy clocks.

To overcome this difficulty, we consider the application of Approx_k to the ECA case, and observe that it does not preserve, *as is*, the correctness of the forward and backward algorithms. We propose a variant of this operator, and show its correctness, using a proof that follows the lines of Bouyer's original proof technique [10].

7.1 Forward and backward algorithms

Let us first introduce the forward and backward algorithms. From now on, we consider an ECA$A = \langle Q, q_i, \Sigma, C, \delta, \alpha\rangle$. We also let

$$\mathsf{Post}\left((q, v)\right) = \{(q', v') \mid \exists t, a : (q, v) \xrightarrow{t,a} (q', v')\}$$

and

$$\mathsf{Pre}\left((q, v)\right) = \{(q', v') \mid \exists t, a : (q', v') \xrightarrow{t,a} (q, v)\}$$

and we extend those operators to sets of states in the natural way. Moreover, given a set of valuations Z and a location q, we abuse notations and denote by (q, Z) the set $\{(q, v) \mid v \in Z\}$. Also, we let $\mathsf{Post}^*\left((q, Z)\right) = \bigcup_{n\in\mathbb{N}} \mathsf{Post}^n\left((q, Z)\right)$ and $\mathsf{Pre}^*\left((q, Z)\right) = \bigcup_{n\in\mathbb{N}} \mathsf{Pre}^n\left((q, Z)\right)$, where $\mathsf{Post}^0\left((q, Z)\right) = (q, Z)$ and $\mathsf{Post}^n\left((q, Z)\right) = \mathsf{Post}\left(\mathsf{Post}^{n-1}\left((q, Z)\right)\right)$, and similarly for $\mathsf{Pre}^n\left((q, Z)\right)$. The Post and Pre operators are sufficient to solve language emptiness for ECA: from the definitions, we immediately deduce:

Lemma 40 (adapted from [4, Lemma 1]) *Let* $A = \langle Q, q_i, \Sigma, C, \delta, \alpha\rangle$ *be an* ECA, *let* $I = \{(q_i, v) \mid v \text{ is initial}\}$, *and let* $\overline{\alpha} = \{(q, v) \mid q \in \alpha \text{ and } v \text{ is final}\}$. *Then:*

$$\mathsf{Post}^*\left(I\right) \cap \overline{\alpha} \neq \varnothing \; iff \; \mathsf{Pre}^*\left(\overline{\alpha}\right) \cap I \neq \varnothing \; iff \; \mathsf{L}(A) \neq \varnothing.$$

Let us show how to compute the $\mathsf{Post}_e\left(\right)$ and $\mathsf{Pre}_e\left(\right)$ operators by means of the symbolic operations on event-zones that we have introduced in Sect. 6.2. First, for all zones Z and clock x, we let $(Z)\left[x := 0\right]$ be the zone obtained by resetting x, i.e.: $(Z)\left[x := 0\right] = \mathsf{rel}_x(Z) \cap (x = 0)$. Then, given a location q, an event-zone Z on $\mathbb{C}_\Sigma$, and an edge $e = (q, a, \psi, q') \in \delta$, we let:

$$\mathsf{Post}_e\left((q_1, Z)\right) = \begin{cases} \left(q', \left(\mathsf{rel}_{\overrightarrow{x_a}}(\overrightarrow{Z} \cap (\overrightarrow{x_a} = 0)) \cap \psi\right)\left[\overleftarrow{x_a} := 0\right]\right) & \text{if } q_1 = q \\ \varnothing & \text{otherwise} \end{cases}$$

$$\mathsf{Pre}_e\left((q_1, Z)\right) = \begin{cases} \left(q, \overleftarrow{\left(\mathsf{rel}_{\overleftarrow{x_a}}(Z \cap (\overleftarrow{x_a} = 0)) \cap \psi\right)\left[\overrightarrow{x_a} := 0\right]}\right) & \text{if } q_1 = q' \\ \varnothing & \text{otherwise} \end{cases}$$

```
Input: An ECA$A = \langle Q, q_i, \Sigma, C, \delta, \alpha\rangle$.
Output: 'Yes' iff $\mathsf{L}(A) \neq \varnothing$, 'No' otherwise.
ForwExact begin
    Let Visited = $\varnothing$ ;
    Let Wait = $\{(q_i, Z_0)\}$ ;
    while Wait $\neq \varnothing$ do
        Get and remove $(q, Z)$ from Wait ;
        if $q \in \alpha$ and $Z \subseteq Z_f$ then return Yes if there is no $(q, Z') \in$ Visited s.t. $Z \subseteq Z'$ then
            Visited := Visited $\cup \{(q, Z)\}$ ;
            Wait := Wait $\cup \{(q', Z') \in \mathsf{Post}((q, Z)) \mid Z' \neq \varnothing\}$ ;
    return No ;
end

Input: An ECA$A = \langle Q, q_i, \Sigma, C, \delta, \alpha\rangle$.
Output: 'Yes' iff $\mathsf{L}(A) \neq \varnothing$, 'No' otherwise.
BackExact begin
    Let Visited = $\varnothing$ ;
    Let Wait = $\{(q, Z_f) \mid q \in \alpha\}$ ;
    while Wait $\neq \varnothing$ do
        Get and remove $(q, Z)$ from Wait ;
        if $q = q_i$ and $Z \subseteq Z_0$ then return Yes if there is no $(q, Z') \in$ Visited s.t. $Z \subseteq Z'$ then
            Visited := Visited $\cup \{(q, Z)\}$ ;
            Wait := Wait $\cup \{(q', Z') \in \mathsf{Pre}((q, Z)) \mid Z' \neq \varnothing\}$ ;
    return No ;
end
```

Algorithm 2: The forward and backward (semi-)algorithms

Then, it is easy to check that:

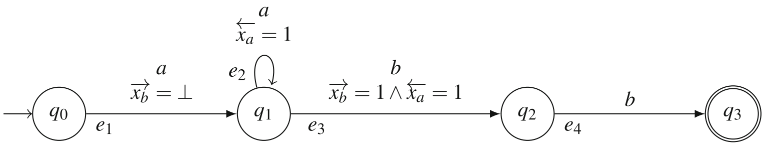
$$\mathsf{Post}((q, Z)) = \cup_{e \in \delta} \mathsf{Post}_e((q, Z))$$
$$\mathsf{Pre}((q, Z)) = \cup_{e \in \delta} \mathsf{Pre}_e((q, Z))$$

With the algorithms on EDBMs presented above, these definitions can be used to compute the Pre and Post of zones using their EDBM encoding. Remark that Pre and Post return *sets of event-zones* as these are not closed under union (like the classical zones for TA).

Let us now consider the ForwExact and BackExact algorithms to test for language emptiness of ECA, shown in Algorithm 2. In these two algorithms Z_0 denotes the zone $\bigwedge_{x \in \mathbb{H}_\Sigma} x = \bot$ containing all the possible initial valuations and Z_f denotes the zone $\bigwedge_{x \in \mathbb{P}_\Sigma} x = \bot$ representing all the possible final valuations. By Lemma 40, it is clear that ForwExact and BackExact are correct when they terminate. Unfortunately, Fig. 8 shows an ECA on which the backward algorithm does not terminate. Since history and prophecy clocks are symmetrical, this example can be adapted to define an ECA on which the forward algorithm does not terminate either. Remark that in the case of TA, when the forward analysis is not guaranteed to terminate, the backward analysis *always terminates* (the proof relies on a bisimulation argument) [11].

Proposition 41 *Neither* ForwExact *nor* BackExact *terminate in general.*

Proof We give the proof for BackExact, a similar proof for ForwExact can then be deduced by symmetry. Consider the ECA in Fig. 8, and observe that q_2 is not reachable from the initial state because of the $\overrightarrow{x_b} = \bot$ constraint on the edge e_1, and the fact that q_2 is reachable only



a
$\overleftarrow{x_a} = 1$
e_2
a
$\overrightarrow{x_b} = \bot$
b
$\overrightarrow{x_b} = 1 \wedge \overleftarrow{x_a} = 1$
b
q_0
q_1
q_2
q_3
e_1
e_3
e_4

Fig. 8 An ECA for which backward analysis does not terminate (e_1,..., e_4 are edge names)

through e_3, which is labeled by a b. Running the the backward analysis algorithm from (q_3, Z_f) yields the following sequence of computed pairs (q, Z):

(1) At the end of the first iteration, `Visited` $= \{(q_3, Z_f)\}$. Moreover, $\mathsf{Pre}_{e_4}\big((q_3, Z_f)\big) = \{(q_2, Z_1)\}$ with:

$$Z_1 = \big(\overrightarrow{x_a} = \bot \wedge \overrightarrow{x_b} \geq 0\big)$$

Hence, `Wait` $= \{(q_2, Z_1)\}$ at the end of the first iteration.

(2) At the end of the second iteration, `Visited` $= \{(q_3, Z_f), (q_2, Z_1)\}$, and (q_2, Z_1) has been picked from `Wait`. Moreover, $\mathsf{Pre}_{e_3}((q_2, Z_1)) = \{(q_1, Z_2)\}$, with:

$$Z_2 = \big(\overrightarrow{x_a} = \bot \wedge 0 \leq \overrightarrow{x_b} \leq 1 \wedge 0 \leq \overleftarrow{x_a} \leq 1 \wedge \overleftarrow{x_a} + \overrightarrow{x_b} = 1\big)$$

Hence, `Wait` $= \{(q_1, Z_2)\}$ at the end of the second iteration.

(3) During the third iteration, (q_1, Z_2) is picked from `Wait`. Then, at the end of the third iteration, `Visited` $= \{(q_3, Z_f), (q_2, Z_1), (q_1, Z_2)\}$. Moreover, $\mathsf{Pre}_{e_2}((q_1, Z_2)) = \{(q_1, Z_3)\}$ with:

$$Z_3 = \big(0 \leq \overrightarrow{x_a} \leq 1 \wedge 1 \leq \overrightarrow{x_b} \leq 2 \wedge 0 \leq \overleftarrow{x_a} \leq 1 \wedge \overleftarrow{x_a} + \overrightarrow{x_b} = 2 \wedge \overleftarrow{x_a} + \overrightarrow{x_a} \\ = 1 \wedge \overrightarrow{x_b} - \overrightarrow{x_a} = 1\big)$$

and $\mathsf{Pre}_{e_1}((q_1, Z_2)) = \{(q_0, \varnothing)\}$, because $\overrightarrow{x_b}$ is different from $\bot$ in Z_1. Hence, at the end of the third iteration, `Wait` $= \{(q_1, Z_3)\}$.

(4) During the fourth iteration, (q_1, Z_3) is picked from `Wait`. Then, at the end of the fourth iteration, `Visited` $= \{(q_3, Z_f), (q_2, Z_1), (q_1, Z_2), (q_1, Z_3)\}$. Moreover, $\mathsf{Pre}_{e_2}((q_1, Z_3)) = \{(q_1, Z_4)\}$ with:

$$Z_4 = \big(0 \leq \overrightarrow{x_a} \leq 1 \wedge 2 \leq \overrightarrow{x_b} \leq 3 \wedge 0 \leq \overleftarrow{x_a} \leq 1 \wedge \overleftarrow{x_a} + \overrightarrow{x_b} = 3 \wedge \overleftarrow{x_a} + \overrightarrow{x_a} \\ = 1 \wedge \overrightarrow{x_b} - \overrightarrow{x_a} = 2\big)$$

The arguments given for iteration 4 can be continued inductively. In general, at iteration $2 + n$ (for $n \geq 1$), the element (q_1, Z_{1+n}) is picked from `Wait`, and the computation of $\mathsf{Pre}_{e_2}((q_1, Z_{1+n}))$ yields the element (q_1, Z_{2+n}), which is inserted into `Wait` and where:

$$Z_{(2+n)} = \left(\begin{array}{c} 0 \leq \overrightarrow{x_a} \leq 1 \wedge n \leq \overrightarrow{x_b} \leq n+1 \wedge 0 \leq \overleftarrow{x_a} \leq 1 \\ \wedge \\ \overleftarrow{x_a} + \overrightarrow{x_b} = n+1 \wedge \overleftarrow{x_a} + \overrightarrow{x_a} = 1 \wedge \overrightarrow{x_b} - \overrightarrow{x_a} = n \end{array}\right)$$

The sequence $Z_3, Z_4, \ldots, Z_{2+n}, \ldots$ contains zones that are all pairwise incomparable, and that are inserted into, then taken from `Wait` one the after the other. Moreover, no elements of the form (q_0, Z) is ever produced by the algorithm. Thus, the condition of the **if** in line 2 is always fulfilled, the algorithm keeps adding new zones to `Wait`, and loops forever. □

```
Input: An ECA $A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$, and a widening operator $f$.
Output: 'Yes' iff $\mathsf{L}(A) \neq \varnothing$, 'No' otherwise.
ForwApprox$_f$ begin
    Let Visited = $\varnothing$ ; Let Wait = $f((q_i, Z_0))$ ;
    while Wait $\neq \varnothing$ do
        Get and remove $(q, Z)$ from Wait ;
        if $q \in \alpha$ and $Z \subseteq Z_f$ then return Yes if there is no $(q, Z') \in$ Visited s.t. $Z \subseteq Z'$ then
            Visited := Visited $\cup \{(q, Z)\}$ ;
            Wait := Wait $\cup \{(q', Z') \in f(\mathsf{Post}((q, Z))) \mid Z' \neq \varnothing\}$ ;
    return No ;
end

Input: An ECA $A = \langle Q, q_i, \Sigma, C, \delta, \alpha \rangle$, and a widening operator $f$.
Output: 'Yes' iff $\mathsf{L}(A) \neq \varnothing$, 'No' otherwise.
BackApprox$_f$ begin
    Let Visited = $\varnothing$ ; Let Wait = $\bigcup_{q \in \alpha} f((q, Z_f))$ ;
    while Wait $\neq \varnothing$ do
        Get and remove $(q, Z)$ from Wait ;
        if $q = q_i$ and $Z \subseteq Z_0$ then return Yes if there is no $(q, Z') \in$ Visited s.t. $Z \subseteq Z'$ then
            Visited := Visited $\cup \{(q, Z)\}$ ;
            Wait := Wait $\cup \{(q', Z') \in f(\mathsf{Pre}((q, Z))) \mid Z' \neq \varnothing\}$ ;
    return No ;
end
```

Algorithm 3: Approximation algorithms for forward and backward analysis

7.2 Widening operators

Nevertheless, we close this paper by adapting *widening operators* from the literature [8,9] and prove that they guarantee the termination of the forward algorithm (the arguments can be easily adapted to the backward case, since these cases are symmetric in the setting of ECA).

Closure by regions We first define a forward algorithm that terminates, and prove its correctness. This algorithm relies on the *closure by region* of event-zone s, a notion adapted from [9]. Let Z be an event-zone , and $\mathscr{R}$ be a set of regions, both on the set of clocks C. Then, the *closure by regions* from $\mathscr{R}$ of Z is $\text{Closure}_{\mathscr{R}}(Z) = \{r \in \mathscr{R} \mid Z \cap r \neq \varnothing\}$. Remark that, for the region equivalences we have defined above, $\text{Closure}_{\mathscr{R}}(Z)$ is a set of zones since each region is a zone. We extend $\text{Closure}_{\mathscr{R}}$ to sets S of pairs (q, Z): $\text{Closure}_{\mathscr{R}}(S) = \{(q, r) \mid r \in \text{Closure}_{\mathscr{R}}(Z), (q, Z) \in S\}$. We define now a new algorithm, parametrised by $\mathscr{R}$, $\mathsf{ForwRegion}_{\mathscr{R}} = \mathsf{ForwApprox}_{\text{Closure}_{\mathscr{R}}}$, as described in Algorithm 3. Assuming that $\mathscr{R}$ is finite, it is clear that $\mathsf{ForwRegion}_{\mathscr{R}}$ terminates. Let us now address its correctness. First, we recall that in the setting of timed automata, we have $\mathsf{Post}((q, \text{Closure}_{\mathscr{R}}(Z))) \subseteq \text{Closure}_{\mathscr{R}}(\mathsf{Post}((q, Z)))$ for all (q, Z) [9], and this property is relied upon to establish the soundness of $\mathsf{ForwRegion}_{\mathscr{R}}$. Unfortunately this is not the case in general with ECA. Indeed, consider the zone Z and the region r in Fig. 9a. Clearly, r is included in $\overrightarrow{\text{Closure}_{\mathscr{R}}(Z)}$, see (b) and (c); but r is not included in $\text{Closure}_{\mathscr{R}}(\overrightarrow{Z})$, see (d) and (e). Nevertheless, we can prove the soundness of $\text{Closure}_{\mathscr{R}}$ by relying on Theorem 23.

Theorem 42 *For all* ECA A *with alphabet* Σ *and maximal constant cmax,* $\mathsf{ForwRegion}_{\mathscr{R}}$ *answers* Yes *if and only if* $\mathsf{L}(A) \neq \varnothing$*, where* $\mathscr{R} = \mathsf{Reg}(\mathbb{C}_{\Sigma}, cmax)$.

Proof The proof relies on the correctness of the region automaton $\mathsf{RegAut}_{\exists}(A)$.

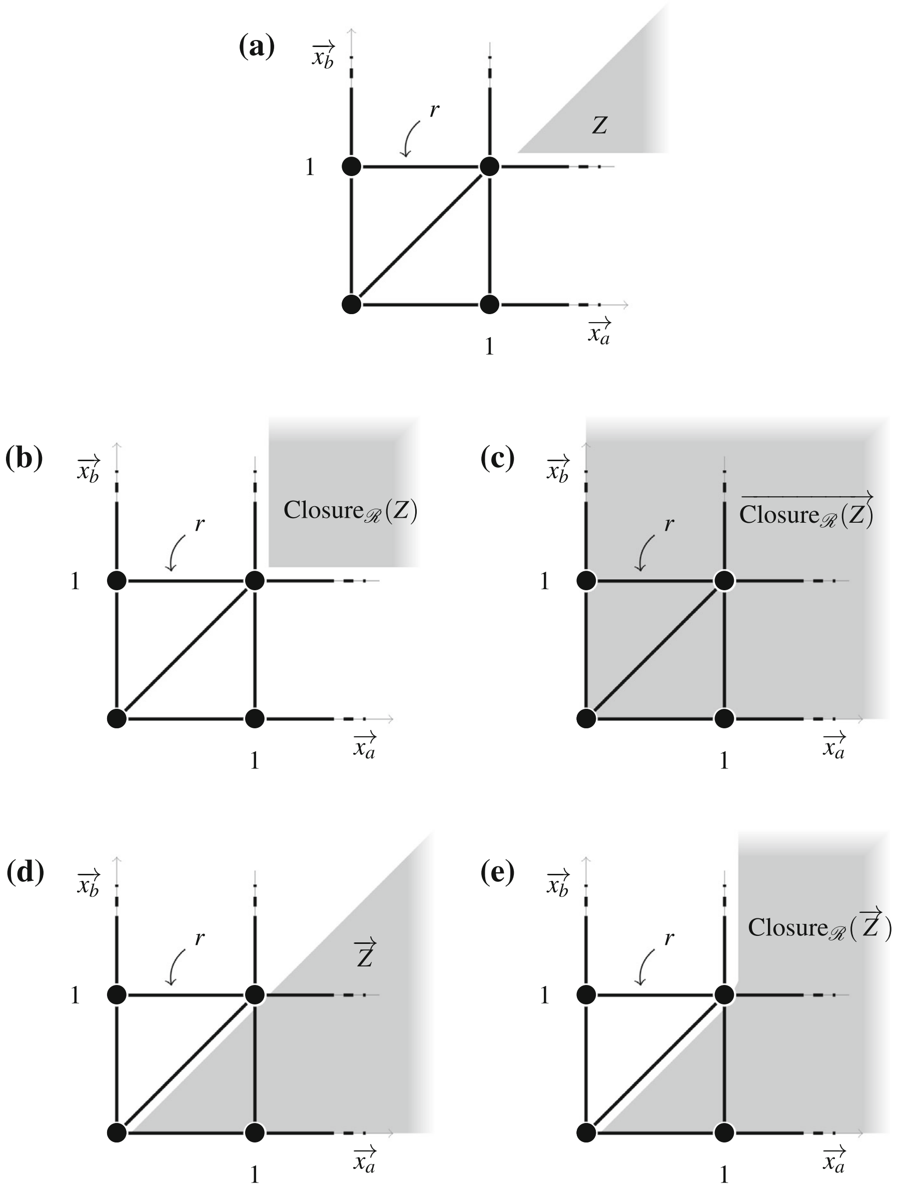


Fig. 9 The 'closure by region' operator and the elapsing of time

By construction, any pair (q, Z) visited during the algorithm is such that $Z \in \mathscr{R}$ is a region. Let ReachClosure $\subseteq Q \times \mathscr{R}$ be the set of pairs reachable by $\mathsf{ForwRegion}_{\mathscr{R}}$ and Reach $\subseteq Q \times \mathscr{R}$ be the set of states reachable from an initial state in $\mathsf{RegAut}_{\exists}(A)$. Observe that ReachClosure $\subseteq$ Reach . Indeed, let $(q_i, r) \in \text{Closure}_{\mathscr{R}}(q_i, Z_0)$. The region r is necessarily initial, and (q_i, r) is an initial state of $\mathsf{RegAut}_{\exists}(A)$ thus $(q_i, r) \in$ Reach . Consider now a pair (q, Z) in ReachClosure visited after having taken the **while** instruction i times. There exists then a pair $(q', Z') \in$ ReachClosure visited at a preceding step and such that $(q, Z) \in \text{Closure}_{\mathscr{R}}(\mathsf{Post}\left((q', Z')\right))$. By induction hypothesis, $(q', Z') \in$ Reach . By definition of $\mathsf{RegAut}_{\exists}(A)$, (q'', Z'') is a successor state of (q', Z') in $\mathsf{RegAut}_{\exists}(A)$, for all $(q'', Z'') \in \text{Closure}_{\mathscr{R}}(\mathsf{Post}\left((q', Z')\right))$. Hence (q, Z) is a reachable state in $\mathsf{RegAut}_{\exists}(A)$. Since the algorithm terminates, ReachClosure $\subseteq$ Reach .

If $\mathsf{ForwRegion}_{\mathscr{R}}$ answers `Yes`, there is a pair $(q, Z) \in$ `Wait` such that $q \in \alpha$ and $Z \subseteq Z_f$. Then, (q, Z) is a reachable state in $\mathsf{RegAut}_{\exists}(A)$ and Z is a final region. We conclude that $\mathsf{L}(\mathsf{RegAut}_{\exists}(A)) \neq \varnothing$ and, as a consequence of Theorem 23, that $\mathsf{L}(A) \neq \varnothing$.

Assume now that $\mathsf{L}(A) \neq \varnothing$. Then again, by Theorem 23, $\mathsf{L}(\mathsf{RegAut}_{\exists}(A)) \neq \varnothing$, and there is a run $(q_0, r_0) \cdots (q_n, r_n)$ of $\mathsf{RegAut}_{\exists}(A)$ such that (q_0, r_0) is an initial state and (q_n, r_n) is a final state of $\mathsf{RegAut}_{\exists}(A)$. Obviously, $q_n \in \alpha$ and $r_n \subseteq Z_f$ and it remains then to show that (q_n, r_n) can eventually be visited by $\mathsf{ForwRegion}_{\mathscr{R}}$. Observe first that any initial region r is included in Z_0. Then, any initial state (q_i, r) of $\mathsf{RegAut}_{\exists}(A)$ belongs to $\text{Closure}_{\mathscr{R}}(q_i, Z_0)$ and initially $(q_0, r_0) \in$ `Wait`. Assume now that (q_i, r_i) is in `Wait` at some point, for some $i < n$. If (q_i, r_i) is not removed from `Wait`, it means that the algorithm has ended and answered `Yes`. Assume now that is is removed from `Wait`. Since r_i is not final, $\text{Closure}_{\mathscr{R}}(\mathsf{Post}\left((q_i, r_i)\right))$ is added to `Wait`. Again, it is clear that $(q_{i+1}, r_{i+1}) \in \text{Closure}_{\mathscr{R}}(\mathsf{Post}\left((q_i, r_i)\right))$, hence (q_{i+1}, r_{i+1}) will be added to `Wait`. Then, either (q_n, r_n) will be removed from `Wait` and, since the run exhibited is accepting, $q_n \in \alpha$ and $r_n \subseteq Z_f$ and $\mathsf{ForwRegion}_{\mathscr{R}}$ answers `Yes`, or (q_n, r_n) is never removed from `Wait`, and again, it means that the algorithm has ended and returned `Yes`. □

k-approximation The $\text{Closure}_{\mathscr{R}}$ widening operator is mainly of theoretical interest, as it is not easily implementable. Let us now adapt the classical *k-approximation* defined for DBMs [9].

Definition 43 (*k-bounded event-zone*) Let $k \in \mathbb{N}$ be a constant and Z be an event-zone . We say that Z is *k-bounded* if Z is a conjunction of constraints of the form

$$\begin{aligned} x = \bot;\, x_1 - x_2 \sim c \quad &\text{with} \quad -\mathbf{4}\cdot k \le c \le \mathbf{4}\cdot k;\\ x \sim c \,\text{with}\; c \le\; k;\, x_1 + x_2 \sim c \quad &\text{with} \quad c \le \mathbf{4}\cdot k \end{aligned}$$

where $\sim \in \{<, \le, >, \ge, =\}$.

Remark that our definition of k-bounded event-zone *does not imply that all the constants appearing in the constraints are* $\le k$. Indeed, for constraints of the form $x_i + x_j \sim c$ or $x_i - x_j \sim c$, we tolerate values up to $\mathbf{4}\cdot k$ for c. Here again, we deviate from the definitions used in the setting of TA, and this point will be of utmost importance to prove the correctness of the algorithm we are about to present.

Since k is finite, the set of k-bounded event-zones containing a given event-zone Z is finite (and non empty).

Definition 44 (*k-approximation*) Let $k \in \mathbb{N}$ be a constant and Z be an event-zone . The *k-approximation* of Z is the intersection of the k-bounded event-zones containing Z and is denoted $\text{Approx}_k(Z)$.

We first remark that $\text{Approx}_k(Z)$ can be computed directly on the EDBM representing Z. For that purpose, we extend the Approx_k operator to EDBMs as follows:

$$\mathrm{Approx}_k(M)_{i,j} = \begin{cases} M_{i,j} & \text{if } (-k, \leq) \leq M_{i,j} \leq (k, \leq) \text{ or if } M_{i,j} \in \{(\bot, =), (?, =)\} \\ (\infty, <) & \text{if } M_{i,j} > (k, \leq) \\ (-k, <) & \text{if } M_{i,j} < (-k, \leq) \end{cases}$$

if $i = 0$ or $j = 0$.

Otherwise,

$$\mathrm{Approx}_k(M)_{i,j} = \begin{cases} M_{i,j} & \text{if } (-4 \cdot k, \leq) \leq M_{i,j} \leq (4 \cdot k, \leq), \\ & \quad \text{or if } M_{i,j} \in \{(\bot, =), (?, =)\} \\ (\infty, <) & \text{if } M_{i,j} > (4 \cdot k, \leq) \\ (-4 \cdot k, <) & \text{if } M_{i,j} < (-4 \cdot k, \leq) \end{cases}$$

The correctness of this construction is established in the following proposition:

Proposition 45 *Let $k \in \mathbb{N}$, and M a non-empty canonical EDBM. Then* $\mathrm{Approx}_k(\llbracket M \rrbracket) = \llbracket \mathrm{Approx}_k(M) \rrbracket$.

Proof By construction, $\llbracket \mathrm{Approx}_k(M) \rrbracket$ is k-bounded and includes $\llbracket M \rrbracket$. Hence, $\mathrm{Approx}_k(\llbracket M \rrbracket) \subseteq \llbracket \mathrm{Approx}_k(M) \rrbracket$. Let us now show that $\llbracket \mathrm{Approx}_k(M) \rrbracket \subseteq \mathrm{Approx}_k(\llbracket M \rrbracket)$. Let M^k be a canonical EDBM such that $\llbracket M^k \rrbracket$ is k-bounded and $\llbracket M \rrbracket \subseteq \llbracket M^k \rrbracket$, and let M^k_{approx} be the canonical EDBM such that $\llbracket M^k_{\mathrm{approx}} \rrbracket = \mathrm{Approx}_k(\llbracket M \rrbracket)$. By Proposition 37, we know that $M_{i,j} \leq M^k_{i,j}$ for all $0 \leq i, j \leq n$. Let $M' = \mathrm{Approx}_k(M)$, and let $0 < i, j \leq n$ such that $M'_{i,j} \neq M_{i,j}$.

(1) If $M_{i,j} > (4 \cdot k, \leq)$ then $M'_{i,j} = (\infty, <)$. Moreover, $(4 \cdot k, \leq) < M_{i,j}$ implies that $(4 \cdot k, \leq) < M^k_{i,j}$. Since M^k is canonical and k-bounded, it cannot be that $M^k_{i,j} = (c, \prec)$ with $c > 4 \cdot k$. Hence, either $M^k_{i,j} = (?, =)$ or $M^k_{i,j} = (\infty, <)$. In both cases, $M'_{i,j} \leq M^k_{i,j}$.

(2) If $M_{i,j} < (-4 \cdot k, \leq)$ then $M'_{i,j} = (-4 \cdot k, <)$, and $M_{i,j} < M'_{i,j} < (-4 \cdot k, \leq) \leq M^k_{i,j}$.

If there is $i = 0$ or $j = 0$ such that $M'_{i,j} \neq M_{i,j}$, then again, either $M_{i,j} > (k, \leq)$ or $M_{i,j} < (-k, \leq)$ and, as above, $M'_{i,j} \leq M^k_{i,j}$.

Then, for all $0 \leq i, j \leq n$, $M'_{i,j} \leq M^k_{i,j}$ and $\llbracket M' \rrbracket \subseteq \llbracket M^k \rrbracket$. Since this is true for any k-bounded canonical EDBM including M, it is true for the smallest one, i.e., M^k_{approx}. Hence, $\mathrm{Approx}_k(\llbracket M \rrbracket) = \llbracket \mathrm{Approx}_k(M) \rrbracket$. □

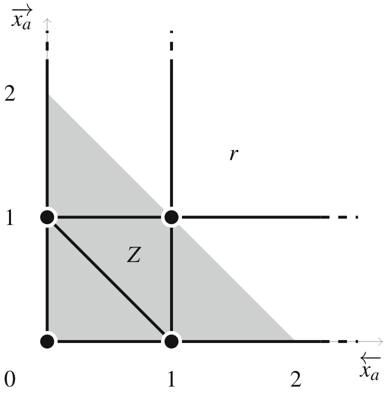
Then, we let $\mathsf{ForwKApp}_k$ be the algorithm (parametrised by k) obtained by letting $f = \mathrm{Approx}_k$ in Algorithm 3. Since there are finitely many k-bounded regions, $\mathsf{ForwKApp}_k$ always terminates. Our proof of correctness of $\mathsf{ForwKApp}_k$ follows the lines of [9], and relies on Proposition 48 hereunder, which state that, for all event-zone Z on set of clocks $\mathbb{C}_\Sigma$, we have:

$$Z \subseteq \mathrm{Approx}_{cmax}(Z) \subseteq \bigcup \mathrm{Closure}_{\mathscr{R}}(Z)$$

where $\mathscr{R} = \mathsf{Reg}\,(\mathbb{C}_\Sigma, cmax)$, and $\bigcup \mathrm{Closure}_{\mathscr{R}}(Z)$ denotes $\cup_{r \in \mathrm{Closure}_{\mathscr{R}}(Z)} r$.

Remark that this property does not hold when using the k-approximation defined for TA, which consists in replacing all constants $> k$ by ∞ in the constraints of the zone. Indeed, consider the event-zone Z defined by $\overleftarrow{x_a} + \overrightarrow{x_a} \leq 2$ in Fig. 10, together with the set of regions $\mathscr{R} = \mathsf{Reg}\left(\mathbb{C}_{\{a\}}, 1\right)$. Clearly, with our definition, $\mathrm{Approx}_k(Z) = Z$, and neither Z nor $\mathrm{Approx}_k(Z)$ intersect with the region r such that $r(\overrightarrow{x_a}) = r(\overleftarrow{x_a}) =]1; \infty[$. However,

Fig. 10 A zone Z s.t. $\text{Approx}_k(Z) = Z$



had we replaced the constraint $\overleftarrow{x_a} + \overrightarrow{x_a} \leq 2$ by $\overleftarrow{x_a} + \overrightarrow{x_a} < \infty$, we would have obtained an approximation Z' that intersects with r, and would not have been contained in $\text{Closure}_{\mathscr{R}}(Z)$. This explains why we tolerate constraints of the form $x_i + x_j \sim c$ with c greater than k, even in the k-approximation of an event-zone .

Before giving the actual proof of Proposition 48, we provide two ancillary lemmata. They provide properties of the EDBM M^r (defined in Sect. 6.3) that characterizes a region r. The first lemma states that, although M^r might not be canonical (as already discussed before), some of its constraints are *tight*. More precisely, these are the constraints M^r_{ij} where the clocks x_i and x_j take values $\leq$ *cmax* in r (possibly with $i = 0$ or $j = 0$):

Lemma 46 *Let r be a region on set of clocks $C = \{x_1, \dots, x_n\}$ and let C' be the extended set of clocks $\{x_0\} \cup C$, where, as usual, $v(x_0)$ denotes the value* 0. *Let x_i and x_j be two clocks from C' s.t. for all $v \in r$: $v(x_i) \leq cmax$ and $v(x_j) \leq cmax$. Let M^r be the EDBM associated to r, and let $i_1, i_2, \dots i_k$ be a sequence of indices s.t. for all $1 \leq \ell \leq k$: $i_\ell \in [1, n]$, and $M^r_{ij} \notin \{(?, =), (\bot, =)\}$. Then:*

$$M^r_{ij} \leq M^r_{ii_1} + M^r_{i_1 i_2} + \cdots + M^r_{i_k j}$$

Proof By a way of contradiction, assume that

$$M^r_{ii_1} + M^r_{i_1 i_2} + \cdots + M^r_{i_k j} = (c, \prec) < M^r_{ij}$$

In this case, we can safely replace M^r_{ij} by $(c, \prec)$ and obtain a new EDBM M' s.t. $[\![M']\!] = [\![M^r]\!]$ (as this replacement would be done by the canonisation algorithm, which does not modify the semantics of the EDBM). However, by inspecting the definition of M^r (when both x_i and x_j are $\leq$ *cmax*), it is easy to see that tightening M^r_{ij} yields an EDBM M' s.t. $[\![M']\!] \neq r$. Since $[\![M^r]\!] = r$, we conclude that $[\![M']\!] \neq [\![M^r]\!]$, which is a contradiction. □

The next lemma characterizes the possible values that a clock can take in a given region r. Intuitively, the lemma says that, in a region, for two clocks x_i and x_j that are below *cmax*, all the possible values of $v(x_i)^{\pm} - v(x_j)^{\pm}$ are in an interval of size 1 at most, and with bound between $-2 \cdot cmax$ and $2 \cdot cmax$. In the case where either x_i or x_j is x_0, the bounds of the interval are even tighter: they must be between $-cmax$ and *cmax*. For instance, assume that, in a given region r, x_i is a history clock in $]c; c + 1[$ and x_j is a prophecy clock in $]d; d + 1[$, with $x_i \sqsubset_r x_j$ (with $c < cmax$ and $d < cmax$). Then, for all $v \in r$: $v(x_i)^{\pm} - v(x_j)^{\pm} \in]c + d + 1; c + d + 2[$ (as can be checked with the definition of M^r).

Lemma 47 *Let r be a region in* $\mathsf{Reg}\,(\mathbb{C}_\Sigma, cmax)$. *Assume that* $\mathbb{C}_\Sigma = \{x_1, \dots, x_n\}$ *and let C be the extended set of clocks* $\mathbb{C}_\Sigma \cup \{x_0\}$, *where, as usual,* $v(x_0)$ *denotes the value* 0, *and* $r(x_0)$ *denotes the interval* $[0; 0]$. *Let x_i and x_j be two clocks from C s.t. for all $v \in r$: $v(x_i) \leq cmax$ and $v(x_j) \leq cmax$. Then, the two following points hold:*

(1) There exists $-2\cdot cmax \leq k \leq 2\cdot cmax$ *s.t.* either *for all* $v \in r$: $v(x_i)^{\pm} - v(x_j)^{\pm} \in]k, k+1[$ or *for all* $v \in r$: $v(x_i)^{\pm} - v(x_j)^{\pm} = k$

(2) ***If*** $i = 0$ *or* $j = 0$, ***then*** *there exists* $-cmax \leq k \leq cmax$ *s.t.* either *for all* $v \in r$: $v(x_i)^{\pm} - v(x_j)^{\pm} \in]k, k+1[$ or *for all* $v \in r$: $v(x_i)^{\pm} - v(x_j)^{\pm} = k$

Proof Immediate by the definition of M^r. □

We are now ready to prove that Approx_{cmax} is a correct widening operator. The proof is adapted from the arguments found in [9].

Proposition 48 *Let Z be an event-zone on set of clocks* $\mathbb{C}_\Sigma$ *and let* $\mathscr{R} = \mathsf{Reg}\,(\mathbb{C}_\Sigma, cmax)$. *Then:* $Z \subseteq \text{Approx}_{cmax}(Z) \subseteq \bigcup \text{Closure}_{\mathscr{R}}(Z)$.

Proof First observe that, by Definition 44, $Z \subseteq \text{Approx}_{cmax}(Z)$ holds trivially. Let us show that $\text{Approx}_{cmax}(Z) \subseteq \bigcup \text{Closure}_{\mathscr{R}}(Z)$. By definition, $\bigcup \text{Closure}_{\mathscr{R}}(Z)$ is a union of regions, i.e. $\bigcup \text{Closure}_{\mathscr{R}}(Z) = \cup\{r \mid r \cap Z \neq \varnothing\}$. Hence, to prove $\text{Approx}_{cmax}(Z) \subseteq \bigcup \text{Closure}_{\mathscr{R}}(Z)$, it suffices to show that, for any $r \in \mathscr{R}$: $Z \cap r = \varnothing$ implies $\text{Approx}_{cmax}(Z) \cap r = \varnothing$.

Thus, let r be a region in $\mathscr{R}$ s.t.

$$Z \cap r = \varnothing \tag{2}$$

and let us show that $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ too. We first rule out the case where $Z = \varnothing$. In this case, $\text{Approx}_{cmax}(Z) = \varnothing$ too hence, $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ for any region r. From now on, we will thus assume that:

$$Z \neq \varnothing \tag{3}$$

Let M^Z be the canonical EDBM s.t. $[\![M^Z]\!] = Z$, let M^{App} be the canonical EDBM s.t. $[\![M^{\text{App}}]\!] = \text{Approx}_{cmax}(Z)$ and let $M^r = (m^r_{ij}, \prec^r_{ij})_{0 \leq i,j \leq n}$ be the EDBM discussed in Sect. 6.3. By Proposition 39, $[\![M^r]\!] = r$. Remember also that M^r is in normal form but might be non-canonical. Since we have assumed that M^Z is canonical, the intersection $M^{\cap} = M^r \cap M^Z$ is well-defined and $[\![M^{\cap}]\!] = [\![M^r]\!] \cap [\![M^Z]\!] = Z \cap r = \varnothing$ (by Proposition 35 and by hypothesis that $Z \cap r = \varnothing$).

Using the fact that $[\![M^{\cap}]\!] = \varnothing$, we will now show that $[\![M^{\text{App}} \cap M^r]\!] = \varnothing$ too. First, assume that there are $0 \leq i < j \leq n$ s.t. neither $M^Z_{ij} \leq M^r_{ij}$ nor $M^r_{ij} \leq M^Z_{ij}$. By definition of $\leq$, this can happen only if $M^Z_{ij} = (\bot, =)$ and $M^r_{ij} \notin \{(\bot, =), (?, =)\}$, or the converse: $M^r_{ij} = (\bot, =)$ and $M^Z_{ij} \notin \{(\bot, =), (?, =)\}$. In both cases, we conclude that $M^{\text{App}}_{ij} = M^Z_{ij}$, hence neither $M^{\text{App}}_{ij} \leq M^r_{ij}$ nor $M^r_{ij} \leq M^{\text{App}}_{ij}$. Thus, by definition of the intersection of two EDBMs, $M^r \cap M^{\text{App}} = M_{\varnothing}$, and thus $\text{Approx}_{cmax}(Z) \cap r = \varnothing$. Thus, for the rest of the proof, we assume that:

$$\text{For all } 0 \leq i < j \leq n : M^Z_{ij} \leq M^r_{ij} \text{ or } M^r_{ij} \leq M^Z_{ij} \tag{4}$$

Hence, for all $0 \leq i < j \leq n$: $M^{\cap}_{ij} = \min(M_{ij}, M^r_{ij})$. Moreover, since $M^{\cap}$ is in normal form (by Proposition 35), there exists a negative cycle in $M^{\cap}$, by Proposition 38. Thus, let us assume that there exists a sequence $i_1, i_2, \dots, i_m$ of indices that form such a negative cycle, i.e. they respect all the hypothesis of Proposition 38, and, in particular:

$$M^{\cap}_{i_1 i_2} + M^{\cap}_{i_2 i_3} + \cdots + M^{\cap}_{i_{m-1} i_m} < (0, \leq) \tag{5}$$

Let us show that we can assume, wlog, that two consecutive elements in the cycle do not come from M^Z. Indeed, if there is k s.t. $M^{\cap}_{kk+1} = M^Z_{kk+1}$ and $M^{\cap}_{k+1k+2} = M^Z_{k+1k+2}$, then, we can replace $M^{\cap}_{kk+1} M^{\cap}_{k+1k+2}$ by M^Z_{kk+2} in the cycle (which yields a shorter cycle). This new cycle is negative too because $M^Z_{kk+2} \leq M^Z_{kk+1} + M^Z_{k+1k+2}$, as M^Z is canonical. By iteratively applying this construction, we obtain a negative cycle in $M^{\cap}$ that does not contain two consecutive elements from M^Z. Let us denote by $\mathscr{S}$ the set of all elements from the cycle that come from M^Z (and not from M^r), i.e.:

$$\mathscr{S} = \{M^Z_{ij} \mid \exists 0 \leq k < m : M^{\cap}_{i_k i_{k+1}} = M^Z_{ij} \text{ and } M^Z_{ij} < M^r_{ij}\} \tag{6}$$

Observe that $\mathscr{S} \neq \varnothing$. Indeed, if $\mathscr{S} = \varnothing$, then the negative cycle is present in M^r too, and $r = \varnothing$, which is not possible as each region is non-empty, by definition. Thus, the cycle must contain at least one element from M^Z (in $\mathscr{S}$).

Thus, so far, we have shown – under hypothesis (2), (3) and (4) – that $M^{\cap} = M^Z \cap M^r$ contains a negative cycle (5), (i) without two consecutive elements from M^Z and (ii) with at least one element M^Z_{ij} from M^Z s.t. $M^Z_{ij} < M^r_{ij}$. The end of the proof consists in examining all the possible cases for this cycle, and concluding, each time, that $\text{Approx}_{cmax}(Z) \cap r = \varnothing$.

We first assume that the cycle contains one element M^Z_{ij} from M^Z s.t. both clocks x_i and x_j are smaller than or equal to *cmax* **in** r. Formally, we assume that there is $M^Z_{ij} \in \mathscr{S}$ s.t. for all $v \in r$: $v(x_i) \leq cmax$ and $v(x_j) \leq cmax$.

By corollary 47, we have to consider four cases. In all these cases, the proof technique we apply is always the same: we exhibit a zone Z' s.t. (i) Z' is *cmax*-bounded; (ii) $Z \subseteq Z'$ and (iii) $Z' \cap r$. By (i) and (ii), we deduce that $\text{Approx}_{cmax}(Z) \subseteq Z'$, as Approx_{cmax} is defined as the intersection of all *cmax*-bounded zones that contain Z. Together with (iii), we conclude that $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ too.

(1) Either $i, j \neq 0$, and there is c s.t. $-2 \cdot cmax \leq c \leq 2 \cdot cmax$ and for all $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) \in]c; c+1[$. In this case, $M^r_{i,j} \leq (c+1, <)$, since $[\![M^r]\!] = r$. Since $M^Z_{ij} < M^r_{ij}$, we obtain: $M^Z_{ij} < (c+1, <)$, and thus, $M^Z_{ij} \leq (c, \leq)$. As a consequence:[6]

$$Z \subseteq Z' = (x^{\pm}_i - x^{\pm}_j \leq c)$$

Remark that, for all $v \in Z'$: $v^{\pm}(x_i) - v^{\pm}(x_j) \leq c$. On the other hand, by hypothesis, $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) > c$. Hence, $r \cap Z' = \varnothing$. Since $-2 \cdot cmax \leq c \leq 2 \cdot cmax$, by hypothesis, Z' is *cmax*-bounded.[7] Thus $\text{Approx}_{cmax}(Z) \subseteq Z'$. Hence $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ too.

(2) Or $i, j \neq 0$, and there is c s.t. $-2 \cdot cmax \leq c \leq 2 \cdot cmax$ and for all $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) = c$. This case follows the same arguments as case (1). Here, $M^r_{i,j} = (c, \leq)$. Thus $M^Z_{ij} < (c, \leq)$, hence $M^Z_{ij} \leq (c, <)$ and:

$$Z \subseteq Z' = (x^{\pm}_i - x^{\pm}_j < c)$$

[6] Recall that $x^{\pm}$ denotes x if $x \in \mathbb{H}_{\Sigma}$, and $-x$ if $x \in \mathbb{P}_{\Sigma}$.

[7] Remark that Z' would not have been *cmax*-bounded, had we relied on the widening operator as defined for TA.

Here again, we observe that Z' has no intersection with r, since for all $v \in Z'$: $v^{\pm}(x_i) - v^{\pm}(x_j) < c$, and for all $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) = c$. Moreover, Z' is *cmax*-bounded, hence, $\text{Approx}_{cmax}(Z) \subseteq Z'$. We conclude that, $r \cap \text{Approx}_{cmax}(Z) = \varnothing$ too.

(3) Or $0 \in \{i, j\}$, and there is c s.t. $-cmax \leq c \leq cmax$ and for all $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) \in]c; c+1[$. This case is treated as case (1). Remark that the zone Z' obtained here is also *cmax*-bounded, as $0 \in \{i, j\}$ but now c is in $[-cmax; cmax]$, instead of $[-2cmax; 2cmax]$ in point (1).

(4) Or $0 \in \{i, j\}$, and there is c s.t. $cmax \leq c \leq cmax$ and for all $v \in r$: $v^{\pm}(x_i) - v^{\pm}(x_j) = c$. This case is treated as case (2).

Otherwise, for all elements M^Z_{ij} of the cycle that come from M^Z, one of the clocks is larger than *cmax* **in the region**. Formally, for all $M^Z_{ij} \in \mathscr{S}$, for all $v \in r$: $v(x_i) > cmax$ or $v(x_j) > cmax$.

Let us first show that, in this case, there are at most two elements in the cycle (5) that come from M^Z, i.e. that $|\mathscr{S}| \in \{1, 2\}$ (recall that $\mathscr{S}$ is non-empty, as argued above, otherwise r would be empty, which is not possible by definition of regions). To establish this, we consider an element of the cycle $M^{\cap}_{ij} = M^Z_{ij} \in \mathscr{S}$. Remember that we have assumed that there are never two consecutive elements from M^Z in the cycle. Hence, the element that precedes M^Z_{ij} in the cycle is of the form M^r_{ki} for some $0 \leq k \leq n$, and the one that follows M^Z_{ij} in the cycle is of the form $M^r_{j\ell}$ for some $0 \leq \ell \leq n$. By hypothesis, we know that for all $v \in r$: $v(x_i) > cmax$ or $v(x_j) > cmax$ (remark that both could hold at the same time). Let us consider these two possibilities separately.

(1) Assume that for all $v \in r$: $v(x_i) > cmax$. By way of contradiction, assume that the element that precedes M^Z_{ij} in the cycle is not M^r_{0i}, i.e. that $k \neq 0$. In this case, by definition of M^r, we have $M^r_{ki} = (\infty, <)$, because $v(x_i) > cmax$ for all $v \in r$. However, in this case the cycle cannot be negative. We conclude that $k = 0$, and thus, by definition of M^r, that $M^r_{ki} = M^r_{0i} = (-cmax, <)$, and that x_i is a history clock.

(2) Assume next that for all $v \in r$: $v(x_j) > cmax$. By the same reasoning *per absurdum*, we deduce that $M^r_{j\ell} = M^r_{j0}$, i.e. that the element that follows M^Z_{ij} in the cycle is $M^r_{j0} = (-cmax, <)$, with $x_j \in \mathbb{P}_{\Sigma}$.

Thus, every time an element of the form $M^Z_{ij} \in \mathscr{S}$ appears in the cycle, it is either preceded by M^r_{0i}, or followed by M^r_{j0} (or both). As we have assumed that all the indices appearing in the cycle are different, we conclude that at most two elements from M^Z can appear in the cycle, i.e. that $|\mathscr{S}| \in \{1, 2\}$. Let us consider separately these two possibilities. Again, the proof technique we use in all the cases consists in finding a *cmax*-bounded Z' s.t. $Z \subseteq Z'$ and $r \cap Z'$, which allows to deduce that $\text{Approx}_{cmax}(Z) \subseteq Z'$, and thus that $\text{Approx}_{cmax}(Z) \cap r = \varnothing$.

(1) The first case is when $|\mathscr{S}| = 1$, i.e., there is exactly one element M^Z_{ij} from M^Z in the cycle. By hypothesis, $v(x_i) > cmax$ for all $v \in r$; or $v(x_j) > cmax$ for all $v \in r$; or both hold. Let us consider those three cases:

(a) Either $v(x_i) > cmax$ and $v(x_j) \leq cmax$ for all $v \in r$, then, as shown above, $x_i \in \mathbb{H}_{\Sigma}$, and the element that precedes M^Z_{ij} in the cycle is $M^r_{0i} = (-cmax, <)$. We further consider two cases:

(i) Either $j = 0$, then the cycle is of the form $M^r_{0i} M^Z_{i0}$. Since the cycle is negative, and since $M^r_{0i} = (-cmax, <)$: $M^Z_{i0} \leq (cmax, \leq)$. Hence,

$$Z \subseteq Z' = x_i \leq cmax$$

As Z' is *cmax*-bounded, $\mathrm{Approx}_{cmax}(Z) \subseteq Z'$ too. However, since $v(x_i) > cmax$ for all $v \in r$, $Z' \cap r = \varnothing$. We conclude that $\mathrm{Approx}_{cmax}(Z) \cap r = \varnothing$ too.

(ii) Or $j \neq 0$. In this case, the cycle is of the form:

$$M^r_{0i} M^Z_{ij} M^r_{jk_1} M^r_{k_1k_2} \cdots M^r_{k_nj} M^r_{j0}$$

However, by Lemma 46, and since $v(x_j) < cmax$ for all $v \in r$, we can consider instead the cycle

$$M^r_{0i} M^Z_{ij} M^r_{j0}$$

We consider two further cases depending on the type of x_j:

(A) If $x_j \in \mathbb{H}_\Sigma$, then, since $v(x_j) \leq cmax$ for all $v \in r$, we conclude that M^r_{j0} is either $(c+1, <)$ or $(c, \leq)$ for some $0 \leq c \leq cmax$, by definition of M^r. In both cases, $M^r_{j0} \leq (c, \leq)$. Thus:

$$\begin{aligned}(0, \leq) &> M^r_{0i} + M^Z_{ij} + M^r_{j0}\\ &\geq (-cmax, <) + M^Z_{ij} + (c, \leq)\end{aligned}$$

and thus, $M^Z_{ij} \leq (cmax - c, \leq)$, which implies that:

$$Z \subseteq Z' = x_i - x_j \leq cmax - c$$

Since $0 \leq c \leq cmax$, the zone Z' is *cmax*-bounded, and $\mathrm{Approx}_{cmax}(Z) \subseteq Z'$. On the other hand, since, $M^r_{0i} = (-cmax, <)$ and $M^r_{j0} \leq (c, \leq)$, we deduce, that, for all $v \in r$: $v(x_j) - v(x_i) < c - cmax$, hence, for all $v \in r$: $v(x_i) - v(x_j) > c - cmax$. Thus, $r \cap Z' = \varnothing$. Hence $\mathrm{Approx}_{cmax}(Z) \cap r = \varnothing$.

(B) If $x_j \in \mathbb{P}_\Sigma$, then, since $v(x_j) \leq cmax$ for all $v \in r$, we conclude that M^r_{j0} is either $(-c, <)$ or $(-c, \leq)$ for some $0 \leq c \leq cmax$, by definition of M^r. In both cases, $M^r_{j0} \leq (-c, <)$. Thus:

$$\begin{aligned}(0, \leq) &> M^r_{0i} + M^Z_{ij} + M^r_{j0}\\ &\geq (-cmax, <) + M^Z_{ij} + (-c, <)\end{aligned}$$

and thus, $M^Z_{ij} \leq (cmax + c, \leq)$, which implies that:

$$Z \subseteq Z' = x_i + x_j \leq cmax + c$$

Since $0 \leq c \leq cmax$, we have $cmax \leq cmax + c \leq 2 \cdot cmax$, and the zone Z' is *cmax*-bounded.[8]. Thus, $\mathrm{Approx}_{cmax}(Z) \subseteq Z'$ On the other hand, since, $M^r_{0i} = (-cmax, <)$ and $M^r_{j0} \leq (-c, <)$, we deduce, that, for all $v \in r$: $-v(x_j) - v(x_i) < -c - cmax$, hence, for all $v \in r$: $v(x_i) + v(x_j) > cmax + c$. Thus, $r \cap Z' = \varnothing$ and $\mathrm{Approx}_{cmax} \cap r = \varnothing$ too.

(b) Or $v(x_i) \leq cmax$ and $v(x_j) > cmax$ for all $v \in r$, then $x_j \in \mathbb{P}_\Sigma$, and the element that follows M^Z_{ij} in the cycle is M^r_{j0}. Again, we have to consider two further cases:

(i) Either $i = 0$, then the cycle is of the form $M^Z_{0j} M^r_{j0}$. This case is symmetrical to the case 1(a)i above.

[8] Ibid.

(ii) Or $i \neq 0$, and, $v(x_i) \leq cmax$ for all $v \in r$. Again, by Lemma 46, we can consider a cycle of the form $M^r_{0i} M^Z_{ij} M^r_{j0}$, and we treat the present case as case 1(a)ii above.

(c) Or both $v(x_i) > cmax$ and $v(x_j) > cmax$ for all $v \in r$. Then, $x_i \in \mathbb{H}_\Sigma$, $x_j \in \mathbb{P}_\Sigma$, and the cycle is of the form $M^r_{0i} M^Z_{ij} M^r_{j0}$. Thus, by definition of M^r, $M^r_{0i} = M^r_{j0} = (-cmax, <)$. Hence:

$$\begin{aligned}(0, \leq) &> M^r_{0i} + M^Z_{ij} + M^r_{j0}\\ &= (-cmax, <) + M^Z_{ij} + (-cmax, <)\end{aligned}$$

and we deduce that $M^Z_{ij} \leq (2 \cdot cmax, \leq)$. Thus:

$$Z \subseteq Z' = x_i + x_j \leq 2 \cdot cmax$$

However, since $v(x_i) > cmax$ and $v(x_j) > cmax$ for all $v \in r$, then: $v(x_i) + v(x_j) > cmax$ for all $v \in r$. Thus, $r \cap Z' = \varnothing$. Moreover, Z' is $cmax$-bounded.[9] We conclude that $\text{Approx}_{cmax}(Z) \subseteq Z'$, and thus that $\text{Approx}_{cmax} \cap r = \varnothing$ too.

(2) The latter case is when $|\mathscr{S}| = 2$, i.e. there are exactly two elements M^Z_{ij} and $M^Z_{k\ell}$ from M^Z in the cycle. By the arguments above, we can assume that x_i is a history clock s.t. $v(x_i) > cmax$ for all $v \in r$, and that x_ℓ is a prophecy clock s.t. $v(x_\ell) > cmax$ for all $v \in r$. Thus the cycle is of the form:

$$M^r_{0i}\ M^Z_{ij}\ \underbrace{M^r_{jx} \cdots M^r_{yk}}_{\text{from } M^r}\ M^Z_{k\ell}\ M^r_{\ell 0}$$

Moreover, still by the discussion above, we know that $v(x_j) \leq cmax$ and that $v(x_k) \leq cmax$ for all $v \in r$. Hence, by Lemma 46, we can consider the cycle

$$M^r_{0i}\ M^Z_{ij}\ M^r_{jk}\ M^Z_{k\ell}\ M^r_{\ell 0} \tag{7}$$

instead, which is still negative. Let us consider the elements appearing in this cycle. We already know that $M^r_{0i} = M^r_{\ell 0} = (-cmax, <)$. Moreover, since $v(x_j) \leq cmax$ and $v(x_k) \leq cmax$ for all $v \in r$, we have $(-2 \cdot cmax, \leq) \leq M^r_{jk} \leq (2 \cdot cmax, \leq)$, by definition of M^r. Hence:

$$\begin{aligned}(0, \leq) &> M^r_{0i} + M^Z_{ij} + M^r_{jk} + M^Z_{k\ell} + M^r_{\ell 0} && (8)\\ &\geq (-cmax, <) + M^Z_{ij} + (-2 \cdot cmax, \leq) + M^Z_{k\ell} + (-cmax, <) && (9)\\ &= (-4 \cdot cmax, <) + M^Z_{ij} + M^Z_{k\ell} && (10)\end{aligned}$$

For the rest of the proof, let us assume that $M^Z_{ij} = (c_i, \prec_i)$ and that $M^Z_{k\ell} = (c_\ell, \prec_\ell)$. Thus, from the above inequalities, we have:

$$(-4 \cdot cmax, <) + (c_i, \prec_i) + (c_\ell, \prec_\ell) < (0, \leq) \tag{11}$$

We finish the proof by considering three cases:

[9] Ibid.

(a) If $c_i \geq 0$ and $c_\ell \geq 0$, then, by (11), both c_i and c_ℓ are $\leq 4 \cdot cmax$, and thus[10] (since i, j, k and ℓ are all different from 0): $\text{Approx}_{cmax}(M^Z)_{ij} = M^z_{ij}$ and $\text{Approx}_{cmax}(M^Z)_{k\ell} = M^z_{k\ell}$. Hence, by (8) we have:

$$M^r_{0i} + \text{Approx}_{cmax}(M^Z)_{ij} + M^r_{jk} + \text{Approx}_{cmax}(M)^Z_{k\ell} + M^r_{\ell 0} < (0, \leq) \qquad (12)$$

Moreover, we know that $M^Z_{0i} \geq M^r_{0i}$, that $M^Z_{jk} \geq M^r_{jk}$ and that $M^Z_{\ell 0} \geq M^r_{\ell 0}$, as M^r_{0i}, M^r_{jk} and $M^r_{\ell 0}$ appear in the negative cycle (7). However $\text{Approx}_{cmax}(M)_{i,j} \geq M^Z_{i,j}$ for all $0 \leq i, j \leq n$, by construction. Thus, we conclude, that $\text{Approx}_{cmax}(M^Z)_{0i} \geq M^r_{0i}$, $\text{Approx}_{cmax}(M^Z)_{jk} \geq M^r_{jk}$ and $\text{Approx}_{cmax}(M^Z)_{k\ell} \geq M^r_{k\ell}$. Hence, (12) describes a negative cycle that appears in $\text{Approx}_{cmax}(M^z) \cap M^r$. As both $\text{Approx}_{cmax}(M^Z)$ and M^r are normal form EDBMs, we conclude that $\text{Approx}_{cmax}(Z) \cap r = \varnothing$, by Proposition 38.

(b) If $c_i < 0$, then, by definition of M^r, this can be achieved only when $x_j \in \mathbb{H}_\Sigma$. Moreover, $c_i < 0$ implies that $M^Z_{ij} \leq (0, <)$. Hence:

$$Z \subseteq Z' = x_i - x_j < 0$$

Observe that Z' is *cmax*-bounded, hence $\text{Approx}_{cmax}(Z) \subseteq Z'$ too. However, $v(x_i) > cmax$ and $v(x_j) \leq cmax$ for all $v \in r$ by hypothesis. Hence, for all $v \in r$: $v(x_i) - v(x_j) \geq 0$. Thus, $Z' \cap r = \varnothing$, hence $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ too.

(c) If $c_\ell < 0$, we treat this case symmetrically to the former one. By definition of M^r, this can be achieved only when $x_k \in \mathbb{P}_\Sigma$. Moreover, $c_\ell < 0$ implies that $M^Z_{k\ell} \leq (0, <)$. Hence:

$$Z \subseteq Z' = -x_k + x_\ell < 0$$

Observe that Z' is *cmax*-bounded, hence $\text{Approx}_{cmax}(Z) \subseteq Z'$ too. However, $v(x_\ell) > cmax$ and $v(x_k) \leq cmax$ for all $v \in r$ by hypothesis. Hence, for all $v \in r$: $v(x_\ell) - v(x_k) \geq 0$. Thus, $Z' \cap r = \varnothing$, hence $\text{Approx}_{cmax}(Z) \cap r = \varnothing$ too.

□

Thanks to Proposition 48 and Theorem 42, we can now conclude:

Theorem 49 *For all* ECA*A with maximal constant cmax,* ForwKApp_{cmax} *terminates and answers* `Yes` **iff** $\text{L}(A) \neq \varnothing$.

Indeed, ForwRegion_{cmax} and ForwKApp_{cmax} are compute over-approximations of the set of valuations reachable by the exact algorithm. Since k-approximation is finer than the closure by regions, correctness of ForwRegion_{cmax} allows to conclude to correctness of ForwKApp_{cmax}.

8 Conclusion

Event-clock automata have been introduced [4] as an alternative model to timed automata. The original paper on ECA, as well as a series of subsequent works, have relied on the classical *region construction* [3] when defining algorithms to analyse them, assuming that the *region equivalence* is a *time-abstract bisimulation* for event-clock valuations.

[10] Again, this would not have been the case, had we relied on the approximation as defined for timed automaton. This case shows why we cannot approximate values up to $4 \cdot cmax$.

In the present work, we have shown that there is, in general, *no finite time-abstract language equivalence for* ECA, which implies, in particular, that the *region equivalence is not a time-abstract bisimulation* for event-clock valuations. Furthermore, we have shown that, for two classical definitions of regions [3,9], the region automaton of an ECAA sometimes recognise a *strict subset* of A's untimed language only.

To overcome this difficulty, we have proposed an alternative semantics for ECA, that we have called the *weak semantics*, for which regions *are* a finite time-abstract bisimulation. Thanks to this new semantics, we have managed to show that a slight modification of the definition of the region automaton, namely the *existential region automaton*, recognises the untimed language of the original ECA. Then, in order to obtain *efficient algorithms*, we have adapted the notions of zones and DBMs [18] to ECA. Unfortunately, neither the forward, nor the backward zone-based analysis of ECA terminate in general. We have thus adapted the classical widening operator [15] and provided an in-depth soundness proof, following the lines of [10].

To sum up, we believe that the present work covers the basic theory which is needed for the analysis of *linear-time properties* of ECA (such as reachability or safety). However, as far as we know, the decidability of *branching time properties* of ECAremains an open question. In the case of timed automata, many questions related to branching, such as *bisimulation*, or the *model-checking of CTL* or TCTL, can be decided thanks to the region automaton [2,13], which is *bisimilar* to the timed automaton it was built from. However, this is not the case anymore for ECA: neither the existential region automaton extracted from the classical semantics, nor the region automata built on top of the weak semantics are bisimilar to the ECA. Nevertheless, the absence of a finite bisimulation quotient for ECA does not necessarily entail that branching logics or games are undecidable. We leave these questions open for future works.

Acknowledgments This work has been supported by the projects: (i) QUASIMODO (FP7- ICT-STREP-214755), Quasimodo: "Quantitative System Properties in Model-Driven-Design of Embedded", http://www.quasimodo.aau.dk/, (ii) GASICS (ESF-EUROCORES LogiCCC), Gasics: "Games for Analysis and Synthesis of Interactive Computational Systems", http://www.ulb.ac.be/di/gasics/, (iii) Moves: "Fundamental Issues in Modelling, Verification and Evolution of Software", http://moves.vub.ac.be, a PAI program funded by the Federal Belgian Government and (iv) The European Union Seventh Framework Programme under Grant Agreement 601148 (Cassting), http://www.cassting-project.eu. Gilles Geeraerts has been supported by a 'Crédit aux chercheurs' from the Belgian FRS/F.N.R.S.

Appendix 1: Event-clock automata and timed automata

As stated in the introduction, ECA have been introduced as an alternative to timed automata, for the specification of timed languages. The original work on ECA[4] contains a thorough comparisons of the expressiveness of these two models. For the sake of completeness, we recall here the most salient result: each ECA can be transformed into a *non-deterministic timed automaton* that has the same language.

Timed automata We first recall briefly the definition of timed automaton, then present the construction.

Definition 50 ([3]) A *timed automaton* (TAfor short) is a tuple $B = \langle Q, q_i, \Sigma, X, \delta, \alpha\rangle$, where:

(1) Q is a finite set of *locations*,
(2) $Q_i \subseteq Q$ is a set of *initial* locations,
(3) Σ is a finite *alphabet*,

(4) X is a finite set of non-negative real-valued variables called *clocks*[11]
(5) $\delta \subseteq Q \times \Sigma \times \mathsf{Constr}(X) \times 2^X \times Q$ of edges,
(6) $\alpha \subseteq Q$ is the set of accepting locations.

We also require that, for each $q \in Q$, $\sigma \in \Sigma$, δ is defined for a finite number of $\psi \in \mathsf{Constr}(X)$.

A valuation of a set of clocks X is a function $v : X \to \mathbb{R}^{\geq 0}$. We denote by $\mathscr{V}(X)$ the set of valuations of X. For a valuation v, and a time delay $t \in \mathbb{R}^{\geq 0}$, we denote by $v + t$ the valuation s.t. $(v + t)(x) = v(x) + t$ for all x. An *extended state* (or simply a state) of a TA with set of locations Q and set of clocks X is a pair (q, v) s.t. $q \in Q$ and v is a valuation of the clocks in X. As for ECA, we define the semantics of timed automata by means of a transition system. We associate to a TA $B = \langle Q, Q_i, \Sigma, X, \delta, \alpha\rangle$ the infinite transition system $\mathsf{TS}_B = \left\langle Q^B, Q_i^B, \to, \alpha^B\right\rangle$, where:

(1) $Q^B = Q \times \mathscr{V}(X)$ is the set of extended states of B,
(2) $Q_i^B = Q_i \times \{v \mid v(x) = 0 \text{ forall } x \in X\}$,
(3) $\alpha^B = \{(q, v) \mid q \in \alpha\}$, and
(4) the transition relation $\to \subseteq \left(Q^B \times \mathbb{R}^{\geq 0} \times Q^B\right) \cup \left(Q^B \times \Sigma \times Q^B\right)$ is s.t.:
   (a) $\left((q, v), t, (q, v')\right) \in \to$ iff $v' = v + t$, and
   (b) $\left((q, v), a, (q', v')\right) \in \to$ iff there is $(q, a, \psi, r, q') \in \delta$ s.t. $v \models \psi$, and $v' = v[r := 0]$

Intuitively, this means that, on all edges (q, a, ψ, r, q'), ψ is a guard that must be satisfied by *the current valuation of the variables*, in order to fire the edge; and that r is a set of clock *that must be reset* when firing the edge. We adapt to TA the notions of run and language previously defined for ECA, as expected.

From ECA *to* TA Let us now recall the construction of [4] to translate an ECA A into a TA B that has the same accepted language.[12] In order to apply the construction, we need to slightly modify the syntax of the guards in the ECA. A *non-punctual event-clock constraint* is an event-clock constraint where the only atomic event-clock constraints containing an equality are of the form $x = \bot$ (thus constraints of the form $x = c$ with $c \in \mathbb{N}$ are disallowed). Remark that each event-clock constraint can be turned into an equivalent non-punctual one by substituting $x \geq c \wedge x \leq c$ to each $x = c$, and $x > c \vee x < c$ to each $x \neq c$. For any event-clock constraint ψ, we denote by $\mathsf{PConstr}(\psi)$ (resp. $\mathsf{HConstr}(\psi)$) the set of all atomic event clock constraints that (i) occur in ψ and (ii) range over a *prophecy* (resp. *history*) *clock*. Let $A = \left\langle Q^A, q_i^A, \Sigma, C, \delta^A, \alpha^A\right\rangle$ be an ECA. By abuse of notation, we let:

$$\mathsf{PConstr}(A) = \left\{\overrightarrow{x_a} = \bot \mid a \in \Sigma\right\} \cup \bigcup_{\substack{\psi \text{ s.t.} \\ (q, a, \psi, q') \in \delta^A}} \mathsf{PConstr}(\psi) \qquad (13)$$

That is, $\mathsf{PConstr}(A)$ is the set of all atomic event clock constraints that appear on the edges of A and that constrain prophecy clocks, plus all the constraints of the form $\overrightarrow{x_a} = \bot$. It is easy to see that:

Lemma 51 *For all* ECA A *without punctual guards,* $|\mathsf{PConstr}(A)| \leq (4 \times cmax + 5) \times |\Sigma|$.

[11] In TA, *clocks* are *not event-clocks*.

[12] Remark that in [4], the construction is given for EPA only. Adapting it to the full class of ECA is straightforward as history clocks are easily encoded in regular timed automata clocks. For the sake of completeness, we give here the construction in the general case of ECA.

Proof We follow the definition of $\mathsf{PConstr}\,(A)$, given by (13). Obviously, $|\{\overrightarrow{x_a} = \bot \mid a \in \Sigma\}| = |\mathbb{P}_\Sigma| = |\Sigma|$. Moreover, each non-punctual atomic clock constraint on a prophecy clock, and different from $\overrightarrow{x_a} = \bot$ (for some $a \in \Sigma$) is of the form $\overrightarrow{x_a} \simeq c$, with $c \in \{0, 1, \ldots, cmax\}$ and $\simeq \in \{<, \leq, \geq, >\}$. Hence, we conclude that:

$$\begin{aligned}|\mathsf{PConstr}\,(A)\,| &\leq |\Sigma| + |\Sigma| \times (cmax + 1) \times 4\\ &= (4 \times cmax + 5) \times |\Sigma|\end{aligned}$$

□

We are now ready to give the construction of the TA that accepts the same language as the $\mathsf{ECA}A$. The prophecy clocks will be encoded using non-determinism: in the TA, a guess is made on the values of the prophecy clocks, that will be checked when the corresponding event occurs. We assume that all guards in A are non-punctual, and contain neither disjunctions nor negations.[13] As a result, all atomic event-clock constraints occurring in A are of one of the following forms: $x \leq c$, $x < c$, $x \geq c$, $x > c$ or $x = \bot$. Then, the corresponding TA is $B = \left\langle Q^B, Q^B_i, \Sigma, X^B, \delta^B, \alpha^B \right\rangle$ where:

(1) $Q^B = Q^A \times 2^{\mathsf{PConstr}(A)} \times \mathscr{F}(\Sigma)$, where $\mathscr{F}(\Sigma)$ is the set of all Boolean functions $f : \Sigma \mapsto \{\mathsf{true}, \mathsf{false}\}$. That is, each location of B is a triple (q, Φ, bot), where q is a location of A, Φ is a set of atomic event-clock constraints on the prophecy clocks of A that need to be fulfilled and, for all letters $a \in \Sigma$, $\mathsf{bot}\,(a)$ indicates whether $\overleftarrow{x_a}$ equals $\bot$ in the original ECA.

(2) $Q^B_i = \{(q_i, \Phi, f_\bot) \mid \Phi$ contains onlyconstraints of the form $\overrightarrow{x_a} = \bot\}$ where for all $a \in \Sigma$: $f_\bot(a) = \mathsf{true}$.

(3) $X^B = \{z_\varphi \mid \varphi \in \mathsf{PConstr}\,(a)\} \cup \{x_a \mid \overleftarrow{x_a} \in C\}$, i.e., B contains one clock z_φ per atomic clock constraint φ on a prophecy clock of A, and one clock x_a per letter of the alphabet (as we will see, x_a will be used to track the value of the corresponding history clock $\overleftarrow{x_a}$ in A).

(4) δ^B contains an edge $\big((q, \Phi, \mathsf{bot}), a, \psi, r, (q', \Phi', \mathsf{bot}')\big)$ iff there is an edge (q, a, χ, q') in δ^A, and:

  (a) $(\overrightarrow{x_a} = \bot) \notin \Phi$,

  (b) The function bot is s.t. for all $b \in \Sigma$, $(\overleftarrow{x_b} \sim c) \in \mathsf{HConstr}\,(\chi)$ implies $\neg\mathsf{bot}\,(b)$, and $(\overleftarrow{x_b} = \bot) \in \mathsf{HConstr}\,(\chi)$ implies $\mathsf{bot}\,(b)$,

  (c) The guard ψ is:

$$\psi = \bigwedge_{(\overrightarrow{x_a} \sim c) \in \Phi} \left(z_{(\overrightarrow{x_a} \sim c)} \sim c\right) \wedge \bigwedge_{\substack{(\overleftarrow{x_b} \sim c) \in \\ \mathsf{HConstr}(\chi) \\ \text{with } b \in \Sigma}} \left(x_b \sim c\right)$$

  with $\sim \in \{\leq, <, >, \geq\}$ and $c \in \mathbb{N}$.

  (d) The reset r is:

$$r = \{x_a\} \cup \bigcup_{\substack{(\overrightarrow{x_b} \succeq c) \in \mathsf{PConstr}(\chi) \\ \text{with } b \in \Sigma}} \left\{z_{(\overrightarrow{x_b} \succeq c)}\right\} \cup \bigcup_{\substack{(\overrightarrow{x_b} \preceq c) \in \mathsf{PConstr}(\chi) \\ \text{s.t. } b = a \,\vee\, (\overrightarrow{x_b} \preceq c) \notin \Phi}} \left\{z_{(\overrightarrow{x_b} \preceq c)}\right\}$$

  with $\succeq \in \{>, \geq\}$, $\preceq \in \{<, \leq\}$ and $c \in \mathbb{N}$.

[13] This is without loss of generality as one can always push the negation inwards, replace each atomic constraint of the form $\neg(x < c)$ (resp. $\neg(x > c)$, $\neg(x = \bot)$, $\neg\mathsf{true}$) by the equivalent atomic constraint $x \geq c$ ($x \leq c$, $x \geq 0$, $x < 0$), write a guard in disjunctive normal form and introduce an edge for each disjunct.

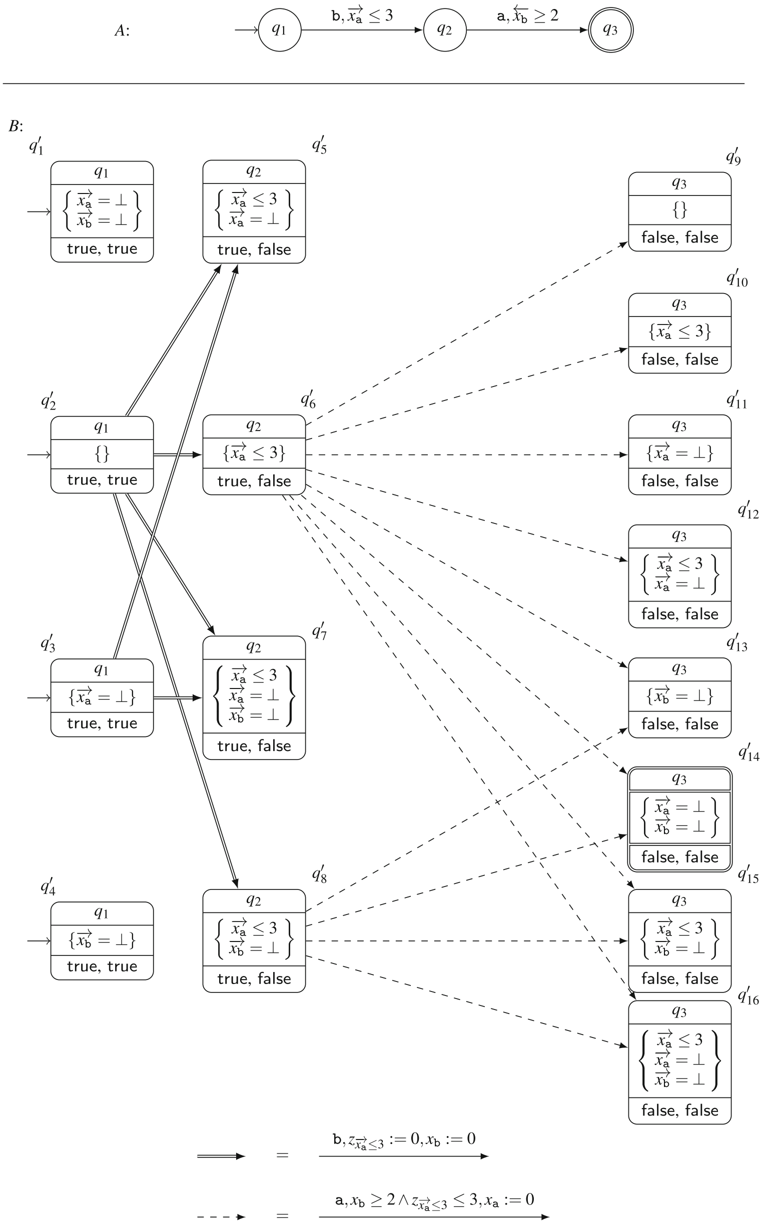

A:
q_1
$\mathsf{b}, \overrightarrow{x_\mathsf{a}} \leq 3$
q_2
$\mathsf{a}, \overleftarrow{x_\mathsf{b}} \geq 2$
q_3
B:
q'_1: q_1; $\{\overrightarrow{x_\mathsf{a}} = \bot, \overrightarrow{x_\mathsf{b}} = \bot\}$; true, true
q'_2: q_1; $\{\}$; true, true
q'_3: q_1; $\{\overrightarrow{x_\mathsf{a}} = \bot\}$; true, true
q'_4: q_1; $\{\overrightarrow{x_\mathsf{b}} = \bot\}$; true, true
q'_5: q_2; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{a}} = \bot\}$; true, false
q'_6: q_2; $\{\overrightarrow{x_\mathsf{a}} \leq 3\}$; true, false
q'_7: q_2; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{a}} = \bot, \overrightarrow{x_\mathsf{b}} = \bot\}$; true, false
q'_8: q_2; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{b}} = \bot\}$; true, false
q'_9: q_3; $\{\}$; false, false
q'_{10}: q_3; $\{\overrightarrow{x_\mathsf{a}} \leq 3\}$; false, false
q'_{11}: q_3; $\{\overrightarrow{x_\mathsf{a}} = \bot\}$; false, false
q'_{12}: q_3; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{a}} = \bot\}$; false, false
q'_{13}: q_3; $\{\overrightarrow{x_\mathsf{b}} = \bot\}$; false, false
q'_{14}: q_3; $\{\overrightarrow{x_\mathsf{a}} = \bot, \overrightarrow{x_\mathsf{b}} = \bot\}$; false, false
q'_{15}: q_3; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{b}} = \bot\}$; false, false
q'_{16}: q_3; $\{\overrightarrow{x_\mathsf{a}} \leq 3, \overrightarrow{x_\mathsf{a}} = \bot, \overrightarrow{x_\mathsf{b}} = \bot\}$; false, false
= $\mathsf{b}, z_{\overrightarrow{x_\mathsf{a}} \leq 3} := 0, x_\mathsf{b} := 0$
= $\mathsf{a}, x_\mathsf{b} \geq 2 \land z_{\overrightarrow{x_\mathsf{a}} \leq 3} \leq 3, x_\mathsf{a} := 0$

Fig. 11 An ECAA (*top*) and its corresponding TA B (*bottom*), with $\mathsf{L}(A) = \mathsf{L}(B)$. In B, only the states that are reachable from the initial state are shown

(e) The set Φ' is s.t.:

$$\{(\overrightarrow{x_b} \simeq d) \in \Phi \mid b \neq a\} \cup \mathsf{PConstr}(\chi) \subseteq \Phi'$$

with $\simeq \in \{\leq, <, =, >, \geq\}$ and $d \in \mathbb{N} \cup \{\bot\}$.

(f) Finally, the function bot' is s.t. for all $b \in \Sigma$:

$$\mathsf{bot}'(b) = \begin{cases} \mathsf{false} & \text{if } b = a \\ \mathsf{bot}\,(b) & \text{otherwise} \end{cases}$$

(5) $\alpha^B = \alpha^A \times \{\Phi_\perp\}$ where $\Phi_\perp = \{\overrightarrow{x_a} = \perp \mid a \in \Sigma\}$.

To illustrate this rather technical construction, we consider the example given in Fig. 11. The $\mathsf{ECA}\,A$ (top of the figure) accepts all timed words of the form $(\mathtt{b}, t_1), (\mathtt{a}, t_2)$ s.t. $t_2 - t_1 \in [2, 3]$. The $\mathsf{TA}\,B$ (bottom of the figure) has been obtained from A by applying the above construction. In the figure, each B state (q, Φ, bot) is drawn with q at the top, the set Φ in the middle and the pair of values $\mathsf{bot}\,(\mathtt{a})$, $\mathsf{bot}\,(\mathtt{b})$ at the bottom. On the edges, an expression of the form $x := 0$ means that x is reset by the edge. As can be seen in this example, in each B state (q, Φ, bot), the set Φ contains guesses on constraints on the prophecy clocks of A that should be fulfilled—this explains the $\subseteq$ symbol in the definition of item (4e). B can move from (q, Φ, bot) to $(q', \Phi', \mathsf{bot}')$, iff there is, in A, a corresponding edge from q to q', s.t. the set of constraints Φ' is updated so that it contains all constraints ranging on prophecy clocks that appear in the guard χ of the edge. For instance, all successors of q_2' are of the form $(q_2, \Phi, \mathsf{bot})$ with $(\overrightarrow{x_\mathtt{a}} \leq 3) \in \Phi$, as all these successors are obtained thanks to the edge from q_1 to q_2, whose guard is $(\overrightarrow{x_\mathtt{a}} \leq 3)$. Remark however, that q_2' has several successors, as the TA B *guesses* a set of constraints on the prophecy clocks that should be fulfilled. For instance, when moving from q_2' to q_8', the TA *guesses* that $\overrightarrow{x_\mathtt{b}} = \perp$, i.e., that no more $\mathtt{b}$'s will be read, but when going from q_2' to q_6', it guesses otherwise. In order to be able to check that the constraints in Φ hold, a clock z_φ is reset every time an edge is crossed whose guard implies that the constraint φ should hold. For instance, the clock $z_{\overrightarrow{x_\mathtt{a}} \leq 3}$ is reset on every outgoing edge of q_2'. Then, the values of those clocks are checked when the corresponding letter is read. For instance, when going from q_8' to q_{14}', one has to check that $z_{\overrightarrow{x_\mathtt{a}} \leq 3} \leq 3$, as the edge is labelled by $\mathtt{a}$, and the constraint $\overrightarrow{x_\mathtt{a}} \leq 3$ occurs in q_8'. To sum up, *prophecies in the* $\mathsf{ECA}\,A$ *are replaced by non-determinism in the* TA B, while remembering the constraints that have to be fulfilled in each state, and using one clock per constraint to check that it holds. History clocks are handled straightforwardly by resetting a clock x_a every time an a-labeled edge is crossed (and relying on the value of $\mathsf{bot}(a)$ that is stored in each state to remember whether the corresponding history clock $\overleftarrow{x_a}$ is equal to $\perp$ or not).

References

1. Alur R (1999) Timed automata. In: Proceedings of CAV'99, vol 1633. Lecture notes in computer science. Springer, Berlin, pp 8–22
2. Alur R, Courcoubetis C, Dill D (1993) Model-checking in dense real-time. Inf Comput 104:2–34
3. Alur R, Dill D (1994) A theory of timed automata. Theor Comput Sci 126(2):183–236
4. Alur R, Fix L, Henzinger TA (1999) Event-clock automata: a determinizable class of timed automata. Theor Comput Sci 211(1–2):253–273
5. Behrmann G, David A, Larsen KG, Håkansson J, Pettersson P, Yi W, Hendriks M (2006) Uppaal 4.0. In: Proceedings of QEST'06. IEEE Computer Society, New York, pp 125–126
6. Bellman R (1957) Dynamic programming. Princeton university press, Princeton
7. Bengtsson J, Griffioen WOD, Kristoffersen KJ, Larsen KG, Larsson F, Pettersson P, Yi W (2002) Automated verification of an audio-control protocol using Uppaal. J Log Algebr Program 52–53:163–181
8. Bouyer P (2002) Modèles et algorithmes pour la vérification des systèmes temporisés. Thèse de doctorat, Laboratoire Spécification et Vérification. ENS Cachan, France
9. Bouyer P (2002) Timed automata may cause some troubles. In Research Report LSV-02-9, LSV, ENS DE CACHAN. http://www.lsv.enscachan.fr/Publis/RAPPORTS_LSV/PS/rrlsv-2002-9.rr.ps

10. Bouyer P (2004) Forward analysis of updatable timed automata. Formal Methods Syst Des 24(3):281–320
11. Bouyer P, Laroussinie F, Reynier P-A (2005) Diagonal constraints in timed automata: Forward analysis of timed systems. In: Pettersson P, Yi W (eds) Formal modeling and analysis of timed systems, vol 3829. Lecture notes in computer science. Springer, Berlin, pp 112–126
12. Bozga M, Daws C, Maler O, Olivero A, Tripakis S, Yovine S (1998) Kronos: A model-checking tool for real-time systems. In: Proceedings of CAV'98, vol 1427. Lecture notes in computer science. Springer, Berlin, pp 546–550
13. Cerans K (1993) Decidability of bisimulation equivalences for parallel timer processes. In: Proceedings of the fourth international workshop on computer aided verification, CAV '92, London. Springer, Berlin, pp 302–315
14. Daws C, Tripakis S (1998) Model checking of real-time reachability properties using abstractions. In Steffen B (ed) Proceedings of TACAS'98, vol 1384. Lecture notes in computer science. Springer, Berlin, pp 313–329
15. Daws C, Tripakis S (1998) Model checking of real-time reachability properties using abstractions. In: TACAS. Springer, Berlin, pp 313–329
16. Di Giampaolo B, Geeraerts G, Raskin J, Sznajder N (2010) Safraless procedures for timed specifications. In: Proceedings of FORMATS'10, vol 6246. Lecture notes in computer science. Springer, Berlin, pp 2–22
17. Diekert V, Gastin P, Petit A (1997) Removing epsilon-transitions in timed automata. In: STACS 97, 14th Annual Symposium on theoretical aspects of computer science, vol 1200. Lecture notes in computer science. Springer, Berlin, pp 583–594
18. Dill DL (1989) Timing assumptions and verification of finite-state concurrent systems. In: Proceedings of automatic verification methods for finite state systems, vol 407. Lecture notes in computer science. Springer, Berlin, pp 197–212
19. Dima C (1999) Kleene theorems for event-clock automata. In: Proceedings of FCT'99, volume 1684 of Lecture Notes in computer science. Springer, Berlin, pp 215–225
20. D'Souza D, Tabareau N (2004) On timed automata with input-determined guards. In Proccedings of FORMATS/FTRTFT'04, vol 3253. Lecture notes in computer science. Springer, Berlin, pp 68–83
21. Geeraerts G, Raskin J-F, Sznajder N (2011) Event-clock automata : from theory to practice. In: Fahrenberg U, Tripakis S (eds) Proceedings of the 9th international conference on formal modelling and analysis of timed systems (FORMATS'11), vol 6919. Lecture notes in computer science. Springer, Berlin, pp 209–224
22. Lindahl M, Pettersson P, Yi W (1998) Formal design and analysis of a gear-box controller. In: Proceedings of the 4th workshop on tools and algorithms for the construction and analysis of systems. vol 1384 .Lecture notes in computer science. Springer, Berlin, pp 281–297
23. Miné A (2006) The octagon abstract domain. Higher-Order Symbol Comput 19(1):31–100, 2006. http://www.di.ens.fr/~mine/publi/article-mine-HOSC06.pdf
24. Pettersson, P, Yi W editors. Formal Modeling and Analysis of Timed Systems. In: Proceedings of the third international conference, FORMATS 2005, Uppsala, September 26–28, 2005, vol 3829. Lecture notes in computer science. Springer, Berlin
25. Raskin J-F, Schobbens P-Y (1998) The logic of event clocks: decidability, complexity and expressiveness. Automatica 34(3):247–282
26. Sorea M (2001) Tempo: a model-checker for event-recording automata. In: Proceedings of RT-TOOLS'01, Aalborg
27. Tang N, Ogawa M (2009) Event-clock visibly pushdown automata. In: Proceedings of SOFSEM'09, vol 5404. Lecture notes in computer science. Springer, Berlin, pp 558–569