Multi-Timed Bisimulation for Distributed Timed Automata

James Ortiz, Moussa Amrani, and Pierre-Yves Schobbens

Computer Science Faculty, University of Namur
{james.ortizvega, moussa.amrani, pierre-yves.schobbens}@unamur.be

Abstract. Formal verification methods, such as model checking, have been used to verify the correctness of large-scale and complex software systems. However, the so-called state space explosion problem often prevents model checking to be used in practice for large-scale and complex software systems. Timed bisimulation is an important technique which can be used for reasoning about behavioral equivalence between different components of a complex real-time system. The verification of timed bisimulation is a difficult and challenging problem because the state explosion caused by both functional and timing constraints must be taken into account. Timed bisimulation was shown decidable for Timed Automata (TA). Distributed Timed Automata (DTA) and TA with Independent Clocks (icTA) were introduced to model Distributed Real-time Systems (DTS). They are a variant of TA with local clocks that may not run at the same rate. In this paper, we first propose to extend the theory of Timed Labeled Transition Systems (TLTS) to Multi-Timed Labeled Transition Systems (MLTS), and relate them by an extension of timed bisimulation to multi-timed bisimulation. We prove the decidability of multi-timed bisimulation and present an EXPTIME algorithm for deciding whether two icTA are multi-timed bisimilar.

1 Introduction

Distributed Real-Time Systems (DTS) are increasing with the scientific and technological advances of computer networks. The high demand for computer networks has caused the development of new complex applications which benefit from the high performance and resources offered by modern telecommunications networks. Current researches in the area of DTS have emerged from the need to specify and analyze the behavior of these systems, where both distributed behavior and timing constraints are present. Formal verification methods, such as model checking, have been used to verify the correctness of complex DTS. Model checking over DTS becomes rapidly intractable because the state space often grows exponentially with the number of components considered. A technique to reduce the state space is to merge states with the same behaviour. For untimed systems, the notion of *bisimulation* [22] is classically used to this end, and its natural extension for real-time systems, *timed bisimulation*, was already shown decidable for Timed Automata (TA) [5, 20]. A timed automaton is a finite automaton augmented with real-valued clocks, represented as variables that increase at the same rate as time progresses. TA assume perfect clocks: all clocks

have infinite precision and are perfectly synchronized. In this paper, we study two variants of TA called Distributed Timed Automata (DTA) and Timed Automata with Independent Clocks (icTA) proposed by [19], [2] and [24] to model DTS, where the clocks are not necessarily synchronized. TA have been used to model DTS such as Controller Area Network (CAN) [23] and WirelessHART Networks [16]. But, TA, icTA and timed bisimulation are based on a sequential semantics of a Timed Labelled Transition Systems (TLTS), i.e., a run of a TLTS is given by a sequence of actions and timestamps.

Unfortunately, a sequential semantics does not describe completely the behavior of the DTS, because interactions between processes with their associated local clocks that are running at the same rate and distribution of the actions over the components are not considered. Also, model checking and bisimulation equivalence algorithms have been implemented in tools [27][26] under the sequential semantics adopted by the model (e.g., TA, TLTS, etc). In contrast, behavioral equivalences for DTS have only been introduced in [6]. It is, however, not clear whether such equivalences agree with the distributed timed properties in DTS. Therefore, we propose an alternative semantics to the classical sequential semantics for TLTS and icTA: specifically, a run of a system in our alternative semantics is given by the sequences of pairs (action, tuples of timestamps). Our alternative semantics is introduced because we need to consider a semantics which express the distribution of the actions and timestamps over the components. With our semantics it is now possible to analyze the local behavior of the components independently. Our alternative semantics enhances the expressiveness of the TLTS (and icTA). Thus, we extend the TLTS to work with the notion of multiple local times, we call this extension Multi-Timed Labelled Transition Systems (MLTS). Furthermore we propose efficient algorithms using partition refinement techniques [25].

Contributions: One of our main contributions is to incorporate a alternative semantics over sequential semantics for TLTS and icTA. Also, we extend the classical theory of timed bisimulation with the notion multi-timed bisimulation and their corresponding decision algorithms. Since TA are special variants of icTA, this result conservatively extends the expressiveness of TA and TLTS. We also present two algorithms:(i) a forward reachability algorithm for the parallel composition of two icTA, which will help us to minimize the state space exploration by our second algorithm, and (ii) a decision algorithms for multi-timed bisimulation using the zone-based technique [9]. For multi-timed bisimulation, an extension of the usual partition refinement algorithm with signatures [26] and [10] is considered. Multi-timed bisimulation is a relation over local clocks (and processes), and cannot be computed with the standard partition refinement algorithm [25]. Instead, we extend the approach in [26] and [10]: our algorithm successively refines a set of zones such that ultimately each zone contains only multi-timed bisimilar states. Furthermore, the complexity of our algorithm for deciding whether two icTA are multi-timed bisimilar is EXPTIME-complete. Since the timed bisimulation for TA considered in [27][26] can be regarded as a special case of multi-timed bisimulation, our results apply also in that setting. Hence our decision algorithms could potentially be used to analyze DTS of larger size.

Structure of the paper. After recalling preliminary notions in Section 2, we introduce our alternative semantics for icTA in Section 3, based on multi-timed words consumed by MLTS. Section 4 deals with bisimulation: we first define multi-timed bisimulation, by adapting the classical definition to MLTS, then show its decidability by exhibiting an EXPTIME algorithm. Finally, Section 5 compares our work with existing contributions, and Section 6 concludes.

2 Preliminaries

In this section, we start by describing clocks, timed words and clock constraints. Also, we briefly describe TLTS and TA that are among the most well-known formalisms for modeling the behavior of the real-time systems.

Let Σ be a finite alphabet of actions. The set of all *finite words* over Σ will be denoted by Σ^*. Let $\mathbb{N}$ be the set of natural numbers, $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_{\geq 0}$ the set of nonnegative real numbers. A timed word [5] over an alphabet Σ is a finite sequence $\theta = ((\sigma_1, t_1), (\sigma_2, t_2) \dots (\sigma_n, t_n))$ of actions $\sigma_i \in \Sigma$ that are paired with nonnegative real numbers $t_i \in \mathbb{R}_{\geq 0}$ such that the sequence $t = t_1 t_2 \cdots t_n$ of time-stamps is nondecreasing (i.e., $t_i \leq t_{i+1}$ for all $1 \leq i < n$). Sometimes, we denote the timed word by the pair $\theta = (\sigma, t)$, where $\sigma \in \Sigma^*$ is an untimed word over Σ and t is a sequence of time-stamps of the same length.

A clock is a variable that increases with time. Let X be a finite set of clock names. A clock constraint $\phi \in \Phi(X)$ is a conjunction of comparisons of a clock with a constant c, given by the following grammar, where $x \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, >, \leq, \geq, =\}$:

$$\phi ::= true \mid x \sim c \mid \phi_1 \ \wedge \ \phi_2$$

A clock valuation over X is a mapping $\nu : X \to \mathbb{R}_{\geq 0}$, the set of valuations is denoted $\mathbb{R}_{\geq 0}^X$. For a valuation $\nu \in \mathbb{R}_{\geq 0}^X$ and a time value $t \in \mathbb{R}_{\geq 0}$, let $\nu + t$ denote the valuation such that $(\nu + t)(x) = \nu(x) + t$, for each clock $x \in X$. Let $Y \subseteq X$, $\nu[Y \to 0]$ be the valuation defined by $\nu[Y \leftarrow 0](x) = 0$ for any $x \in Y$ and $\nu[Y \leftarrow 0](x) = \nu(x)$ otherwise. Given $Y \subseteq X$, the projection of ν on Y, written $\nu\rfloor_Y$, is the valuation over Y only containing the values in ν of clocks in Y.

2.1 Timed Transition Systems (TLTS)

TLTS are used to describe the behavior of time systems.

Definition 1. *A TLTS [18] is a tuple $\mathcal{D} = (Q, q_0, \Sigma, \to_{tlts})$ where:*

(i) Q is a set of states.
(ii) $q_0 \in Q$ is the initial state.
(iii) Σ is a finite alphabet.
(iv) $\to_{tlts} \subseteq Q \times (\Sigma \uplus \mathbb{R}_{\geq 0}) \times Q$ is a set of transitions.

The transitions from state to state of a TLTS are noted in the following way: (*i*) A transition (q, a, q') is denoted $q \xrightarrow{a} q'$ and is called a *discrete transition*, if

$a \in \Sigma$ and $(q, a, q') \in \rightarrow_{tlts}$. (*ii*) A transition (q, d, q') is denoted $q \xrightarrow{d} q'$ and is called a *delay transition*, if $d \in \mathbb{R}_{\geq 0}$ and $(q, d, q') \in \rightarrow_{tlts}$.

A run of $\mathcal{D}$ can be defined as a finite sequence of moves, where discrete and continuous transitions alternate: : $\rho = q_0 \xrightarrow{d_1} q'_0 \xrightarrow{a_1} q_1 \xrightarrow{d_2} q'_1 \xrightarrow{a_2} q_2 \ldots q_{n-2} \xrightarrow{d_{n-1}} q'_{n-2} \xrightarrow{a_{n-1}} q_{n-1}$, where $\forall 0 \leq i \leq n-1$, $q_i \in Q$, $\forall j \geq 0$, $d_j \in \mathbb{R}_{\geq 0}$, $q'_j \in Q$ and $a_j \in \Sigma$. A run is initial if it starts in q_0. A run is maximal if it ends in a state without outgoing transitions. We write $Runs(\mathcal{D}, q_0)$ for the set of maximal finite runs of $\mathcal{D}$ from state q_0.

2.2 Strong Timed Bisimulation

In the context of real-time systems, several bisimulation relations are of interest, like the time-abstract bisimulation or the weak and strong timed bisimulations.

Definition 2. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two TLTS over the set of actions Σ. Let $Q_{\mathcal{D}_1}$ (resp., $Q_{\mathcal{D}_2}$) be the set of states of $\mathcal{D}_1$ (resp., $\mathcal{D}_2$). Let $\mathcal{R}$ be a binary relation over $Q_{\mathcal{D}_1} \times Q_{\mathcal{D}_2}$. We say that $\mathcal{R}$ is a strong timed bisimulation [14] whether, for all $q_{\mathcal{D}_1} \mathcal{R} q_{\mathcal{D}_2}$:*

(i) For every discrete transition $q_{\mathcal{D}_1} \xrightarrow{a}_{\mathcal{D}_1} q'_{\mathcal{D}_1}$ with $a \in \Sigma$, there exists a matching transition $q_{\mathcal{D}_2} \xrightarrow{a}_{\mathcal{D}_2} q'_{\mathcal{D}_2}$ such that $q'_{\mathcal{D}_1} \mathcal{R} q'_{\mathcal{D}_2}$ and symmetrically.

(ii) For every delay transition $q_{\mathcal{D}_1} \xrightarrow{d}_{\mathcal{D}_1} q'_{\mathcal{D}_1}$ with $d \in \mathbb{R}_{\geq 0}$, there exists a matching transition $q_{\mathcal{D}_2} \xrightarrow{d}_{\mathcal{D}_2} q'_{\mathcal{D}_2}$ such that $q'_{\mathcal{D}_1} \mathcal{R} q'_{\mathcal{D}_2}$ and symmetrically.

Two states $q_{\mathcal{D}_1}$ and $q_{\mathcal{D}_2}$ are timed bisimilar, written $q_{\mathcal{D}_1} \sim q_{\mathcal{D}_2}$, iff there is a timed bisimulation that relates them. $\mathcal{D}_1$ and $\mathcal{D}_2$ are timed bisimilar, written $\mathcal{D}_1 \sim \mathcal{D}_2$, if there exists a timed bisimulation relation $\mathcal{R}$ over $\mathcal{D}_1$ and $\mathcal{D}_2$ containing the pair of initial states.

2.3 Timed Automata

A timed automaton is a classical finite automaton which can manipulate clocks, evolving continuously and synchronously with global time; its clocks are perfectly synchronized.

Definition 3. *A Timed Automaton is a tuple $\mathcal{A} = (\Sigma, X, S, s_0, \rightarrow_{ta}, I, F)$, where:*

(i) Σ is a finite alphabet.
(ii) X is a finite set of positive real variables called clocks.
(iii) S is a finite set of locations.
(iv) $s_0 \in S$ is the initial location.
(v) $\rightarrow_{ta} \subseteq S \times \Sigma \times \Phi(X) \times 2^X \times S$ is a finite set of transitions.
(vi) $I : S \rightarrow \Phi(X)$ gives the invariant of each location
(vii) $F \subseteq S$ is a subset of accepting locations.

A timed transition is tuple $(s, a, \phi, Y, s') \in \rightarrow_{ta}$ (it is often written $s \xrightarrow{a,\phi,Y} s'$), where the locations s and $s' \in S$, the action $a \in \Sigma$, the guard $\phi \in \Phi(X)$ and reset clocks $Y \subseteq X$.

The semantics of a timed automaton $\mathcal{A}$ is given by a $\mathsf{TLTS}(\mathcal{A}) = (Q, q_0, \Sigma \cup \mathbb{R}_{\geq 0}, \rightarrow_{tlts})$ where the set Q of states is $\{q = (s, \nu) \in S \times \mathbb{R}^X_{\geq 0} \mid \nu \models I(s)\}$, the starting state is $q_0 = (s_0, \nu_0)$, where ν_0 is the valuation that assigns 0 to all the clocks, and the transition relation $\rightarrow_{tlts} \subseteq (Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q)$ is composed of:

(i) A transition $((s, \nu), t, (s, \nu + t))$ is denoted $(s, \nu) \xrightarrow{t}_{tlts} (s, \nu + t)$, and is called a *delay transition*, if $t \in \mathbb{R}_{\geq 0}$ and $\forall\ 0 \leq t' \leq t, \nu + t' \models I(s)$,

(ii) A transition $((s, \nu), a, (s', \nu))$ is denoted $(s, \nu) \xrightarrow{a}_{tlts} (s', \nu)$, and is called a *discrete transition*, if $\exists\ s \xrightarrow{a,\phi,Y} s'$ such that $\nu \models \phi \wedge I(s)$, $\nu' = \nu[Y \rightarrow 0]$, and $\nu' \models I(s')$.

Example 1. In the timed automaton given in Figure 1, the clock constraints $x > 3$, $x < 10$ and $y = 9$ are the guards. To take the transition from s_1 to s_2, action a must occur and the clock variable x must have a value greater than 3. If this transition is taken, the clock variable x is set to 0. The constraint $x < 7$ is an invariant of s_1 and forces to take the transition from s_1 to s_2 while the clock variable x has a value smaller than 7. Note that using this invariant does not have the same effect as having the guard $x > 3 \wedge x < 7$ on the transition. Due to the invariant true in s_2, time can progress without bound while being in s_2. Note that the invariant $x < 7$ in location s_1, leads to the effect that x cannot progress beyond 7.

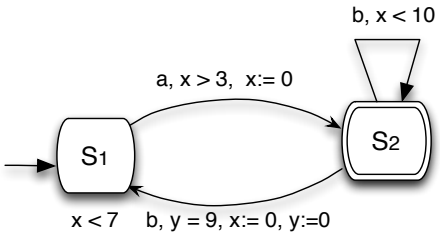

Fig. 1. Timed Automaton

TA are neither determinizable nor complementable. Their emptiness problem can be solved using the region construction, but their universality and inclusion problems are undecidable [5].

For modeling concurrent real-time systems, it is more convenient to be able to compose several TA. Networks of TA (NTA) [5] is a formalism used for modeling a network of components (or automata) running in parallel and synchronizing with each other (i.e. $\mathcal{A}_1 \parallel \mathcal{A}_2$ where $\mathcal{A}_1$ and $\mathcal{A}_2$ are TA).

2.4 Timed Automata with Independent Clocks (icTA)

icTA is an extension of Timed Automata (TA): they model the behaviour of distributed systems with a single, global TA where each clock is associated to

the process that owns it. An icTA is therefore always parameterised by a set Proc of processes, and is associated with an *ownership map* π, and each process runs at its own rate τ_p computed from a global clock.

Definition 4 (TA with Independent Clocks (icTA) [2, Def. 3]). *An Timed Automaton with Independent Clocks (icTA) is a tuple $\mathcal{A} = (\mathcal{B}, \pi)$, where $\mathcal{B}$ is a TA and π is a function maps each clock to a process $\pi : X \to \mathsf{Proc}$ called the ownership map, maps each clock to its owning process.*

A Rate is a tuple $\tau = (\tau_q)_{q \in Proc}$ of local time functions. Each local time function τ_q maps the reference time to the time of process q, i.e, $\tau_q : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. The functions τ_q must be continuous, strictly increasing, divergent, and satisfy $\tau_q(0) = 0$. The set of all these tuples τ is denoted by Rates.

Definition 5. *[2] A* run *of an icTA $\mathcal{A}$ for $\tau \in Rates$ is a sequence $(s_0, \nu_0) \xrightarrow{t_1} (s'_0, \nu'_0) \xrightarrow{a_1} (s_1, \nu_1) \xrightarrow{t_2} (s'_1, \nu'_1) \ldots (s_{n-2}, \nu_{n-2}) \xrightarrow{t_{n-1}} (s'_{n-2}, \nu'_{n-2}) \xrightarrow{a_{n-1}} (s_{n-1}, \nu_{n-1})$ where $\forall 1 \leq i \leq n$, $s_i \in S$ and $\forall j \leq n-1$, $t_j \in \mathbb{R}_{\geq 0}$ and $a_j \in \Sigma$. A finite run of an icTA $\mathcal{A}$ over a untimed word is called an accepting run, iff $q_n \in F$. The untimed language of $\mathcal{A}$ for τ is denoted $\mathcal{L}(\mathcal{A}, \tau)$. $\mathcal{L}(\mathcal{A}, \tau)$ is defined as the set of untimed words $\sigma \in \Sigma^*$ of accepting runs of $\mathcal{A}$ for τ. The existential language is defined as $\mathcal{L}_\exists(\mathcal{A}) = \bigcup_{\tau \in Rates} \mathcal{L}(\mathcal{A}, \tau)$ and the universal language is defined as $\mathcal{L}_\forall(\mathcal{A}) = \bigcap_{\tau \in Rates} \mathcal{L}(\mathcal{A}, \tau)$.*

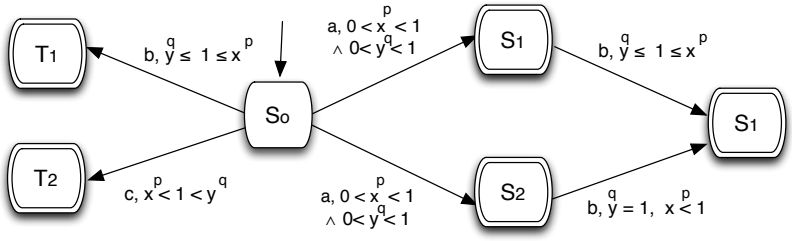

Fig. 2. Example of icTA from [2]

Example 2. Let us consider the icTA of the Figure 2. This icTA $\mathcal{B}$ contains the set of processes $\{p, q\}$, the automaton contains 6 locations, $\Sigma = \{a, b, c\}$, s_0 is the initial location and also we assume $\pi(x) = p$ and $\pi(y) = q$. If both clocks are completely synchronized, they follow the same local clock rate (i.e, $\tau_p = \tau_q$), then the automaton corresponds to a standard TA. The language in this case is $\mathcal{L}(\mathcal{B}, \tau) = \{a\}$. If the clock y runs slower than clock x (i.e, $\tau'_q \leq \tau'_p$). The language in this case is $\mathcal{L}(\mathcal{B}, \tau') = \{a, ab, b\}$. The existential language is $\mathcal{L}_\exists(\mathcal{B})$ $= \{a, ab, b, c\}$ and the universal language is $\mathcal{L}_\forall(\mathcal{B}) = \{a, ab\}$.

3 An Alternative Semantics for DTA

In this section, we define an alternative semantics (which we will call multi-timed semantics) for icTA as opposed to the mono-timed semantics of [2]. The main

problem with the semantics of [2] is that they use the reference time. The benefits of this new definition are threefold. First, the multi-timed semantics preserves the untimed language of the icTA. Second, the multi-timed semantics can work with multi-timed words. Third, the region equivalence defined in [2] could form a finite time-abstract bisimulation on the multi-timed semantics. Hence, the multi-timed semantics allows to build a region automaton that accepts exactly $\mathsf{Untime}(\mathcal{L}(\mathcal{A}))$ for all icTA $\mathcal{A}$ [2]. Thus, we extend TLTS and icTA to their multi-timed version.

3.1 Multi-Timed Actions

Let $Proc$ be a non-empty set of processes, then, we denote by $\mathbb{R}_{\geq 0}^{Proc}$ the set of functions from $Proc$ to $\mathbb{R}$, that we call *tuples*. A tuple $\boldsymbol{d} \in \mathbb{R}_{\geq 0}^{Proc}$ is smaller that $\boldsymbol{d}'$, noted, $\boldsymbol{d} < \boldsymbol{d}'$ iff $\forall i \in Proc\ \boldsymbol{d}_i \leq \boldsymbol{d}'_i$ and $\exists i \in Proc\ \boldsymbol{d}_i < \boldsymbol{d}'_i$. A Monotone Sequence of Tuples (MST) is a sequence $\boldsymbol{d} = \boldsymbol{d}_1\boldsymbol{d}_2 \cdots \boldsymbol{d}_n$ of tuples of $\mathbb{R}_{\geq 0}^{Proc}$ where : $\forall j \in 1 \cdots n-1$, $\boldsymbol{d}_j \leq \boldsymbol{d}_{j+1}$. A multi-timed word on Σ is a pair $\theta = (\sigma, \boldsymbol{d})$ where $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ is a finite word $\sigma \in \Sigma^*$, and $\boldsymbol{d} = \boldsymbol{d}_1\boldsymbol{d}_2 \ldots \boldsymbol{d}_n$ is a MST of the same length. This is the analog of a *timed word* (or *multi-timed action*) [5]. A *multi-timed word* can equivalently be seen as a sequence of pairs in $\Sigma \times \mathbb{R}_{\geq 0}^{Proc}$.

3.2 Multi-Timed Labeled Transition Systems

Our multi-timed semantics is defined in terms of *runs* that record the state and clock values at each transition points traversed during the consumption of a *multi-timed* word. Instead of observing actions at a global time, a multi-timed word allows to synchronise processes on a common action that may occur at a specific process time.

Definition 6 (Multi-Timed Labelled Transition System). *A Multi-Timed Labelled Transition System (MLTS) over a set of processes Proc is a tuple* $\mathcal{M} = (Q, q_0, \Sigma, \rightarrow_{mlts})$ *such that:* (i) Q *is a set of states.* (ii) $q_0 \in Q$ *is the initial state.* (iii) Σ *is a finite alphabet.* (v) $\rightarrow_{mlts} \subseteq Q \times (\Sigma \uplus \mathbb{R}_{\geq 0}^{Proc}) \times Q$ *is a set of transitions.*

The transitions from state to state of a MLTS are noted in the following way: (i) A transition (q, a, q') is denoted $q \xrightarrow{a} q'$ and is called a *discrete transition*, if $a \in \Sigma$ and $(q, a, q') \in \rightarrow_{mlts}$, (ii) A transition $(q, \boldsymbol{d}, q')$ is denoted $q \xrightarrow{\boldsymbol{d}} q'$ and is called a *delay transition*, if $\boldsymbol{d} \in \mathbb{R}_{\geq 0}^{Proc}$ and $(q, \boldsymbol{d}, q') \in \rightarrow_{mlts}$.

A run of $\mathcal{M}$ can be defined as a finite sequence of moves, where discrete and continuous transitions alternate: $\rho = q_0 \xrightarrow{\boldsymbol{d}_1} q'_0 \xrightarrow{a_1} q_1 \xrightarrow{\boldsymbol{d}_2} q'_1 \xrightarrow{a_2} q_2 \ldots q_{n-1} \xrightarrow{\boldsymbol{d}_{n-1}} q'_{n-1} \xrightarrow{a_{n-1}} q_n$, where $\forall\, 0 \leq i \leq n-1$, $q_i \in Q$, $\forall j \leq n-1$, $\boldsymbol{d}_j \in \mathbb{R}_{\geq 0}^{Proc}$, $q'_j \in Q$ and $a_j \in \Sigma$. The *multi-timed word* of ρ is $\theta = ((a_1, \boldsymbol{t}_1), (a_2, \boldsymbol{t}_2) \ldots, (a_n, \boldsymbol{t}_n))$, where $\boldsymbol{t}_i = \sum_{j=1}^{i} \boldsymbol{d}_j$. A multi-timed word θ is *accepted* by $\mathcal{M}$ iff there is a maximal initial run whose multi-timed word is θ. The *language* of $\mathcal{M}$, denoted $\mathcal{L}(\mathcal{M})$, is defined as the set of multi-timed words accepted by some run of $\mathcal{M}$. Note that

MLTS are a proper generalisation of TLTS: each TLTS can be seen as a MLTS with a single process and conversely.

For example, consider the two transition systems in Figure 3.2: a TLTS on the left ($\mathcal{M}_1$) and a MLTS on the right ($\mathcal{M}_2$) with the finite input alphabet $\Sigma = \{a, b, c\}$.

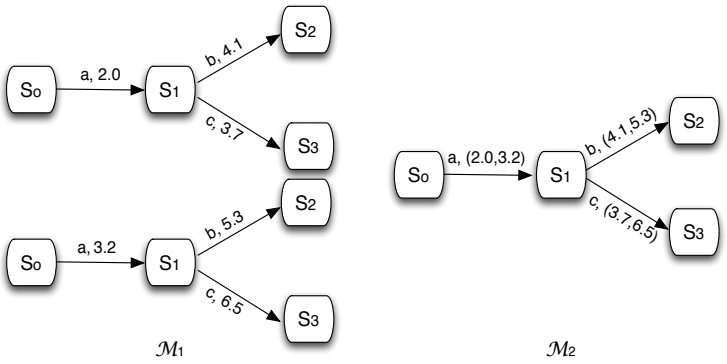

Fig. 3. Multi-Timed and Timed Labelled Transition Systems

3.3 A Multi-timed Semantics for icTA

We now present the icTA that are a formalism for describe both distributed and timed systems. The operational semantics of an icTA has been associated to a sequential notion. Here, we want to associate operational semantics of a icTA to a MLTS.

Given $\pi : X \to Proc$, a clock valuation $\nu : X \to \mathbb{R}_{\geq 0}$ and $\boldsymbol{d} \in \mathbb{R}_{\geq 0}^{Proc}$: the valuation $\nu +_\pi \boldsymbol{d}$ is defined by $(\nu +_\pi \boldsymbol{d})(x) = \nu(x) + \boldsymbol{d}_{\pi(x)}$ for all $x \in X$.

Definition 7. *Let $\mathcal{A}$ be an icTA and $\tau \in Rates$. Our multi-timed semantics of the icTA $\mathcal{A}$ is given by a MLTS over Proc, denoted by MLTS($\mathcal{A}$, τ) = (Q,q_0,Σ,$\to_{mlts}$). The set of states Q consists of triples composed of a location, a clock valuation and lastly the reference time: $Q = \{(s, \nu, t) \in S \times \mathbb{R}_{\geq 0}^X \times \mathbb{R}_{\geq 0} \mid \nu \models I(s)\}$. The starting state is $q_0 = (s_0, \nu_0, 0)$, where ν_0 is the valuation that assigns 0 to all the clocks. Σ is the alphabet of $\mathcal{A}$. The transition relation $\to_{mlts}$ is defined by:*

(i) A transition $(q_i, \boldsymbol{d}, q_i')$ is denoted $q_i \xrightarrow{\boldsymbol{d}} q_i'$, and is called a *delay transition*, where $q_i = (s_i, \nu_i, t_i)$, $q_i' = (s_i, \nu_i +_\pi \boldsymbol{d}, t_{i+1})$, $\boldsymbol{d} = \tau(t_{i+1}) - \tau(t_i)$ and $\forall t \in [t_i, t_{i+1}] : \nu_i +_\pi (\tau(t) - \tau(t_i)) \models I(s_i)$.

(ii) A transition (q_i, a, q_{i+1}) is denoted $q_i \xrightarrow{a} q_{i+1}$, and is called a *discrete transition*, where $q_i = (s_i, \nu_i, t_i)$, $q_{i+1} = (s_{i+1}, \nu_{i+1}, t_{i+1})$, $a \in \Sigma$, there exists a transition $(s_i, a, \phi, Y, s_{i+1}) \in \to_{\mathsf{ic}}$, such that $\nu_i \models \phi$, $\nu_{i+1} = \nu_i[Y \to 0]$, $\nu_{i+1} \models I(s_{i+1})$, $t_i = t_{i+1}$.

In Definition 5, we have introduced a multi-timed semantics for icTA, following ideas of [2]. Meanwhile, Akshay et al [2] by contrast defines a run of an icTA $\mathcal{A}$ for $\tau \in Rates$ with a sequential semantics as a sequence $(s_1, \nu_1) \xrightarrow{t_1, a_1} (s_2, \nu_2) \xrightarrow{t_2, a_2}$

$(s_2, \nu_3) \ldots (s_{n-1}, \nu_{n-1}) \xrightarrow{t_{n-1}, a_{n-1}} (s_n, \nu_n)$ where $\forall 1 \leq i \leq n$, $s_i \in S$ and $\forall j \leq n-1$, $t_j \in \mathbb{R}_{\geq 0}$ and $a_j \in \Sigma$. A run of an icTA $\mathcal{A}$ for $\tau \in Rates$ with our multi-timed semantics is an initial path in $\mathsf{MLTS}(\mathcal{A}, \tau)$ where discrete and continuous transition alternate. A multi-timed word is accepted by $\mathcal{A}$ for $\tau \in Rates$ if it is accepted by $\mathsf{MLTS}(\mathcal{A}, \tau)$. The multi-timed language accepted by $\mathcal{A}$ for τ with our multi-timed semantics is denoted as $\mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)$. The multi-timed language generated by $\mathsf{MLTS}(\mathcal{A}, \tau)$ is denoted as $\mathcal{L}(\mathsf{MLTS}(\mathcal{A}, \tau))$ (and $\mathcal{L}\ (\mathsf{MLTS}(\mathcal{A}, \tau)) = \mathcal{L}\ (\mathcal{A}, \tau)$). Anyway, we can find an obvious correspondence between the multi-timed semantics and sequential semantics: $(s_i, \nu_i, t_i) \xrightarrow{\boldsymbol{d}} (s_i, \nu_i +_\pi \boldsymbol{d}, t_i + 1) \xrightarrow{a} (s_{i+1}, \nu_{i+1}, t_{i+1})$ iff $(s_i, \nu_i) \xrightarrow{t_{i+1}, a} (s_{i+1}, \nu_{i+1})$ and $\boldsymbol{d} = \tau(t_{i+1}) - \tau(t_i)$.

Example 3. The Figure 4 shows an icTA $\mathcal{M}$ with the finite input alphabet $\Sigma = \{a, b, c, d\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau = (2t, t)$ i.e. $\tau_p(t) = 2t$ and $\tau_q(t) = t$. A run of $\mathcal{M}$ on multi-timed word $\theta = ((a, (2.0, 1.0))(b, (3.0, 1.5))(c, (4.2, 2.1))(d, (6.0, 3.0)))$ is given by $\rho\ (s_0, [x^p = 0.0, y^p = 0.0], 0.0) \xrightarrow{(2.0,1.0)} (s_0, [x^p = 2.0, y^p = 1.0], 1.0) \xrightarrow{a} (s_0, [x^p = 2.0, y^p = 0.0], 1.0) \xrightarrow{(1.0,0.5)} (s_1, [x^p = 3.0, y^p = 0.5], 0.5) \xrightarrow{b} (s_1, [x^p = 3.0, y^p = 0.5], 0.5) \xrightarrow{(0.6,0.5)} (s_1, [x^p = 4.2, y^p = 1.1], 0.6) \xrightarrow{c} (s_1, [x^p = 4.2, y^p = 0.0], 0.6) \xrightarrow{(0.6,0.9)} (s_1, [x^p = 6.0, y^p = 0.9], 0.9) \xrightarrow{c} (s_1, [x^p = 0.0, y^p = 0.9], 0.9)$.

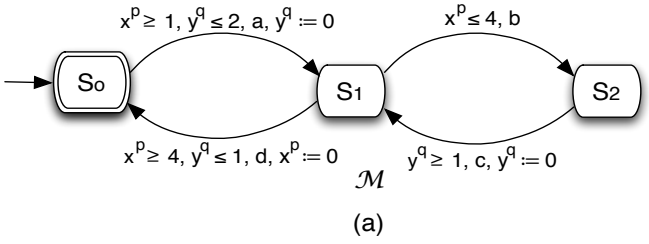

Fig. 4. An icTA $\mathcal{M}$

Theorem 1. *Let $\mathcal{A}$ be an icTA. Then, for every $\tau \in Rates$, $\tau(\mathcal{L}(\mathcal{A}, \tau)) = \mathcal{L}(MLTS(\mathcal{A}, \tau))$.*

Proof. The proof consists of two steps, first showing, $\forall\, \tau \in Rates$, $\mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A}, \tau)) \subseteq \tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau))$ and then showing $\forall\, \tau \in Rates$, $\tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)) \subseteq \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A}, \tau))$.

(i) $\forall\ \tau \in Rates$, $\tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)) \subseteq \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A}, \tau))$:

$$\begin{aligned} \forall\ \tau\ \in\ Rates,\ \tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)) &= \tau(\{(\theta, d) \mid (\theta, d) \in \mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)\}) \\ &= \{\tau(\theta, d) \mid (\theta, d) \in \mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)\} \\ &= \{(\theta, \tau(d)) \mid (\theta, d) \in \mathcal{L}_{\mathcal{M}}(\mathcal{A}, \tau)\} \\ &\subseteq \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A}, \tau)) \end{aligned}$$

(ii) $\forall\ \tau \in Rates,\ \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A},\tau) \subseteq \tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A},\tau))$:

$$\begin{aligned}\forall\ \tau,\ \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A},\tau)) &= \{(\theta,\boldsymbol{d}) \mid (\theta,\boldsymbol{d}) \in \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A},\tau))\}\\ &= \{(\theta,\boldsymbol{d}) \mid (\theta,\boldsymbol{d}) \in \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A},\tau))\}\\ &= \{(\theta,\boldsymbol{d}) \mid (\theta,\boldsymbol{d}) \in \mathcal{L}_{\mathcal{M}}(\mathsf{MLTS}(\mathcal{A},\tau))\}\\ &\subseteq \tau(\mathcal{L}_{\mathcal{M}}(\mathcal{A},\tau))\end{aligned}$$

□

4 Multi-Timed Bisimulation

From a distributed approach, a DTS consist of several processes with their associated local clocks that are not running at the same rate. Thus, in order to formalize preservation of distributed timed behavior, we extend the classical definition of timed bisimulation [14] towards a multi-timed semantics. Our motivation for extending the classical definition of timed bisimulation is twofold: first, efficient algorithms checking for timed and time-abstract bisimulation have been discovered [20][26]. Nonetheless, these algorithms depend on sequential semantics (i.e, TLTS and TA). Second, verifying the preservation of distributed timed behavior in DTS could be used to avoid the combinatorial explosion of the size of the model due to the composition of the processes.

4.1 Strong Multi-Timed Bisimulation

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MLTS over the same set of actions Σ. Let $Q_{\mathcal{M}_1}$ (resp., $Q_{\mathcal{M}_2}$) be the set of states of $\mathcal{M}_1$ (resp., $\mathcal{M}_2$). Let $\mathcal{R}$ be a binary relation over $Q_{\mathcal{M}_1} \times Q_{\mathcal{M}_2}$. We say that $\mathcal{R}$ is a strong multi-timed bisimulation whenever the following transfer property holds (note that technically this is simply strong bisimulation over $\Sigma \uplus \mathbb{R}_{\geq 0}^{Proc}$):

Definition 8. *A strong multi-timed bisimulation over MLTS $\mathcal{M}_1$, $\mathcal{M}_2$ is a binary relation $\mathcal{R} \subseteq Q_{\mathcal{M}_1} \times Q_{\mathcal{M}_2}$ such that, for all $q_{\mathcal{M}_1}\mathcal{R}q_{\mathcal{M}_2}$, the following holds:*

(i) For every $a \in \Sigma$ and for every discrete transition $q_{\mathcal{M}_1} \xrightarrow{a}_{\mathcal{M}_1} q'_{\mathcal{M}_1}$, there exists a matching discrete transition $q_{\mathcal{M}_2} \xrightarrow{a}_{\mathcal{M}_2} q'_{\mathcal{M}_2}$ such that $q'_{\mathcal{M}_1}\mathcal{R}q'_{\mathcal{M}_2}$ and symmetrically.

(ii) For every $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{R}_{\geq 0}^{Proc}$, for every delay transition $q_{\mathcal{M}_1} \xrightarrow{\boldsymbol{d}}_{\mathcal{M}_1} q'_{\mathcal{M}_1}$, there exists a matching delay transition $q_{\mathcal{M}_2} \xrightarrow{\boldsymbol{d}}_{\mathcal{M}_2} q'_{\mathcal{M}_2}$ such that $q'_{\mathcal{M}_1}\mathcal{R}q'_{\mathcal{M}_2}$ and symmetrically.

Two states $q_{\mathcal{M}_1}$ and $q_{\mathcal{M}_2}$ are multi-timed bisimilar, written $q_{\mathcal{M}_1} \approx q_{\mathcal{M}_2}$, iff there is a multi-timed bisimulation that relates them. $\mathcal{M}_1$ and $\mathcal{M}_2$ are multi-timed bisimilar, written $\mathcal{M}_1 \approx \mathcal{M}_2$, if there exists a multi-timed bisimulation relation $\mathcal{R}$ over $\mathcal{M}_1$ and $\mathcal{M}_2$ containing the pair of initial states.

As a consequence of Definition 4, the notion of multi-timed bisimulation extends to icTA and we have the following definition:

Definition 9. *Let $\mathcal{A}$ and $\mathcal{B}$ be two icTA. We say the automata $\mathcal{A}$ and $\mathcal{B}$ are multi-timed bisimilar, denoted $\mathcal{A} \approx \mathcal{B}$, iff $\forall\, \tau \in Rates$ MLTS($\mathcal{A}, \tau$) $\approx$ MLTS($\mathcal{B}, \tau$).*

When there is only one process, i.e. $Proc = \{q\}$, the multi-timed bisimulation is the usual timed bisimulation. Consider the two icTA $\mathcal{A}_p$ (left) and $\mathcal{A}_q$ (right) in Figure 5 with the finite input alphabet $\Sigma = \{a\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau = (t^2, 3t)$ i.e. $\tau_p(t) = t^2$ and $\tau_q(t) = 3t$. $\mathcal{A}_p$ and $\mathcal{A}_q$ in Figure 5 depicts an icTA. $\mathcal{A}_p$ performs nondeterministically the transition with the guard $x^p \leq 2$, the action a, resets clock x^p to 0 and enters location s_1. Similarly, $\mathcal{A}_q$ performs nondeterministically the transitions with the guard $y^q \leq 2$, the action a, resets clock y^q to 0 and enters location t_1. We will show that these icTA are not multi-timed bisimilar (Definition 6): We have $(s_0, [x^p = 0], 0)$ in MLTS$(\mathcal{A}_p, \tau_p)$ and $(t_0, [y^q = 0], 0)$ since $\mathcal{A}_p$ can run the delay transition $(s_0, [x^p = 0], 0) \xrightarrow{(1,3)} (s_0, [x^p = 1.0], 1)$ and $\mathcal{A}_q$ in MLTS$(\mathcal{A}_q, \tau_q)$. We have $(s_0, [x^p = 0], 0) \not\approx (t_0, [y^q = 0], 0)$ can only match this transition with $(t_0, [y^q = 0], 0) \xrightarrow{(1,3)} (t_0, [y^q = 3], 1)$. From these states MLTS$(\mathcal{A}_p, \tau_p)$ can fire a while MLTS$(\mathcal{A}_q, \tau_q)$ cannot.

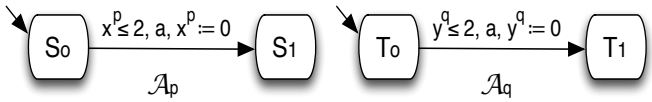

Fig. 5. An example of Multi-timed Bisimulation

Proposition 1. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MLTS over the set of actions Σ. For any $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_1 \parallel \mathcal{M}_2 \approx \mathcal{M}_2 \parallel \mathcal{M}_1$.*

Proof. The proof of this proposition consists in showing that each transition of $\mathcal{M}_1 \parallel \mathcal{M}_2$ can be found in $\mathcal{M}_2 \parallel \mathcal{M}_1$ and vice versa, where $\mathcal{R}$ is obviously the swapping i.e., $(q_{\mathcal{M}_1}, q_{\mathcal{M}_2})\ \mathcal{R}\ (q_{\mathcal{M}_2}, q_{\mathcal{M}_1})$. Based on the MLTS composition, there exists two types of transitions on the resulting system. Let $\mathcal{R} = \{(q_{\mathcal{M}_1}, q_{\mathcal{M}_2}) | (q_{\mathcal{M}_2}, q_{\mathcal{M}_1}) \in Q_{\mathcal{M}_2 \parallel \mathcal{M}_1}\}$. It directly follows from the definition of parallel composition in MLTS (Definition 35) that:

(i) For any discrete transition $(q_{\mathcal{M}_1}, q_{\mathcal{M}_2}) \xrightarrow{a}_{\mathcal{M}_1 \parallel \mathcal{M}_2} (q'_{\mathcal{M}_1}, q'_{\mathcal{M}_2})$ with $a \in \Sigma$, there exists a corresponding transition $(q_{\mathcal{M}_2}, q_{\mathcal{M}_1}) \xrightarrow{a}_{\mathcal{M}_2 \parallel \mathcal{M}_1} (q'_{\mathcal{M}_2}, q'_{\mathcal{M}_1})$ with $((q_{\mathcal{M}_1}, q_{\mathcal{M}_2}), (q_{\mathcal{M}_2}, q_{\mathcal{M}_1})) \in \mathcal{R}$.

(ii) For any delay transition $(q_{\mathcal{M}_1}, q_{\mathcal{M}_2}) \xrightarrow{d}_{\mathcal{M}_1 \parallel \mathcal{M}_2} (q'_{\mathcal{M}_1}, q'_{\mathcal{M}_2})$ with $d \in \mathbb{R}^n_{\geq 0}$, there exists a corresponding transition $(q_{\mathcal{M}_2}, q_{\mathcal{M}_1}) \xrightarrow{d}_{\mathcal{M}_2 \parallel \mathcal{M}_1} (q'_{\mathcal{M}_2}, q'_{\mathcal{M}_1})$ with $((q_{\mathcal{M}_1}, q_{\mathcal{M}_2}), (q_{\mathcal{M}_2}, q_{\mathcal{M}_1})) \in \mathcal{R}$.

Since every initial state $(q^0_{\mathcal{M}_1}, q^0_{\mathcal{M}_2})$ of $\mathcal{M}_1 \parallel \mathcal{M}_2$ has a match $(q^0_{\mathcal{M}_2}, q^0_{\mathcal{M}_1})$ in the initial states of $\mathcal{M}_2 \parallel \mathcal{M}_1$, and $((q^0_{\mathcal{M}_1}, q^0_{\mathcal{M}_2}), (q^0_{\mathcal{M}_2}, q^0_{\mathcal{M}_1}) \in \mathcal{R}$. Therefore, $\mathcal{R}$ is a

bisimulation for $\mathcal{M}_1 \parallel \mathcal{M}_2 \approx \mathcal{M}_2 \parallel \mathcal{M}_1$. Finally, by following a similar steps, we could show that $\mathcal{M}_2 \parallel \mathcal{M}_1 \approx \mathcal{M}_1 \parallel \mathcal{M}_2$.

□

In the context of multi-timed bisimulation and MLTS, compositionality [12] is captured by the following definition:

Definition 10 (Compositionality). *A binary relation $\approx$ between two MLTS $\mathcal{M}_1$, $\mathcal{M}_2$ is compositional if $\mathcal{M}_1 \approx \mathcal{M}_2$ and $\mathcal{M}_3 \approx \mathcal{M}_4$ implies $\mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_4$.*

Proposition 2. *Let $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ be three MLTS over the set of actions Σ. For any $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$, $\approx$ is compositional if and only if $\mathcal{M}_1 \approx \mathcal{M}_2 \Rightarrow \mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_3$ (invariant under composition).*

□

Proof. The proof of this proposition consists in showing for the sufficient direction that $\mathcal{M}_1 \approx \mathcal{M}_2 \Rightarrow \mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_3$. Assume that $\approx$ is compositional. Since $\approx$ is reflexive $\mathcal{M}_3 \approx \mathcal{M}_3$ holds. Using the definition of compositionality, $\mathcal{M}_1 \approx \mathcal{M}_2$ and $\mathcal{M}_3 \approx \mathcal{M}_3$ imply $\mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_3$. For the necessary direction, assume that $\approx$ is invariant under composition. Then $\mathcal{M}_1 \approx \mathcal{M}_2$ implies $\mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_3$. Also, $\mathcal{M}_3 \approx \mathcal{M}_4$ and commutativity of composition implies $\mathcal{M}_2 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_4$, and by transitivity $\mathcal{M}_1 \parallel \mathcal{M}_3 \approx \mathcal{M}_2 \parallel \mathcal{M}_4$.

□

4.2 Decidability

There are two popular symbolic representations of a set of clock valuations: the region-based [5] and the zone-based [9] representations. Roughly speaking, regions and zones are abstract representations of (infinite) sets of valuations, but zones are larger for extrapolated zones. Inspired by [20], we show that for given icTA $\mathcal{A}$, $\mathcal{B}$, checking whether $\mathcal{A} \approx \mathcal{B}$ is decidable via a suitable zone graph [20]. In order to define the notion of clock zone over a set of clocks X, we need to consider the set $\Phi^+(X)$ of extended clock constraints.

Definition 11. *A clock constraint ϕ is a conjunction of comparisons of a clock with a constant c, given by the following grammar, where ϕ ranges over $\Phi^+(X)$, $x_i, x_j \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, >, \leq, \geq, =\}$: $\phi ::= true \mid x_i \sim c \mid x_i - x_j \sim c \mid \phi_1 \wedge \phi_2$. A clock constraint of the form $x_i - x_j \sim c$ is called diagonal constraint and x_i, x_j must belong to the same process. The notion of satisfaction of a clock constraint $\phi \in \Phi^+(X)$ by a valuation is given by the clause $\nu \models x_i - x_j \sim c$ iff $\nu(x_i) - \nu(x_j) \sim c$.*

Formally, a clock zone $\mathcal{Z}$ is a conjunction of extended clock constraints ($\phi \in \Phi^+(X)$) with inequalities of clock differences. Its semantics is the set of clock valuations that satisfy it $[\![\mathcal{Z}]\!] = \{\nu \mid \nu \models \phi\}$. We omit the semantics brackets when obvious.

For any clock zones $\mathcal{Z}$, $\mathcal{Z}'$ and finite set of clocks X, the semantics of the intersection, clock reset, inverse clock reset, time successor and time predecessor events on clock zone can be defined as:

1. $\mathcal{Z} \cap \mathcal{Z}' = \{\nu \mid \nu \in \mathcal{Z} \wedge \nu \in \mathcal{Z}'\}$
2. $\mathcal{Z}\downarrow_x = \{\nu[x \to 0] \mid \nu \in \mathcal{Z} \text{ and } x \in X\}$
3. $\mathcal{Z}\uparrow_x = \{\nu \mid \nu[x \to 0] \in \mathcal{Z} \text{ and } x \in X\}$
4. $\mathcal{Z}\uparrow = \{\nu + \boldsymbol{d} \mid \nu \in \mathcal{Z} \text{ and } \boldsymbol{d} \in \mathbb{R}_{>0}^{Proc}\}$
5. $\mathcal{Z}\downarrow = \{\nu - \boldsymbol{d} \mid \nu \in \mathcal{Z} \text{ and } \boldsymbol{d} \in \mathbb{R}_{>0}^{Proc}\}$

Notice that the operations can be defined syntactically a clock zones as follows [4]: A zone graph is similar to a region graph [5] with the difference that each node consists of pair (called a zone) of a timed automaton location s and a clock zone $\mathcal{Z}$ (*i.e.* $q = (s, \mathcal{Z})$). For $q = (s, \mathcal{Z})$, we write $(s', \nu) \in q$ if $s = s'$ and $\nu \in \mathcal{Z}$, indicating that a state is included in a zone. Due to their convexity, clock zones are easy to manipulate in practice. A common data structure used for the representation of clock zones is a Difference bound matrices (DBMs)[9][3]. A DBMs for a set $C = \{x_1, x_2, \cdots, x_n\}$ of n clocks is an $(n+1)$ square matrix $\mathcal{D}_{ij}$ where an extra variable x_0 is introduced such that the value of x_0 is always 0. An element $\mathcal{D}$ is of the form $(d_{ij}, \prec)$ where $\prec \in \{<, \leq\}$ such that $x_i x_j \prec d_{ij}$. Thus an entry d_{i0} denotes the constraint $x_i x_0 \prec d_{i0}$ which is equivalent to $x_i \prec d_{i0}$.

Analogously, we can write $(s, \mathcal{Z}) \subseteq (s', \mathcal{Z}')$ to indicate that $s = s'$ and $\mathcal{Z} \subseteq \mathcal{Z}'$. We will use the notation $\mathsf{Action}(e)$ to denote the action a of the edge e. Furthermore, we extend the satisfaction relation to regions in the following way:

Definition 12. *Let $(s, \mathcal{Z})$ be a zone and $e = (s, a, \phi, Y, s') \in \to_{icta}$ be a transition of an icTA $\mathcal{A}$, then $post(\mathcal{Z}, e) = \{\nu' \mid \exists \nu \in \mathcal{Z}, \exists t \in \mathbb{R}_{\geq 0}, (s, \nu, t) \xrightarrow{e}_{mlts(\mathcal{A})} (s', \nu', t)\}$ is the set of valuations than can reach $(s, \mathcal{Z})$ by taking the transition e.*

Definition 13. *Let $(s, \mathcal{Z}')$ be a zone and $e = (s, a, \phi, Y, s') \in \to_{icta}$ be a transition of an icTA $\mathcal{A}$, then $pred(\mathcal{Z}', e) = \{\nu \mid \exists \nu' \in \mathcal{Z}', \exists t \in \mathbb{R}_{\geq 0}, (s, \nu, t) \xrightarrow{e}_{mlts(\mathcal{A})} (s', \nu', t)\}$ is the set of valuations than can reach $(s, \mathcal{Z}')$ by executing the transition e.*

Intuitively, we can say that the set $(s', \mathsf{post}(\mathcal{Z}, e))$ describes the discrete successor of the zone $(s, \mathcal{Z})$ under the transition e, and the set $(s, \mathsf{pred}(\mathcal{Z}', e))$ describes the discrete predecessor of the zone $(s', \mathcal{Z}')$ under the transition e. The set $\mathsf{post}(\mathcal{Z}, e)$ can be obtained using the operations $\downarrow_Y$ (clock reset) and the standard intersection on clock zones as follows:

$$\mathsf{post}(\mathcal{Z}, e) = ((\mathcal{Z} \cap (\phi \cap I(s))\downarrow_Y \cap I(s'))$$

The set $\mathsf{pred}(\mathcal{Z}', e)$ can be obtained using the operations $\downarrow$ (predecessor) and the standard intersection on clock zones as follows:

$$\mathsf{pred}(\mathcal{Z}', e) = ((\mathcal{Z}'\uparrow_Y \cap \phi) \cap I(s))$$

The sets $\mathsf{post}(\mathcal{Z}, e)$ and $\mathsf{pred}(\mathcal{Z}', e)$ are also clock zones.

Definition 14 (Symbolic Multi-timed Zone Graph). *Given an icTA $\mathcal{A} = (\Sigma, X, S, s_0, \to_{icta}, I, F, \pi)$, its symbolic multi-timed zone graph ($ZG(\mathcal{A})$) is a transition system $ZG(\mathcal{A}) = (Q, q_0, (\Sigma \cup \{\uparrow\}), \to_{ZG})$, where :*

1. *Q consists of pairs $q = (s, \mathcal{Z})$ where $s \in S$, and $\mathcal{Z} \in \Phi^+(X)$ is a clock zone with $\mathcal{Z} \subseteq I(s)$.*
2. *$q_0 \in Q$ is the initial zone $q_0 = (s_0, \mathcal{Z}_0)$ with $\mathcal{Z}_0 = [\![\bigwedge_{x \in X} x = 0]\!]$.*
3. *Σ is the set of labels of $\mathcal{A}$.*
4. *$\rightarrow_{ZG} \subseteq Q \times (\rightarrow_{icta} \cup \{\uparrow\}) \times Q$ is a set of transitions, where each transition in $ZG(\mathcal{A})$ is a labelled by a transition $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$, where s and s' are the source and target locations, ϕ is a clock constraint defining the guard of the transition, a is the action of the edge and Y is the set of clocks to be reset by the transition in the icTA $\mathcal{A}$. For each $e \in \Sigma$, transitions are defined by the rules:*

   (i) For every $e = (s, a, \phi, Y, s')$ and clock zone $\mathcal{Z}$, there exists a discrete transition (q, e, q'), where $q = (s, \mathcal{Z}) \xrightarrow{e}_{ZG} q' = (s', post(\mathcal{Z}, e))$ if $post(\mathcal{Z}, e) \neq \emptyset$.

   (ii) For a clock zone $\mathcal{Z}$, there exists a delay transition $(q, \uparrow, q')$, where $q = (s, \mathcal{Z}) \xrightarrow{\uparrow}_{ZG} q' = (s, \mathcal{Z}')$ and $\mathcal{Z}' = \mathcal{Z} \uparrow \cap I(s)$.

Note that $\uparrow$ is used here as a symbol to represent symbolic positive delay transitions. Only the reachable part is constructed.

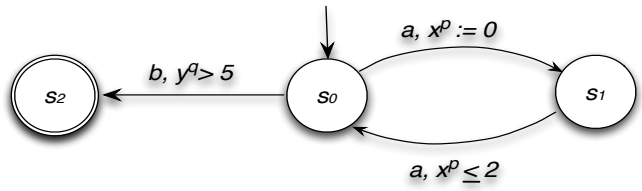

Fig. 6. An icTA $\mathcal{A}$

Example 4. Consider the icTA $\mathcal{A}$ in Figure 6 with the finite input alphabet $\Sigma = \{a, b\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$. Clock x^p ensures that the automaton returns to s_0 before 2 time units of p before coming back to s_0 or stays in s_1 forever. Clock y^q ensures that the automaton can go to the final state s_2 only after 5 time units of q. Figure 7 shows $ZG(\mathcal{A})$.

Lemma 1. *Let $(s, \mathcal{Z})$ be a zone and $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$ be a transition of an icTA $\mathcal{A}$, then $\mathcal{Z} \uparrow$, $\mathcal{Z} \uparrow_x$, $\mathcal{Z} \downarrow$, $post(\mathcal{Z}, e)$ and $pred(\mathcal{Z}', e)$ are also zones.*

Proof. The proof follows from the fact that zones operations preserve convexity. □

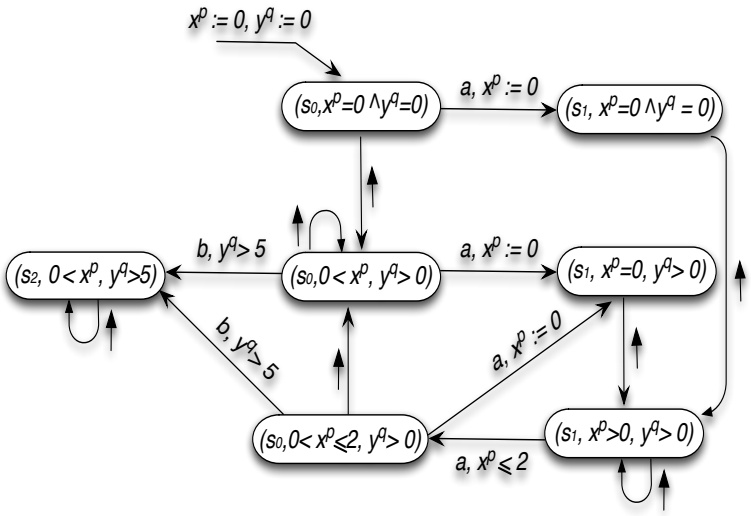

Fig. 7. The zone graph for the automaton in Figure 6

Algorithm 1: Reachable Multi-timed Zone Graph

Input : An icTA $\mathcal{C} = (\Sigma, X, S, s_0, \rightarrow_{icta}, I, F, \pi)$.
Output: A reachable zone graph $\mathsf{ZG}(\mathcal{C}) = (Q, q_0, \Sigma, \rightarrow_{\mathsf{ZG}})$.

```
// s ∈ S is a location of C, Z_{1≤i≤4} are clock zones, M is a set of clock zones.
// T_ZG is a set of transitions (i.e. →_ZG= T_ZG), E_ZG is a set of labels.
// D and Q are a set of pairs S × Z, D is the set of open states.
Function BuildSymbZoneGraph(C)
    q_0 = (s_0, Z_0) such that for all x ∈ X and ν ∈ Z_0, ν(x) = 0 ;
    Q, D ← {q_0}, T_ZG ← ∅, M ← ∅ ;
    while D ≠ ∅ do
        Choose and Remove (s, Z_1) from D ;
        M ← {Z_1} ;
        for each transition e = (s, a, φ, Y, s') ∈→_icta such that Z_1 ∧ φ ≠ ∅ do
            if exists Z_3 ∈ M then
                // Z_2 is the successor such that Z_2 ∧ I(s') ≠ ∅
                Z_2 ← Extra^+_{LU_(s)}(post(Z_1, e)) ;
                E_ZG ← E_ZG ∪ {e} ;
                if exists (s', Z_4) ∈ Q such that Z_2 ⊆ Z_4 then
                    T_ZG ← T_ZG ∪ {(s, Z_1) -e->_ZG (s', Z_4)} ;
                else
                    T_ZG ← T_ZG ∪ {(s, Z_1) -e->_ZG (s', Z_2)} ;
                    Q ← Q ∪ {(s', Z_2)}, D ← D ∪ {(s', Z_2)} ;
                end
            else
                continue ;
            end
        end
        Z_2 ← Z_1 ↑ ∧ I(s) ;
        if exists (s, Z_3) ∈ Q such that Z_2 ⊆ Z_3 then
            T_ZG ← T_ZG ∪ {(s, Z_1) -↑->_ZG (s, Z_3)} ;
        else
            T_ZG ← T_ZG ∪ {(s, Z_1) -↑->_ZG (s', Z_2)} ;
            Q ← Q ∪ {(s, Z_2)}, D ← D ∪ {(s, Z_2)} ;
        end
    end
    return (Q, q_0, Σ, →_ZG) ;
end
```

Multi-timed Zone Graph Algorithm: In algorithm 1, we use the zone's concepts obtained above to build a reachable multi-timed zone graph ($\mathsf{ZG}(\mathcal{C})$) from the parallel composition of two icTA ($\mathcal{C} = \mathcal{A} \parallel \mathcal{B}$). Algorithm 1 build a multi-timed zone graph, starting with the pair $(s_0, \mathcal{Z}_0)$ (s_0 initial location of the automaton $\mathcal{A}$ with $\mathcal{Z}_0 = [\![\bigwedge_{x \in X} x = 0]\!]$ represents the initial zone). However, the multi-timed zone graph can be infinite, because constants used in zones may grow for ever. Therefore, we incorporate to our implementation a termination condition which bounds the number of states to be generated [7][8]. We use a technique called extrapolation abstraction (known also as k - extrapolation [7]), where k is a constant supposed to be greater than the maximal constant occurring in $\mathcal{A}$ (i.e., $C_x \in \mathbb{N}$). The main idea of extrapolation abstraction is that if an atomic constraint (i.e., true, $x_i \sim c$, $x_i - x_j \sim c$) of $\mathcal{A}$ which compares two clocks (x_i *and* x_j) is not satisfied by any clock valuation of a zone, then it should not be satisfied by any clock valuation of the extrapolated one. Similarly, if all the clock valuations of a zone satisfy a difference constraint, then so should also all the clock valuations of the extrapolated one [7][8]. Among the various Max-bound and LU-bound based extrapolation abstraction [8], we choose to use LU-extrapolation abstraction [8]: $Extra_{LU}$, $Extra^{+}_{LU}$ (LU-bound), where L is the maximal lower bound and U is the maximal upper bounds. For every zone $q = (s, \mathcal{Z})$ of a $\mathsf{ZG}(\mathcal{A})$, there are bound functions LU and the symbolic zone graph using $Extra^{+}_{LU_{(s)}}$. Then, we build zones of the form $q_{\mathsf{ZG}} = (s, Extra^{+}_{LU_{(s)}}(\mathsf{post}(\mathcal{Z}, e))$.

The operator $Extra_{LU}$ has been the most used in different implementations of zones, but some improvement has been made to this operator to get better LU-bound to $Extra^{+}_{LU}$. A naive way to choose L bounds is to take for each clock the maximum constant appearing in a guard anywhere in the automaton that lower-bounds the clock. Similarly we choose U from among the upper bounded guards that occur anywhere in the automaton. In [8], instead of considering global bound L, U for all locations in an automaton, they use different bound for each location of the automaton (LU to denote the two bound functions L and U). For every zone $q = (s, \mathcal{Z})$ of a $\mathsf{ZG}(\mathcal{A})$, there are bound functions LU and the symbolic zone graph using $Extra^{+}_{LU_{(s)}}$. Initially, we create a multi-timed zone graph, where we use $Extra^{+}_{LU_{(s)}}$ method in the build of the graph. We will build states of the form $q_{\mathsf{ZG}} = (s, Extra^{+}_{LU_{(s)}}(\mathsf{post}(\mathcal{Z}, e))$.

Lemma 2 (Completeness). *Let* $\theta = (s_0, \nu_0, t_0) \xrightarrow{\boldsymbol{d}_0, a_0} (s_1, \nu_1, t_1) \xrightarrow{\boldsymbol{d}_1, a_1} \ldots \xrightarrow{\boldsymbol{d}_{n-1}, a_{n-1}} (s_n, \nu_n, t_n)$ *be a run of* $\mathsf{MLTS}(\mathcal{A}, \tau)$*, where* $\tau \in$ *Rates. Then, for all state* (s_i, ν_i, t_i) *where* $0 \leq i \leq n$*, there exists a symbolic zone* $(s_i, \mathcal{Z}_i)$ *added in* Q *such that* $\nu_i \in \mathcal{Z}_i$*.*

Proof. We proceed by induction on the length of the run leading to (s_i, ν_i, t_i). Base case: We know that $\nu_0 \in \mathcal{Z}_0$. The zone $(s_0, \mathcal{Z}_0)$ is added to D and Q in line 9. For the base case, $(s_0, \mathcal{Z}_0)$ is the required zone.

Induction case: Assume that for all $0 \leq i \leq m$, there exists $(s_i, \mathcal{Z}_i)$ in Q such that $\nu_i \in \mathcal{Z}_i$. We will now show that there exists $(s_{m+1}, \mathcal{Z}_{m+1})$ in Q such that $\nu_{m+1} \in \mathcal{Z}_{m+1}$. By the induction hypothesis, we have $(s_m, \mathcal{Z}_m)$ in Q such

that $\nu_m \in \mathcal{Z}_m$. Consider the transition $(s_m, \nu_m, t_m) \xrightarrow{\boldsymbol{d}_m, a_m} (s_{m+1}, \nu_{m+1}, t_{m+1})$ of the run θ. As $(s_m, \mathcal{Z}_m)$ is in Q, the discrete transition $\xrightarrow{e_m}_{\mathsf{ZG}}$ with $e_m = (s_m, a_m, \phi_m, Y_m, s_{m+1}) \in \rightarrow_{icta}$ has been considered in the for loop of line 13. As $(s_m, \mathcal{Z}_m)$ is in Q, the delay transition $\xrightarrow{\uparrow}_{\mathsf{ZG}}$ with $\uparrow = d_m \in \mathbb{R}^n_{\geq}$ has been considered in the for loop of line 29. Let $(s_m, \mathcal{Z}_m) \xrightarrow{e_m}_{\mathsf{ZG}} (s_{m+1}, \mathcal{Z}_{m+1})$ be the discrete transition in the zone graph in lines 19, 21. Let $(s_m, \mathcal{Z}_m) \xrightarrow{d_m}_{\mathsf{ZG}} (s_{m+1}, \mathcal{Z}_{m+1})$ be the delay transition in the zone graph in lines 31, 33. By definition of the symbolic transition, $\nu_{m+1} \in \mathcal{Z}_{m+1}$. If $(s_{m+1}, \mathcal{Z}_{m+1})$ is in Q, we are done. The only other case when $(s_{m+1}, \mathcal{Z}_{m+1})$ is not in Q is when there exists $(s_{m+1}, \mathcal{Z}^{'}_{m+1})$ in Q such that $\mathcal{Z}_{m+1} \subseteq \mathcal{Z}_{m+1}$. Therefore, $\nu_{m+1} \in \mathcal{Z}_{m+1}$ and since $(s_{m+1}, \mathcal{Z}^{'}_{m+1})$ is in Q, our required zone would be $(s_{m+1}, \mathcal{Z}^{'}_{m+1})$.

□

The above lemma tell that the algorithm 1 can overapproximate reachability of a set of zones correctly. Now, we can establish the termination of the algorithm 1, because there are finitely many $Extra^+_{LU}$ zones, so that finitely many LU-bound of zones can be explored for each state of the automaton [11][8].

Complexity Algorithm 1: Starting from the initial state $(q_0, \mathcal{Z}_0)$ it computes the successor state during the search. When a new clock zone $\mathcal{Z}' = Extra^+_{LU_{(s)}}(post(Z, e))$ and state are obtained, it is checked if there exists an already visited state $(s', Extra^+_{LU_{(s)}}(\mathcal{Z}''))$ such that $Extra^+_{LU_{(s)}}(\mathcal{Z}') \subseteq Extra^+_{LU_{(s')}}(\mathcal{Z}'')$. If there does exist one such state, the new transition is considered for the zone graph. As $Extra^+_{LU}$ is convex, this inclusion is just an inclusion check between two zones and can be done efficiently in time $O(|X|^2)$. The time complexity of this algorithm is given in terms of the number of clocks, the number of clocks and the number of transitions of the icTA: $O(|S| \times | \rightarrow_{\mathsf{icTA}} | \times |X|^2))$ where $|S|$ represent the number of states in the icTA $\mathcal{A}$, $|X|$ the number of clocks in $\mathcal{A}$ and $| \rightarrow_{\mathsf{icTA}} |$ the number of transitions in $\mathcal{A}$

Partition-Refinement Algorithm: Now, we describe a partition refinement algorithm with signature to compute the multi-timed bisimulation from a zone graph $\mathsf{ZG}(\mathcal{C}) = \mathsf{ZG}(\mathcal{A}) \parallel \mathsf{ZG}(\mathcal{B})$. Essentially, our algorithm is based on the partition refinement technique [25]. The refinement algorithm [25] partitions the state space Q into equivalent blocks (i.e., pairwise disjoint sets of states). The algorithm starts from an initial partition Π_0 that respects state labeling and the partition Π is then successively refined until Π contains only bisimilar states. The refinement is based on the fact that a bisimulation induces a stable partition. The main steps are shown in Algorithm 2.

Algorithm 2: The general partition refinement algorithm

Input : An labelled transition system $\mathcal{A} = (Q, q_0, \Sigma, \rightarrow_{\mathsf{lts}})$ and the initial partition Π_0.
Output: A coarsest partition Π.

```
// q ∈ Q is a zone of ZG(C), Π is a set of zones, Z, Z' are clock zones.
// Q is a set of pairs S × Z.
Function GeneralPartition(A), Π_0)
    // Phase I - Initial partition Π_0
    Π ← Π_0 ;
    i ← 0 ;
    Repeat
        // Phase II - Refine Π
        Π_{i+1} ← Refine(Π_i) ;
        i ← i + 1 ;
    Until Π_i = Π_{i−1} does not change;
    Return Π_i ;
end
```

Due to the fact that the runs of our zone graph ($\mathsf{ZG}(\mathcal{C})$) involve an sequence of moves, where discrete and time-elapse $\uparrow$ transitions alternate, the partition refinement algorithm has thus to deal with the following difficulties: (i) when taking a $\uparrow$ transition, where the clocks x and y are not perfectly synchronous $\pi(x) \neq \pi(y)$, it should take into consideration that the time elapse traverses continuously diagonal, vertical and horizontal time successor zones. Conversely, due to the nature of TA, where the clocks are perfectly synchronous $\pi(x) = \pi(y)$, the time elapsing traverses only continuously diagonal time successor zones. Thus, the time splitter operator presented in [26] is not applicable within our algorithm. Figure 8 presents an example : (a) a time elapsing traversing the clock zones 1 to 3, (b) a time elapsing traversing continuously diagonal, horizontal and vertical time successor zones.

Moreover, since a discrete transition results in a sequence of time elapse transitions, the discrete splitter operator presented in [26] is not applicable within our algorithm. Therefore, we need to extend the discrete and time splitter operators presented in [26]. Also, our algorithm adopts the idea of the signature-based technique [10], which assigns states to equivalence blocks according to a characterizing signature. In each refinement iteration, the set of zones are refined according to a signature. The algorithm in [10], cannot be applied in our setting in a straightforward way, as in that case untimed systems are addressed, while in our case, the time and discrete transition should be considered. Based on [10], we introduce a signature splitter operator which refine the set of zones until a fixed point is reached, which is the complete multi-timed bisimulation. Thus, we introduce the timed and discrete predecessor operators.

As we have two types of transitions (delay and discrete), there are two operations i.e. TimePred and ActionPred, which return the set of all discrete predecessors and time predecessors of states respectively.

Definition 15. *Let $z = (s, \mathcal{Z})$ and $z' = (s, \mathcal{Z}')$ be two zones, then : $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}') = \{\nu \in \mathcal{Z} \mid \exists\ \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0},\ \exists\ \tau \in Rates,\ \exists\ t, t'' \geq 0,\ \ t \leq t''$ and $\forall t', t \leq t' \leq t'',$ and $\boldsymbol{d} = \tau(t'') - \tau(t),\ (\nu +_{\pi} \boldsymbol{d}) \in\ \mathcal{Z}',$ and $\boldsymbol{d'} = \tau(t') - \tau(t)$ then $(\nu +_{\pi} \boldsymbol{d'}) \in (\mathcal{Z} \cup \mathcal{Z}')\}$ is the set of valuations in the zone $\mathcal{Z}$ from which a valuation of $\mathcal{Z}'$ can be reached through the elapsing of time, without entering any other zones besides $\mathcal{Z}$ and $\mathcal{Z}'$ (i.e., $\mathcal{Z} \cup \mathcal{Z}'$).*

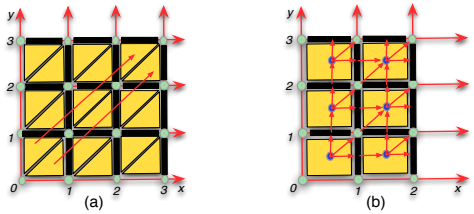

Fig. 8. (a) A time elapsing traversing 0 to 3, (b) Diag., horiz. and vert. time successors.

The definition above consider two clock zones where $\mathcal{Z}$, $\mathcal{Z}'$ are from the current partition, such that $\mathcal{Z}$ is not pre-stable with regard to $\mathcal{Z}'$, then the $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$ operator splitting $\mathcal{Z}$ with regard to $\mathcal{Z}'$ such that each newly created clock zones obtained by splitting $\mathcal{Z}$ is pre-stable with regard to $\mathcal{Z}'$.

Definition 16. *Let $z = (s, \mathcal{Z})$ and $z' = (s', \mathcal{Z}')$ be two zones from the current partition and let $e = (s, a, \phi, Y, s')$ be an action then : $\mathsf{ActionPred}_e(\mathcal{Z}, \mathcal{Z}') = \{\nu \in \mathcal{Z} \mid \nu \models \phi$ and $\nu[Y \to 0] \in \mathcal{Z}'\}$ is the set of valuations in the clock zone $\mathcal{Z}$, which lead to a valuation in $\mathcal{Z}'$ when the clock reset $Y \subseteq X$ is applied to it.*

The definition above considers two zones such that $e = (s, a, \phi, Y, s') \in \Sigma$ and $\mathcal{Z}$ is not pre-stable with regard to $\mathcal{Z}'$, then the $\mathsf{ActionPred}_e(\mathcal{Z}, \mathcal{Z}')$ operator is used in backward analysis of the timed automaton, where e is a transition in $\mathsf{ZG}(\mathcal{A})$ and s and s' are the source and target locations.

Lemma 3 ([26]). *Let $z = (s, \mathcal{Z})$, $z' = (s, \mathcal{Z}') \in Q$ be two zones, then $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$ is a clock zone.*

Our proof follows the same lines as the proof of [26]: we also show that $\mathsf{TimePred}_{\uparrow}$ is convex.

Proof. Let $z = (s, \mathcal{Z})$, $z' = (s, \mathcal{Z}')$ and $\mathcal{Z}'' = \mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$. According to ZG semantics this equivalent to show that if $\nu_1, \nu_2 \in \mathcal{Z}''$ then $\nu = k\nu_1 + (1-k)\nu_2 \in \mathcal{Z}''$, for $0 \leq k \leq 1$. $\nu_1, \nu_2 \in \mathcal{Z}''$ implies that $\nu_1, \nu_2 \in \mathcal{Z}$ and $\exists\ \boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathbb{R}^{Proc}_{>0}$ such that $\nu_1 +_\pi \boldsymbol{d}_1$, $\nu_2 +_\pi \boldsymbol{d}_2 \in \mathcal{Z}'$ and $\exists\ t_1, t_2 \geq 0$, $t_1 \leq t_2$, $\forall\ t'$, $t_1 \leq t' \leq t_2$, $\exists\ \tau_1, \tau_2 \in \mathit{Rates}$ and $\tau_1 \neq \tau_2$, $\boldsymbol{d}_1 = \tau_1(t_2) - \tau_1(t_1)$ and $\boldsymbol{d}_2 = \tau_2(t_1) - \tau_2(t')$ then $\nu_1 +_\pi \boldsymbol{d}_1$, $\nu_2 +_\pi \boldsymbol{d}_2 \in (\mathcal{Z} \cup \mathcal{Z}')$. Let $\boldsymbol{d} = k\boldsymbol{d}_1 + (1-k)\boldsymbol{d}_2$, then $\nu +_\pi \boldsymbol{d} = k(\nu_1 +_\pi \boldsymbol{d}_1) + (1-k)(\nu_2 +_\pi \boldsymbol{d}_2)$, implying that $\nu +_\pi \boldsymbol{d} \in \mathcal{Z}'$, since $\mathcal{Z}'$ is convex. Now, we have to show that $\forall\ t'$, $t_1 \leq t' \leq t_2$, $\exists\ \tau_1, \tau_2 \in \mathit{Rates}$, $\boldsymbol{d} = \tau_1(t_2) - \tau_1(t_1)$ and $\boldsymbol{d}' = \tau_2(t_1) - \tau_2(t')$, $\nu +_\pi \boldsymbol{d}' \in (\mathcal{Z} \cup \mathcal{Z}')$. Given $\boldsymbol{d}'$, $\boldsymbol{d}$, we can write $\boldsymbol{d}'$ as $k\boldsymbol{d}_3 + (1-k)\boldsymbol{d}_4$, for some $\boldsymbol{d}_3$, $\boldsymbol{d}_1$ and $\boldsymbol{d}_4$, $\boldsymbol{d}_2$. We have $\nu_1 +_\pi \boldsymbol{d}_3$, $\nu_2 +_\pi \boldsymbol{d}_4 \in (\mathcal{Z} \cup \mathcal{Z}')$. If both $\nu_1 +_\pi \boldsymbol{d}_3$, $\nu_2 +_\pi \boldsymbol{d}_4 \in \mathcal{Z}$ or $\nu_1 +_\pi \boldsymbol{d}_3$, $\nu_2 +_\pi \boldsymbol{d}_4 \in \mathcal{Z}'$, we are done, since $\mathcal{Z}$ and $\mathcal{Z}'$ are both convex. Considerer the case $\nu_1 +_\pi \boldsymbol{d}_3 \in \mathcal{Z}$ and $\nu_2 +_\pi \boldsymbol{d}_4 \in \mathcal{Z}'$. Let g be the smallest positive real such that $\nu_2 +_\pi \boldsymbol{d}_4 - g \in \mathcal{Z}$ or $\nu_1 +_\pi \boldsymbol{d}_3 - g(1-\frac{1}{k}) \in \mathcal{Z}'$. Assume the first case, we have $\nu_1 +_\pi \boldsymbol{d}_5$, $\nu_2 +_\pi \boldsymbol{d}_6 \in \mathcal{Z}$, for $\boldsymbol{d}_6 = \boldsymbol{d}_4 - g$ and $\boldsymbol{d}_5 = \boldsymbol{d}_3 - g(1-\frac{1}{k})$. Moreover, $\boldsymbol{d}' = k\boldsymbol{d}_5 + (1-k)\boldsymbol{d}_6$, which means that $\nu +_\pi \boldsymbol{d}' = k(\nu_1 +_\pi \boldsymbol{d}_5) + (1-k)(\nu_2 +_\pi \boldsymbol{d}_6)$. By convexity of $\mathcal{Z}$, $\nu +_\pi \boldsymbol{d}' \in \mathcal{Z}$.

□

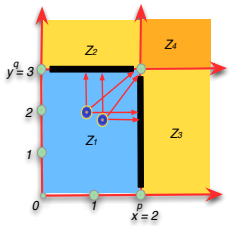

Fig. 9. Diagonal, horizontal and vertical successor zones.

Figure 9 presents the time elapsing traverses continuously time successors.

Theorem 2. *Let $\mathcal{A}$ be an icTA. Let $ZG(\mathcal{A})= (Q, q_0, \Sigma, \rightarrow_{ZG})$ be the symbolic zone graph of $\mathcal{A}$. Let $e = (s, a, \phi, Y, s')$ be a transition of $\mathcal{A}$. Let $(s, \mathcal{Z})$ and $(s', \mathcal{Z}')$ be two zones, for which $(s, \mathcal{Z}) \xrightarrow{e} (s', \mathcal{Z}') \in \rightarrow_{ZG}$ such that $\mathcal{Z} \subseteq \mathsf{ActionPred}_e(\phi, I(s'))$, then:*

1. *For discrete transition $(s, \mathcal{Z}) \xrightarrow{e} (s', \mathcal{Z}')$, $\mathcal{Z} = \mathsf{ActionPred}_e(\mathcal{Z}, \mathcal{Z}')$ iff $(s, \mathcal{Z}) \xrightarrow{e} (s', \mathcal{Z}')$ is pre-stable.*
2. *For delay transition $(s, \mathcal{Z}) \xrightarrow{\uparrow} (s, \mathcal{Z}')$, $\mathcal{Z} = \mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}\downarrow)$ iff $(s, \mathcal{Z}) \xrightarrow{\uparrow} (s, \mathcal{Z}')$ is pre-stable.*

Proof. 1. To prove the first part of this theorem we may assume that $\mathcal{Z} = \mathsf{ActionPred}_e\ (\mathcal{Z}, \mathcal{Z}')$, which means that $\forall \nu \in \mathcal{Z}$, $\exists \nu'\ \mathcal{Z}'$, $\exists Y \subseteq X$, $\nu[Y \rightarrow 0] = \nu'$. Then, from $\forall\ (s, \mathcal{Z}) \xrightarrow{e} (s', \mathcal{Z}') \in \rightarrow_{\mathsf{ZG}}$ such that $\mathcal{Z} \subseteq \mathsf{ActionPred}_e(\phi, I(s'))$, it follows that $\mathcal{Z} \subseteq \phi$ and this is equivalent to $\forall \nu \in \mathcal{Z}$, $\exists \nu'\ \mathcal{Z}'$, $\exists Y \subseteq X$, $(\nu \in \phi \wedge \nu[Y \rightarrow 0] = \nu')$, which is according to $\mathcal{A}$ equivalent to $\forall (s, \nu, t) \in (s, \mathcal{Z})$, $\exists (s', \nu', t) \in (s', \mathcal{Z}')$, $(s, \nu, t) \xrightarrow{e}_{icta} (s', \nu', t)$, which means that the discrete transition is pre-stable.

2. To prove the second part of this theorem we consider a delay transition $(s, \mathcal{Z}) \xrightarrow{\uparrow} (s, \mathcal{Z}')$ of the zone graphs. From the definition of zone graph $\exists (s, \nu, t) \in (s, \mathcal{Z})$ which has a outcoming delay transition. Also, from the definition of zone graphs it follows that $(\mathcal{Z} \cup \mathcal{Z}') \subseteq I(s)$. We may assume that $\mathcal{Z} = \mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$, which means that $\forall \nu \in \mathcal{Z}$, $\exists \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0}$, $\exists\ \tau \in Rates$, $\exists\ t, t'' \geq 0$ and $t \leq t''$, $\forall t', t \leq t' \leq t''$, $(\nu +_{\pi}\ \boldsymbol{d}) \in \mathcal{Z}'$, $\boldsymbol{d} = \tau(t'') - \tau(t)$ and $\forall\ \boldsymbol{d}'$, $0 \leq \boldsymbol{d}' \leq \boldsymbol{d}$ then $(\nu +_{\pi}\ \boldsymbol{d}') \in (\mathcal{Z} \cup \mathcal{Z}')$, $\boldsymbol{d}' = \tau(t') - \tau(t)$. Give that $(\mathcal{Z} \cup \mathcal{Z}') \subseteq I(s)$, this is equivalent to $\forall (s, \nu, t) \in (s, \mathcal{Z})$, $\exists \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0}$, $\exists\ \tau \in Rates$, $\exists\ t, t'' \geq 0$ and $t \leq t''$, $\forall t', t \leq t' \leq t''$, $(s, \nu +_{\pi}\ \boldsymbol{d},\ t) \in (s, \mathcal{Z}')$, $\boldsymbol{d} = \tau(t'') - \tau(t)$ and $\forall\ \boldsymbol{d}'$, $0 \leq \boldsymbol{d}' \leq \boldsymbol{d}$ then $(s, \nu +_{\pi}\ \boldsymbol{d}',\ t) \in ((s, \mathcal{Z}) \cup (s, \mathcal{Z}'))$, $\boldsymbol{d}' = \tau(t') - \tau(t)$ and $(s, \nu +_{\pi}\ \boldsymbol{d}',\ t) \in I(s)$. According to icTA this equivalent to $\forall (s, \nu, t) \in (s, \mathcal{Z})$, $\exists \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0}$, $\exists\ \tau \in Rates$, $\exists\ t, t'' \geq 0$ and $t \leq t''$, $\forall t', t \leq t' \leq t''$, $(s, \nu, t) \xrightarrow{\boldsymbol{d}}_{\mathsf{icTA}} (s, \nu', t')$, $\boldsymbol{d} = \tau(t'') - \tau(t)$ and $\forall\ \boldsymbol{d}'$, $0 \leq \boldsymbol{d}' \leq \boldsymbol{d}$ then $\forall\ (s, \nu'', t'') \in Q_{\mathsf{icTA}}$, $(s, \nu, t) \xrightarrow{\boldsymbol{d}'}_{\mathsf{icTA}} (s, \nu'', t'')$, then $(s, \nu'', t') \in ((s, \mathcal{Z})$

$\cup\ (s, \mathcal{Z}'))$, $\boldsymbol{d'} = \tau(t') - \tau(t)$, which expresses that the delay transition is pre-stable.

□

Definition 17. *Let Π be a partition. Let $q = (s, \mathcal{Z})$ be a zone, then the signature* $\mathsf{ActionSigPred}(q, \Pi)$ *with regard to the partition Π is defined as:*

$\mathsf{ActionSigPred}(q, \Pi) = \{(Action(e), \mathcal{Z}) \mid \forall e, \exists\ \nu' \in \mathcal{Z}', \exists t \in \mathbb{R}_{\geq 0}\ (s, \nu, t) \xrightarrow{Action(e)}_{\Pi} (s', \nu', t')$ *and* $\mathcal{Z}, \mathcal{Z}' \in \Pi\}$

$\mathsf{ActionSigPred}(q, \Pi)$ operator is used in backward analysis of the set of zones. Our algorithm 2 consists of three steps: computing the timed predecessors (TimeSplit operator, see Definition 11 below), and computing the discrete signature predecessors (DiscreteSigSplit operator, see Definition 12 below) and computing the set of zones until a stable zones are reached, which is the multi-timed bisimulation. Stable zone are a multi-timed bisimulation relation if every two states of every zone in the set have the same signature with respect to every computed refinement. A detailed explication about building a stable zones follows:

- Initial set of zones: Let Q be a set of zones in $\mathsf{ZG}(\mathcal{C})$, where Q=$\{((s_{\mathcal{A}}, s_{\mathcal{B}}), \mathcal{Z}) \in Q \mid \mathcal{Z}$ *is a convex zone*$\}$. The initial partition is $\Pi_0 = Q$.

- Refinement: A existing set of zones are iteratively refined until all zones becomes stable simultaneously with respect to all their timed predecessors, discrete predecessors. For simplicity, we will write $(s, \mathcal{Z})$ to denote the pairs $((s_{\mathcal{A}}, s_{\mathcal{B}}), \mathcal{Z})$.

Definition 18. *Let Π be a set of zones in Q and $q = (s, \mathcal{Z})$, $q' = (s', \mathcal{Z}')$ two zones in Π. Then for the delay transitions, the splitter functions is defined as follows:*

$\mathsf{TimeSplit}(\mathcal{Z}, \Pi) = \{\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}') \mid \mathcal{Z}' \in \Pi, q \xrightarrow{\uparrow}_{\Pi} q'\}$

Definition 19. *Let Π be a set of zones in Q and $q = (s, \mathcal{Z}) \in \Pi$. Then the refinement of a zone q is defined as follows:*

$\mathsf{DiscreteSigSplit}(\mathcal{Z}, \Pi) = \{\nu' \in \mathcal{Z} \mid \forall \nu' \in \mathcal{Z}' \in \Pi, \mathsf{ActionSigPred}(q, \Pi) = \mathsf{ActionSigPred}(q', \Pi)$ *and* $\nu \in \mathcal{Z}\}$

Algorithm 3: The partition refinement algorithm for a reachable ZG

Input : A $\mathsf{ZG}(\mathcal{C}) = (Q = Q_{\mathcal{A}} \times Q_{\mathcal{B}}, q_0 = (q^0_{\mathcal{A}}, q^0_{\mathcal{B}}), \Sigma = \Sigma_{\mathcal{A}} \cup \Sigma_{\mathcal{B}}, \rightarrow_{\mathsf{ZG}})$, Π.
Output: A coarsest partition Π.

```
// q ∈ Q is a zone of ZG(C), Π is a set of zones, Z, Z' are clock zones.
// Q is a set of pairs S × Z.
Function PartitionZoneGraph(ZG(C), Π)
    // Phase I - Get the input partition Π
    Π' ← Π ;
    Repeat
        // Phase II - Refine Π' by delay transitions:
        for each zone (or block) Z ∈ Π' do
            Π' ← TimeSplit(Z, Π') ;
        end
        // Phase III - Refine Π' by discrete transitions:
        for each zone (or block) Z ∈ Π' do
            Π' ← DiscreteSigSplit(Z, Π') ;
        end
    Until Π' does not change;
    Return Π' ;
end
```

Lemma 4. *Let $(s, \mathcal{Z})$ be a class of Π and let e be an edge of the $\mathsf{ZG}(\mathcal{C})$, then each of $\mathsf{TimeSplit}(\mathcal{Z}, \Pi)$ and $\mathsf{DiscreteSigSplit}(\mathcal{Z}, \Pi)$ forms a partition of $\mathcal{Z}$ in zones.*

Our proof follows the same lines as the proof of [26].

Proof. Consider $\Pi_1 = \mathsf{TimeSplit}(\mathcal{Z}, \Pi)$ first. By Lemma 8, all members of Π are zones. It remains to show that they are disjoint and that their union yields $\mathcal{Z}$. Let $\mathcal{Z}_i \in \Pi_1$, $\mathcal{Z}_i = \mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}'_i)$, where $\mathcal{Z}_i \in \Pi$. Since Π is a partition, $\mathcal{Z}$, $\mathcal{Z}'_1$ and $\mathcal{Z}'_2$. Assumes $\nu \in \mathcal{Z}_1 \cap \mathcal{Z}_2$. For $i = 1, 2$, $\exists\ \boldsymbol{d}_i \in \mathbb{R}^{Proc}_{>0}$ such that $\nu +_\pi \boldsymbol{d}_i \in \mathcal{Z}_i$ and $\forall \nu \in \mathcal{Z}$, $\exists \boldsymbol{d}_i \in \mathbb{R}^{Proc}_{>0}$, $\exists\ \tau \in Rates$, $\exists\ t, t'' \geq 0$ and $t \leq t''$, $\forall t', t \leq t' \leq t''$, $(\nu +_\pi \boldsymbol{d}_i) \in \mathcal{Z}'_i$, $\boldsymbol{d}_i = \tau(t'') - \tau(t)$ and $\forall\ \boldsymbol{d}'_i$, $0 \leq \boldsymbol{d}'_i \leq \boldsymbol{d}_i$ then $(\nu + \boldsymbol{d}'_i) \in (\mathcal{Z} \cup \mathcal{Z}'_i)$, $\boldsymbol{d}'_i = \tau(t') - \tau(t)$. Observe that $\boldsymbol{d}_1 \neq \boldsymbol{d}_2$, since $\mathcal{Z}'_1$ and $\mathcal{Z}'_2$ are disjoint. Without loss of generality, assume $\boldsymbol{d}_1 < \boldsymbol{d}_2$. We have that $\nu +_\pi \boldsymbol{d}_1 \in \mathcal{Z}'_1$ and $\nu +_\pi \boldsymbol{d}_1 \in \mathcal{Z} \cup \mathcal{Z}'_2$, that is, either $\nu +_\pi \boldsymbol{d}_1 \in \mathcal{Z}'_1 \cap \mathcal{Z}$ or $\nu +_\pi \boldsymbol{d}_1 \in \mathcal{Z}'_1 \cap \mathcal{Z}'_2$, which contradicts the fact that $\mathcal{Z}$, $\mathcal{Z}'_1$ and $\mathcal{Z}'_2$ are all disjoint. This proves that $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are disjoint. Now,let $\nu \in \mathcal{Z}$. We can find $\mathbb{R}^{Proc}_{>0}$ and $\mathcal{Z}' \in \Pi$ such that $\nu +_\pi \boldsymbol{d} \in \mathcal{Z}'$ and $\forall \nu \in \mathcal{Z}$, $\exists \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0}$, $\exists\ \tau \in Rates$, $\exists\ t, t'' \geq 0$ and $t \leq t''$, $\forall t', t \leq t' \leq t''$, $(\nu +_\pi \boldsymbol{d}) \in \mathcal{Z}'$, $\boldsymbol{d} = \tau(t'') - \tau(t)$ and $\forall\ \boldsymbol{d}'$, $0 \leq \boldsymbol{d}' \leq \boldsymbol{d}$ then $(\nu +_\pi \boldsymbol{d}') \in (\mathcal{Z} \cup \mathcal{Z}')$, $\boldsymbol{d}' = \tau(t') - \tau(t)$. By definition, $\nu \in \mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$. Now, consider $\Pi_2 = \mathsf{DiscreteSplit}(\mathcal{Z}, e, \Pi)$. By Lemma 5 , all members of Π_2 are zones. By the distributivity of pred over union ($\mathsf{pred}(\mathcal{Z}_1 \cup \mathcal{Z}_2, e) = \mathsf{pred}(\mathcal{Z}_1, e) \cup \mathsf{pred}(\mathcal{Z}_2, e)$) members of Π_2 cover $\mathcal{Z}$. It remains to show that they are disjoint. Let $\mathcal{Z}_i \in \Pi_2$, $\mathcal{Z}_i = \mathcal{Z} \cap \mathsf{Actionpred}(\mathcal{Z}'_i, e)$, where $\mathcal{Z}'_i \in \Pi$, for $i = 1, 2$. Since Π is a partition, $\mathcal{Z}'_1$ and $\mathcal{Z}'_2$ are disjoint. Assume $\nu \in \mathcal{Z}_1 \cap \mathcal{Z}_2$. Recall that the successor of ν, say ν', is unique. Since $\nu \in \mathsf{pred}(\mathcal{Z}'_1, e) \cap \mathsf{pred}(\mathcal{Z}'_2, e)$, it must be that $\nu' \in \mathcal{Z}'_1 \cap \mathcal{Z}'_2$, which contradicts $\mathcal{Z}'_1 \cap \mathcal{Z}'_2 = \emptyset$.

Correctness and Termination Algorithm 3: We can analyze the correctness and termination of the algorithm 2 by evaluating the behavior of the splitter functions, we may obtain new blocks (or zones), where for any two zones in different blocks, they are not bisimilar and these blocks are disjoint. The correctness

of the algorithm 2 follows from the standard partition refinement algorithm [26]. The definition TimeSplit($\mathcal{Z}, \Pi$) above to generate a finer partition, which deals with delay transitions. The definition of DiscreteSplit($\mathcal{Z}, e, \Pi$), generate also a finer partition and distinguishes the states with discrete transitions. Termination is ensured by Lemma 9. However, in the worst case, the algorithm will generate the partition induced by the region equivalence. Time-abstract bisimulation and timed bisimulation have been shown decidable for TA in EXPTIME [21].

Multi-timed Bisimulation Algorithm: Inspired by [20][26], algorithm 3 starts with zone graph and an initial partition and then refines it through the function PartitionZoneGraph. Algorithm 3 checks whether two initial states zone graph $\mathcal{A}$ and $\mathcal{B}$ are related according to some multi-timed bisimulation relation $\mathcal{R}$.

Algorithm 3 describes the main steps of the decision procedure DecideMulti-TimedBisim. It is based on three functions: (1) function BuildSymbZoneGraph returns a zone graph. (2) function StartingInitialPartition returns an initial partition Π_0. It build an initial given a zone graph. We omit the pseudo code from this paper. (3) function PartitionZoneGraph returns stable partition Π. Given a partition Π, distinguishes the states in Π that are incompatible, and divides Π into subclasses, thus refining the current partition. This means that the algorithm 3 computes the states $((s_{\mathcal{A}}, s_{\mathcal{B}}), \mathcal{Z})$ from Π that are bisimilar up until the desired initial state $((s^0_{\mathcal{A}}, s^0_{\mathcal{B}}), \mathcal{Z}_0)$. Note that $\nu^0_{\mathcal{A}}$ and $\nu^0_{\mathcal{B}} \in \mathcal{Z}_0$. The projection of a zone $\mathcal{Z}^0_{\mathcal{A}}$ on a clock subset $Y_{\mathcal{A}} \subseteq X$ is $\mathcal{Z}^0_{\mathcal{A}} = \{\nu\rfloor_{Y_{\mathcal{A}}} \mid \nu \in \mathcal{Z}_0\}$ and $Y_{\mathcal{B}} \subseteq X$ is $\mathcal{Z}^0_{\mathcal{B}} = \{\nu\rfloor_{Y_{\mathcal{B}}} \mid \nu \in \mathcal{Z}_0\}$. A common data structure used for the representation of clock zones is a Difference bound matrices (DBMs)[9]. A canonical form of the DBMs simplify some operations over zones like the test for inclusion (inclusion($\mathcal{Z}^0_{\mathcal{A}}, \mathcal{Z}^0_{\mathcal{B}}$) between zones. For any two DBMs $\mathcal{Z}^0_{\mathcal{A}}$, $\mathcal{Z}^0_{\mathcal{B}}$ in canonical form, the inclusion operation denoted as $\mathcal{Z}^0_{\mathcal{A}} \subseteq \mathcal{Z}^0_{\mathcal{B}}$ returns true or false.

Example 5. An example of the zone graph, partition and multi-timed bisimulation computed by our algorithms can be found in Figure 10. The Figure 10 (a) shows two icTA $\mathcal{A}$ and $\mathcal{B}$ with the finite input alphabet $\Sigma = \{a, b\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau_p > \tau_q$. The Figure 10 (b) shows the zone graph computed by algorithm 1. The Figure 10 (c) shows the multi-timed bisimualtion for $\mathcal{A}$ and $\mathcal{B}$.

Complexity Algorithm 3: Now, we analyze the problem of deciding whether two icTA are multi-tumed bisimilar is EXPTIME-complete. Our algorithm 3 use the idea of composition (or product) of two reachable zone graph ZG($\mathcal{C}$) = ZG($\mathcal{A}$) $\parallel$ ZG($\mathcal{B}$) to decide multi-timed bisimulation from [1]. Our approach is based on a simple reduction from linearly bounded alternating Turing machines. Our reduction is standard [1][21], and it can be applied to both on TA and icTA (under our multi-timed semantics). Timed bisimulation have been shown decidable for TA in EXPTIME [1] [17] [21].

An Alternating Turing Machine (ATM) : An ATM [15] is a tuple $\mathcal{M} = \langle Q, q_0, \Gamma, \rightarrow, Q_F\rangle$, where $Q = Q_\vee \cup Q_\wedge$ is a set of states partitioned into disjunctive states $Q_\vee$ and conjunctive states $Q_\wedge$, $q_0 \in Q$ is the initial state, $\Gamma = \{a, b\}$

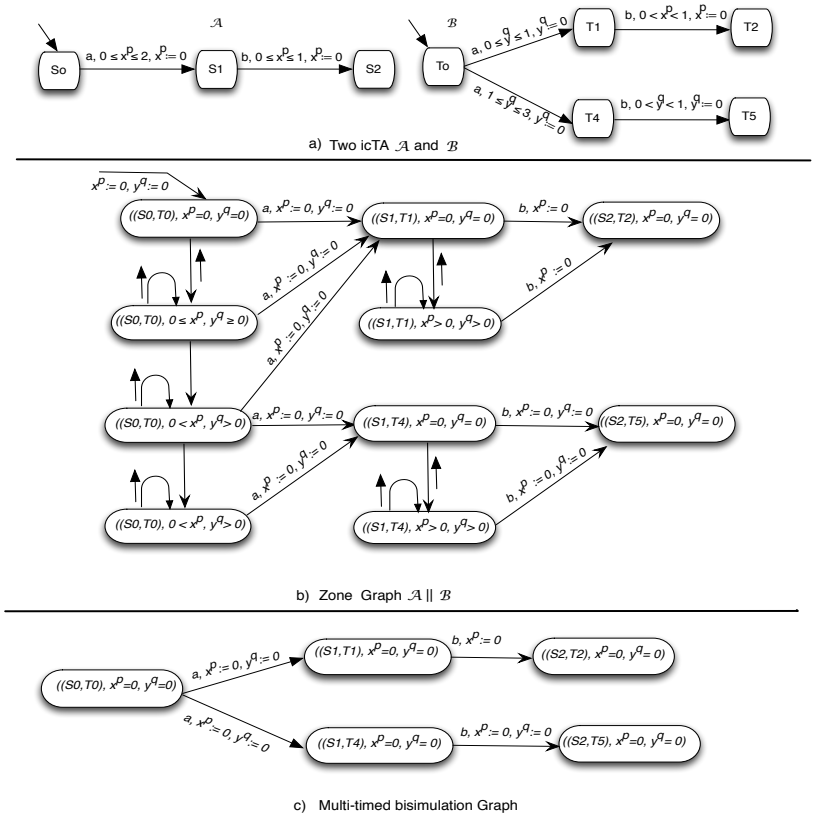

Fig. 10. (a) Composition of icTAs; (b) Zone graph; (c) bisimulation

is the tape alphabet and containing a special blank symbol, denoted by $\$ \in \Gamma$, $Q_F \subseteq Q$ is the set of accepting states and $\rightarrow \subseteq Q \times \Gamma \times Q \times \Gamma \times \{-1, 1\}$ is the transition relation. In each non terminal step (i.e., the current state $q \in Q$), $\mathcal{M}$ overwrites the tape cell being scanned, and the tape head moves one position to the left 1 or right 1. The idea of the reduction is based on the fact that instead of considering a computation that just stops in an accepting state we will encode existence of a computation that after reaching an accepting state the machine restarts in the initial configuration.

A configuration is a triple $\alpha = (q, i, w)$, where $q \in Q$ is the current state, $w \in \Gamma^*$ is word describing the tape content and $0 \leq i \ |w|$ is the position of the head on the tape. The symbol written in the ith cell of the tape is denoted by $w(i)$. A configuration (q, i, w) is final iff $q = q_F \in Q$. An ATM moves like a usual nondeterministic Turing machine: for example, if $\alpha = (q, i, w)$, $w(i) = a$ and $(q, a, q', b, \Delta) \in \rightarrow$, then $\mathcal{M}$ may move from α to $\alpha' = (q', i', w')$, written $\alpha \rightarrow \alpha'$ where w' is w updated by writing a b in position i, and i' is $i + 1$ if $\Delta = 1$ or $i - 1$ if $\Delta = -1$. We say that α' is a successor of α. We also assume that $\mathcal{M}$ has only a finite number of successor (q', i', w') for which $q = q_F$, and that $i = 1$ and $w = a^n$.

A run of $\mathcal{M}$ from some configuration α_0 is a tree, the root of which corresponds to α_0, and where every node corresponding to α has a child node for each successor α' of α. For $k \in \mathbb{N}$, a run rooted at some disjunctive configuration α is accepting in k steps if and only if its state is q_F or $k \geq 1$ and at least one of its children is accepting in $k1$ steps. A run rooted at some conjunctive configuration α is accepting in k steps if and only if $k \geq 1$ and all of its children is accepting in $k1$ steps (and there is at least one child). A word v is accepted by $\mathcal{M}$ if and only if there exists some k such that the run from $(q_0, 1, v)$ is accepting in k steps. We say that $\mathcal{M}$ is linearly bounded (LB-ATM) on v if all configurations (q, i, w) in the run of $\mathcal{A}$ have $|w| \leq |v|$. The problem of acceptance of a LB-ATM, which we denote by LB-ATM accept, is written as: Input An ATM $\mathcal{M}$ and a word $v \in \Gamma^*$ such that $\mathcal{M}$ is linearly bounded on v. Output Yes, iff $\mathcal{M}$ accepts v. No otherwise.

A classical result says that the problem LB-ATM accept is EXPTIME-complete [15]. In the following, we assume, that along a single branch of a run of an LB-ATM, no configuration is repeated; thus every branch is finite. This assumption does not change the complexity issues: one can easily reduce an instance $(\mathcal{M}, v)$ of LB-ATM accept to some instance $(\mathcal{M}', v')$ where $\mathcal{M}'$ avoids repetitions by inserting on the tape a counter (encoded in binary) whose value is bounded by $2^{|v|} \times |Q| \times |v|$ (the maximum number of distinct configurations along the run). Then $\mathcal{M}'$ simulates the moves of $\mathcal{M}$ and increases the counter by 1 for every simulated move of $\mathcal{M}$.

We reduce the acceptance problem for LB-ATM to the problem of deciding if two icTA are multi-timed bisimilarities.

Theorem 3. *Deciding multi-timed bisimulation between two icTA is EXPTIME-complete.*

Proof. Let $\mathcal{M} = \langle Q, q_0, \Gamma, \rightarrow, Q_F \rangle$, where $Q = Q_\vee \cup Q_\wedge$ be an LB-ATM and v be a word of length n. We define an icTA $\mathcal{C} = (\Sigma, X, S, s_0, \rightarrow_{icta}, I, F, \pi)$ as the parallel composition of two icTA $\mathcal{A}$ and $\mathcal{B}$. The behavior of $\mathcal{M}$ over v can be encoded by a icTA $C_{\mathcal{M},v}$, which models the run of $\mathcal{M}$ over v. Then we let $S = (Q \times \{1, \ldots, n\}) \cup \{init, end\}$, $\Sigma = \{a, b\}$, and $X = \{x_1, \ldots, x_n, y_1, \ldots, y_n, t\}$. The contents of the tape of $\mathcal{M}$ are encoded by the relative values of the clocks $x_1, \ldots, x_n, y_1, \ldots, y_n$: cell i contains a if $x_i = y_i$, and b if $x_i < y_i$. Clock t is used to ensure the elapse of time of length 1 between transitions. The initial state is $q_0^{\mathcal{M}_c} = (q_0, 1)$. The transition relation $\rightarrow_{icta}$ of $C_{\mathcal{M},v}$ is defined as follows: (i) For each $q \in Q_\vee$, each $1 \leq i \leq n$, and each $e = (q, o, q', o', \delta) \in \rightarrow_{\mathcal{M}}$, a transition $(q, i) \xrightarrow{e,i} (q', i + \delta)$ is included in $\rightarrow_{C_{\mathcal{M},v}}$ if $e = (q, o, q', o', \delta)$ and $i + \delta$ denote $i + 1$ (or $i - 1$) if $\delta = 1$ and $i < n$ or $\delta = -1$ and $i > 1$. The transition $(q, i) \xrightarrow{e,i} (q', i + \delta)$ is replaced by a transition $(q, i) \xrightarrow{t=1 \wedge \phi, e, Y} (q', i + \delta)$, where the guards ϕ of the discrete transition from a given location (q, i) can test whether the current tape symbol is a or b by checking whether $x_i = y_i$ or x_i ¡ y_i, respectively. Furthermore, the writing of a symbol in a tape cell can be replicated by clock resets: for example, to represent the writing of a in cell i, we reset clocks x_i and y_i to 0 (so that $x_i = y_i$), whereas to write b we reset only x_i (so that x_i ¡

y_i). The target locations of the discrete transitions are derived from the target states of the transition of $\mathcal{M}$ involved in the definition of the discrete transition and by the associated movement of the tape head. The initialization of the tape with the input word v can be encoded by $init \xrightarrow{t=1,s_0,Y_0} (q_0, 1)$ where $Y_0 = \{t\} \cup \{x_i \mid v(i) = b\}$.

The size of $\mathcal{C}_{\mathcal{M},v}$ is $O(n \times |Q| \times | \rightarrow |^2)$ and the reduction can be done in logarithmic space. Now we show that $\mathcal{M}$ accepts v if and only if icYA returns Yes for $\mathcal{C}_{\mathcal{M},v}$ with the initial state $(q_0, 1)$, and the set containing the single final state $(q_F, 1)$.

□

5 Related Work

We have reviewed several models considered in the TA literature to study timed bisimulation. Decidability for timed bisimulation between TA was given in [14] using a region construction. An implementation based on region construction was proposed [14]. In [27] a zone-based algorithm for checking (weak) timed bisimulation was proposed though never implemented in any tool. Time-abstracting bisimulation between TA was given in [26]. A tool based on time-abstracting bisimulation was proposed [26]. In [13] is studied timed simulation through a formalism for defining simulation-checking games. A tool based on simulation-checking games was proposed [8]. The notion of clock drifts were considered in [19] and [24] in the context of DTA, but the notion of timed bisimulation is not considered. In [2], the notion of timed bisimulation was proposed using region construction though never implemented in any tool.

6 Conclusions

In this paper we have presented four main contributions: (1) An extension of TLTS and icTA based on independent clocks. (2) A new approach to timed bisimulation, called multi-timed bisimulation which applies the idea of independent clocks. (3) An EXPTIME algorithm for deciding multi-timed bisimulation have been presented, from the theory of [26] and [8]. Finally, as future work, we envisage to implement the algorithms 1, 2, and 3 for calculating their runtimes and to compare the algorithms proposed in the current paper with other algorithms proposed in [26][8].

References

1. L. Aceto and F. Laroussinie. Is your model checker on time? on the complexity of model checking for timed modal logics. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *MFCS*, volume 1672 of *LNCS*, pages 125–136. Springer, 1999.
2. S. Akshay, B. Bollig, P. Gastin, M. Mukund, and K. N. Kumar. Distributed timed automata with independently evolving clocks. In *CONCUR*, LNCS. Springer, 2008.

3. R. Alur. Timed automata. In N. Halbwachs and D. A. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99)*, volume 1633 of *Lecture Notes in Computer Science*, pages 8–22. Springer-Verlag, July 1999.
4. R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *13th IEEE Real-Time Systems Symposium*, Phoenix, Az, dec 1992.
5. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994.
6. S. Balaguer and T. Chatain. Avoiding shared clocks in networks of timed automata. *Logical Methods in Computer Science*, 2013.
7. G. Behrmann, P. Bouyer, E. Fleury, and K. G. Larsen. Static guard analysis in timed automata verification. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 254–277. Springer, 2003.
8. G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *STTT*, 2006.
9. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lecture Notes on Concurrency and Petri Nets*, 2004.
10. S. Blom and S. Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. *Electr. Notes Theor. Comput. Sci.*, 2002.
11. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.
12. M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988.
13. P. Bulychev, T. Chatain, A. David, and K. G. Larsen. Efficient on-the-fly algorithm for checking alternating timed simulation. In *FORMATS*, LNCS. Springer, 2009.
14. K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *CAV*, London, UK, 1993.
15. A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
16. M. De Biasi, C. Snickars, K. Landernäs, and A. Isaksson. Simulation of process control with wirelesshart networks subject to clock drift. In *COMPSAC*, 2008.
17. M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS '95, pages 453–, Washington, DC, USA, 1995. IEEE Computer Society.
18. T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In *REX Workshop*, LNCS. Springer, 1991.
19. P. Krishnan. Distributed timed automata. *In Workshop on Distr. Systems*, 1999.
20. F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic - and back. In *MFCS*, LNCS. Springer, 1995.
21. F. Laroussinie and Ph. Schnoebelen. The state-explosion problem from trace to bisimulation equivalence. In J. Tiuryn, editor, *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 192–207, Berlin, Germany, Mar. 2000. Springer.
22. R. Milner. *Communication and concurrency.* Prentice Hall, 1989.
23. A. Monot, N. Navet, and B. Bavoux. Impact of clock drifts on CAN frame response time distributions. In *ETFA*, Toulouse, France, 2011.
24. J. J. Ortiz, A. Legay, and P.-Y. Schobbens. Distributed event clock automata. In *CIAA*, LNCS. Springer, 2011.
25. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, pages 973–989, 1987.

26. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 2001.
27. C. Weise and D. Lenzkes. Efficient scaling-invariant checking of timed bisimulation. In *STACS*, LNCS. Springer, 1997.