Fundamenta Informaticae XX (2013) 1–31
IOS Press

Distributed Timed Automata with Independently Evolving Clocks*

S. Akshay[1,2,3], Benedikt Bollig[1], Paul Gastin[1], Madhavan Mukund[2], and K. Narayan Kumar[2]

[1]*LSV, ENS Cachan, CNRS, INRIA, France*
[2]*Chennai Mathematical Institute, Chennai, India*
[3]*Dept. of CSE, IIT Bombay, Mumbai, India*

Abstract. We propose a model of distributed timed systems where each component is a timed automaton with a set of local clocks that evolve at a rate independent of the clocks of the other components. A clock can be read by any component in the system, but it can only be reset by the automaton it belongs to.

There are two natural semantics for such systems. The *universal* semantics captures behaviors that hold under any choice of clock rates for the individual components. This is a natural choice when checking that a system always satisfies a positive specification. To check if a system avoids a negative specification, it is better to use the *existential* semantics—the set of behaviors that the system can possibly exhibit under some choice of clock rates.

We show that the existential semantics always describes a regular set of behaviors. However, in the case of universal semantics, checking emptiness or universality turns out to be undecidable. As an alternative to the universal semantics, we propose a *reactive* semantics that allows us to check positive specifications and yet describes a regular set of behaviors.

Keywords: timed automata, distributed systems

1. Introduction

In today's world, it is becoming increasingly important to look at networks of timed systems, which allow real-time systems to operate in a distributed manner. Many real-life systems, such as mobile phones, computer servers, and railway crossings, depend crucially on timing while usually consisting of

*Partially supported by ARCUS, DOTS (ANR-06-SETIN-003), and P2R MODISTE-COVER/RNP Timed-DISCOVERI.

many interacting systems. In general, there is no reason to assume that different timed systems in the networks refer to the same time or evolve at the same rate.

Timed automata [2] are a well-studied formalism to describe systems that require timing. However, networks of timed automata, under the assumption of knowledge of a global time, as done in [6, 7, 13], do not really reflect the distributed model.

In this paper, we provide a framework to look at distributed systems with independently evolving local clocks. Each constituent system is modeled by a timed automaton. All clocks belonging to this timed automaton evolve at the same rate. However clocks belonging to different processes are allowed to evolve at rates that are independent of each other. We allow clocks belonging to one process to be read/checked by another but we require that a clock can only be reset by the automaton it belongs to. This relation of "belonging to" is formalized by defining an *ownership map* assigning clocks to processes.

Since we have unrelated time values on different processes, we are interested in the underlying untimed behaviors of these distributed timed automata rather than their timed behaviors. Thus, the clocks (and time itself) are implementation or synchronization tools rather than being a part of the observation. This is a crucial point where our work departs from other related works (see below).

When we assume that clocks behave asynchronously and are subject to external, uncontrollable effects, the precise system behavior is unpredictable and not amenable to verification. It is therefore natural to look at different approximations depending on the specification that we want our system to satisfy. When we want to guarantee that our system exhibits a positive specification, we look at the *universal* semantics. This semantics describes the behaviors exhibited by the system no matter how time evolves in the constituent processes. Thus, the universal semantics is an under-approximation of the actual system behavior. However, if we want to check that our system avoids a negative specification, then we prefer to look at the *existential* semantics. This is the set of behaviors that the system might exhibit under some (bad) choice of local time rates in the constituent processes. This semantics is an over-approximation of the system behavior. We perform a region construction on our distributed timed automata to show that the existential semantics always gives a regular set of untimed behaviors. Thus the model checking problem of distributed timed automata against regular negative specifications is decidable. On the other hand, we show that checking emptiness or universality for the universal semantics is undecidable. This result is further strengthened to a bounded case, where we have restrictions on the relative time rates. To overcome the negative result and be able to check positive specifications, we introduce a third semantics: The *reactive semantics*, which will rely on a game view of our distributed timed systems, is an under-approximation of the universal semantics and, therefore, of the actual system behavior. We show that the reactive semantics is always regular and computable so that it allows us to check positive specifications effectively.

In the above, we have considered systems with a distributed state space consisting of several timed automata. But if we consider the ownership map assigning clocks to processes, and take a product of the states of these timed automata, we obtain a single timed automaton with a global state space having independently evolving clocks. This model with a global state space is more general than the distributed one, in the sense that any distributed system can be described over a global state space by taking an asynchronous product of the local state spaces, but the reverse is not true. In this paper, we explicitly consider both these types of independently evolving clock models, having global and distributed state spaces. In fact, for many of our results, it will turn out to be simpler to work with global state spaces. Then, the positive results can be immediately inferred for the distributed state space, while we need to do some more work to prove the same for the negative results. Nevertheless, all the results in this paper

go through for distributed as well as global state spaces.

Related work In [9, 17, 18], classical timed automata are equipped with an additional parameter Δ, which allows a clock to diverge over a period t from its actual value by Δt. This model conforms, in a sense, to our existential semantics, where we restrict the set of clock rates to those corresponding to Δ (see Section 5). Syntactically, our model coincides with that from [10]: A clock can only be reset by the owner process, whereas it can be read by any process. However, existing works differ from ours since they consider timed words rather than untimed languages and we use clocks and time as synchronization tools only. This also explains why our automata differ from hybrid automata [12]. In the model of [4], clocks are not shared and clocks on different processes drift only as long as the processes do not communicate (via synchronizing actions). These assumptions make partial-order–reduction techniques applicable. Another fundamental difference between all these approaches and our work is that we do not restrict to the study of system configurations that can be reached under *some* local-time behavior, which is only a suitable abstraction if we consider safety specifications. To check positive specifications, we also provide semantics that can check if a system exhibits some behavior under *all* relative clock speeds.

Structure of the paper In Section 2, we introduce our distributed automaton model with independently evolving clocks, and define its existential and universal semantics. Section 3 extends the regions of a timed automaton to our distributed setting, allowing us to compute a finite automaton recognizing the existential semantics. Section 4 shows that checking emptiness and universality of the universal semantics is undecidable. This result is sharpened towards bounded clock drifts in Section 5. Section 6 deals with the reactive semantics, and Section 7 identifies some directions for future work.

A preliminary version of this paper appeared as [1].

2. Distributed timed automata

Preliminaries For a set Σ, we let Σ^* and Σ^ω denote the set of finite and, respectively, infinite words over Σ. The empty word is denoted by ε. We set $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The concatenation of words $u \in \Sigma^*$ and $v \in \Sigma^\infty$ is denoted by $u \cdot v$. An *alphabet* is a non-empty finite set. Given an alphabet Σ, we denote by Σ_ε the set $\Sigma \uplus \{\varepsilon\}$. The sets of non-negative/positive real numbers are denoted by $\mathbb{R}_{\geq 0}$/$\mathbb{R}_{>0}$, respectively. For $t \in \mathbb{R}_{\geq 0}$, $\lfloor t \rfloor$ and $\mathit{fract}(t)$ refer to the integral and, respectively, fractional part of t, hence $t = \lfloor t \rfloor + \mathit{fract}(t)$.

The set $\mathrm{Form}(\mathcal{Z})$ of *clock formulas* over a set of clocks $\mathcal{Z}$ is given by the grammar

$$\varphi ::= \mathrm{true} \mid x \bowtie c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

where x is a clock from $\mathcal{Z}$, $\bowtie \in \{<, \leq, >, \geq, =\}$, and c ranges over $\mathbb{N} = \{0, 1, 2, \ldots\}$. A *clock valuation* over $\mathcal{Z}$ is a mapping $\nu : \mathcal{Z} \to \mathbb{R}_{\geq 0}$. We say that ν satisfies $\varphi \in \mathrm{Form}(\mathcal{Z})$, written $\nu \models \varphi$, if φ evaluates to true using the values given by ν. For $R \subseteq \mathcal{Z}$, $\nu[R]$ denotes the clock valuation defined by $\nu[R](x) = 0$ if $x \in R$ and $\nu[R](x) = \nu(x)$, otherwise.

The model Let us recall the notion of timed automata [2]. These will constitute the building blocks of our distributed timed automata. A *timed automaton* is a tuple $\mathcal{A} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F)$ where S is a finite set of *states*, Σ is the alphabet of *actions*, $\mathcal{Z}$ is a finite set of *clocks*, $\delta \subseteq S \times \Sigma_\varepsilon \times \mathrm{Form}(\mathcal{Z}) \times 2^{\mathcal{Z}} \times S$ is the finite set of *transitions*, $I : S \to \mathrm{Form}(\mathcal{Z})$ associates with each state an *invariant*, $\iota \in S$ is the *initial*

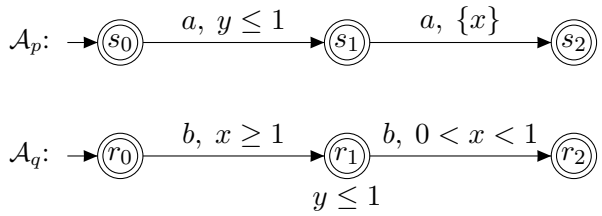


Figure 1. A distributed timed automaton over $\{p, q\}$

state, and $F \subseteq S$ is the set of *final states*. We let $Reset(\mathcal{A}) = \{x \in \mathcal{Z} \mid \text{there is } (s, a, \varphi, R, s') \in \delta \text{ such that } x \in R\}$ be the set of clocks that might be reset in $\mathcal{A}$. Without loss of generality, we will assume in this paper that $I(\iota)$ is satisfied by the clock valuation over $\mathcal{Z}$ that maps each clock to 0.

We will now extend the above definition to a distributed setting. First, we fix a non-empty finite set $Proc$ of processes. For a tuple t that is indexed by $Proc$, t_p refers to the projection of t onto $p \in Proc$.

Definition 2.1. A *distributed timed automaton (DTA)* over the set of processes $Proc$ is a structure $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ where the $\mathcal{A}_p = (S_p, \Sigma_p, \mathcal{Z}_p, \delta_p, I_p, \iota_p, F_p)$ are timed automata such that the alphabets Σ_p are pairwise disjoint, and π is a (total) mapping from $\bigcup_{p \in Proc} \mathcal{Z}_p$ to $Proc$ such that, for each $p \in Proc$, we have $Reset(\mathcal{A}_p) \subseteq \pi^{-1}(p) \subseteq \mathcal{Z}_p$.

Note that $\mathcal{Z}_p$ refers to the set of clocks that might occur in the timed automaton $\mathcal{A}_p$, either as clock guard or reset. The same clock may occur in both $\mathcal{Z}_p$ and $\mathcal{Z}_q$, since it may be read as a guard in both processes. However, any clock evolves according to the time evolution of some particular process. This clock is then said to *belong* to that process, and the *owner* map, π, formalizes this in the above definition. This will become more clear when we describe the formal semantics later in this section. Finally, we assume that a clock can only be reset by the process it belongs to.

Example 2.1. Suppose $Proc = \{p, q\}$. Consider the DTA $\mathcal{D}$ as given by Figure 1. It consists of two timed automata, $\mathcal{A}_p$ and $\mathcal{A}_q$ with $\mathcal{Z}_p = \mathcal{Z}_q = \{x, y\}$. In both automata, we suppose all states to be final. Moreover, the owner mapping π maps clock x to p and clock y to q. Note that $Reset(\mathcal{A}_p) = \{x\}$ and $Reset(\mathcal{A}_q) = \emptyset$. Before we define the semantics of $\mathcal{D}$ formally and in a slightly more general setting, let us give some intuitions on the behavior of $\mathcal{D}$. If both clocks are completely synchronized, i.e., they follow the same local clock rate, then our model corresponds to a standard network of timed automata [6]. For example, we might execute a within one time unit, and at time 1, execute b, ending up in the global state (s_1, r_1) and clock valuation $\nu(x) = \nu(y) = 1$. If we now wanted to perform a further b, this should happen instantaneously. But this also requires a reset of x in the automaton $\mathcal{A}_p$ and, in particular, a time elapse greater than zero, violating the invariant at the local state r_1. Thus, the word $abab$ will not be in the semantics that we associate with $\mathcal{D}$ wrt. synchronized local-time evolution. Now suppose clock y runs slower than clock x. Then, having executed ab, we might safely execute a further a while resetting x and, then, let some time elapse without violating the invariant. Thus, $abab$ will be contained in the *existential* semantics, as there are local time evolutions that allow for the execution of this word. Observe that a and aa are the only sequences that can be executed no matter what the relative time speeds are: the guard $y \leq 1$ is always satisfied for a while. But we cannot guarantee that the guard

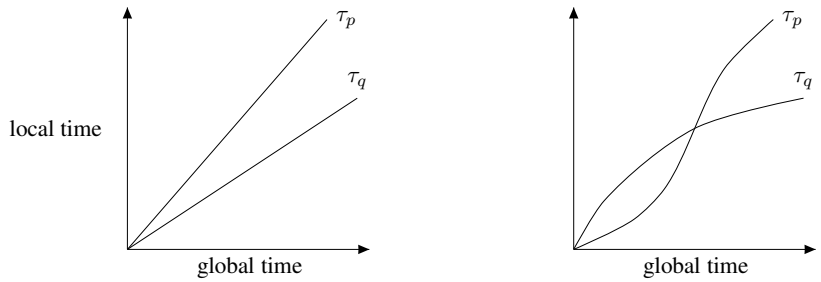


Figure 2. Examples of local time rate functions

$x \geq 1$ and the invariant $y \leq 1$ are satisfied at the same time, which prevents a word containing b from being in the *universal* semantics of $\mathcal{D}$.

The semantics The semantics of a DTA depends on the (possibly dynamically changing) time rates at the processes. To model this, we assume that these rates depend on some absolute time, i.e., they are given by a tuple $\tau = (\tau_p)_{p \in Proc}$ of functions $\tau_p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Thus, each local time function maps every point in global time to some local time instant. Then, we require (justifiably) that these functions are continuous, strictly increasing, and divergent. Further, they satisfy $\tau_p(0) = 0$ for all $p \in Proc$. The set of all these tuples τ is denoted by $Rates$. We might also consider τ as a mapping $\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}^{Proc}$ so that, for $t \in \mathbb{R}_{\geq 0}$, $\tau(t)$ denotes the tuple $(\tau_p(t))_{p \in Proc}$.

By superimposing the local time rate maps for each process on the same graph, we can represent τ pictorially, as in Figure 2. Thus, in the first picture, clocks on process p evolve steadily faster than clocks on process q. Whereas, in the second picture, the clocks on process q are initially faster than clocks on process p but start to lag behind them after some time.

Now, a distributed system can usually be described by an asynchronous product of automata. In the case of DTA, the semantics can be defined using such a product and a mapping that assigns any clock to its owner process. For this, we start by introducing the following more general model, with a unified state space, for which it will be easier to define the semantics.

Definition 2.2. A *timed automaton with independently evolving clocks (icTA)* over $Proc$ is a tuple $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ where $(S, \Sigma, \mathcal{Z}, \delta, I, \iota, F)$ is a timed automaton and $\pi : \mathcal{Z} \to Proc$ maps each clock to a process.

Below, we define the semantics of a DTA $\mathcal{D}$ in terms of an icTA $\mathcal{B}_\mathcal{D}$. Most of the following definitions and results are based on this more general notion of a timed system and therefore automatically carry over to the special case of DTAs.

Now, we would like to define a run of an icTA. Intuitively, this is done in the same spirit as a run of a timed automaton over a timed word except for one difference. The time evolution, though according to absolute time, is perceived by each process as its *local time* evolution. Let $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ be an icTA. Then, given a clock valuation $\nu : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ and a tuple $t \in \mathbb{R}_{\geq 0}^{Proc}$, the valuation $\nu + t$ is defined by $(\nu + t)(x) = \nu(x) + t_{\pi(x)}$ for all $x \in \mathcal{Z}$.

Thus, for $\tau \in Rates$, we define a *τ-run* of $\mathcal{B}$ as a sequence

$$(s_0, \nu_0) \xrightarrow{a_1, t_1} (s_1, \nu_1) \xrightarrow{a_2, t_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n, t_n} (s_n, \nu_n)$$

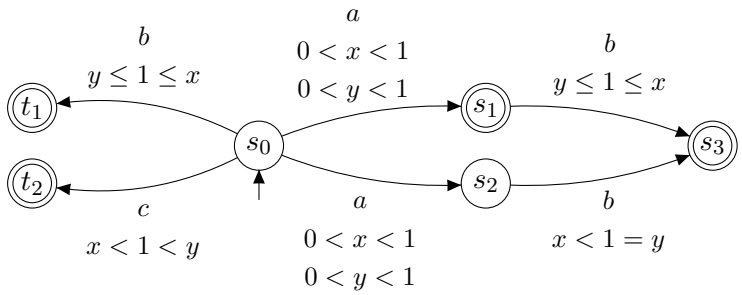


Figure 3. An icTA $\mathcal{B}$ with independent clocks x and y

where $n \geq 0$, $s_i \in S$, $a_i \in \Sigma_\varepsilon$, and $(t_i)_{1\leq i\leq n}$ is a non-decreasing sequence of values from $\mathbb{R}_{\geq 0}$ (we assume $t_0 = 0$). Further, $\nu_i : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ with $\nu_0(x) = 0$ for all $x \in \mathcal{Z}$. Finally, for all $i \in \{1, \ldots, n\}$, there are $\varphi_i \in \mathrm{Form}(\mathcal{Z})$ and $R_i \subseteq \mathcal{Z}$ such that the following conditions hold:

$$(s_{i-1}, a_i, \varphi_i, R_i, s_i) \in \delta \tag{1}$$

$$\nu_{i-1} + \tau(t') - \tau(t_{i-1}) \models I(s_{i-1}) \qquad \text{for each } t' \in [t_{i-1}, t_i] \tag{2}$$

$$\nu_{i-1} + \tau(t_i) - \tau(t_{i-1}) \models \varphi_i \tag{3}$$

$$\nu_i = (\nu_{i-1} + \tau(t_i) - \tau(t_{i-1}))[R_i] \tag{4}$$

$$\nu_i \models I(s_i) \tag{5}$$

In this case, we write $(\mathcal{B}, \tau) : s_0 \xrightarrow{a_1 \cdot \ldots \cdot a_n} s_n$ or also $(\mathcal{B}, \tau) : s_0 \xrightarrow{a_1 \cdot \ldots \cdot a_i} s_i \xrightarrow{a_{i+1} \cdot \ldots \cdot a_n} s_n$ to abstract from the time instances. The latter thus denotes that $\mathcal{B}$ can, reading w, go from s_0 via s_i to s_n, while respecting the local-time rates τ.

Definition 2.3. Let $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ be an icTA and $\tau \in \mathit{Rates}$. The *language* of $\mathcal{B}$ wrt. τ, denoted by $L(\mathcal{B}, \tau)$, is the set of words $w \in \Sigma^*$ such that $(\mathcal{B}, \tau) : \iota \xrightarrow{w} s$ for some $s \in F$. Moreover, we define $L_\exists(\mathcal{B}) = \bigcup_{\tau \in \mathit{Rates}} L(\mathcal{B}, \tau)$ to be the *existential semantics* and $L_\forall(\mathcal{B}) = \bigcap_{\tau \in \mathit{Rates}} L(\mathcal{B}, \tau)$ to be the *universal semantics* of $\mathcal{B}$.

If $|\mathit{Proc}| = 1$, then an icTA $\mathcal{B}$ actually reduces to an ordinary timed automaton and we have $L_\forall(\mathcal{B}) = L(\mathcal{B}, \tau) = L_\exists(\mathcal{B})$ for any $\tau \in \mathit{Rates}$. Moreover, if $|\mathit{Proc}| \geq 1$ and $\tau \in \mathit{Rates}$ exhibits, for all $p \in \mathit{Proc}$, the same local time evolution, then $L(\mathcal{B}, \tau)$ is the untimed language of $\mathcal{B}$ considered as an ordinary timed automaton.

Example 2.2. Consider an example icTA $\mathcal{B}$ over the set of processes $\{p, q\}$ and $\Sigma = \{a, b, c\}$ as depicted in Figure 3. Assuming $\pi(x) = p$ and $\pi(y) = q$, we have $L(\mathcal{B}, \mathrm{id}) = \{a, ab, b\}$, where id_p is the identity on $\mathbb{R}_{\geq 0}$ for all $p \in \mathit{Proc}$ (i.e., id models synchronization of any process with the absolute time). We observe that $L_\forall(\mathcal{B}) = \{a, ab\}$ and $L_\exists(\mathcal{B}) = \{a, ab, b, c\}$.

Now, we define the semantics of a DTA in terms of an associated icTA. As usual in a distributed (timed and untimed) setting, we give an interleaving semantics [4, 19]. It is obtained by taking a product of the components of the DTA that is slightly more complicated than a direct asynchronous product, in the sense that it simulates the simultaneous firing of independent actions (as in the DTA).

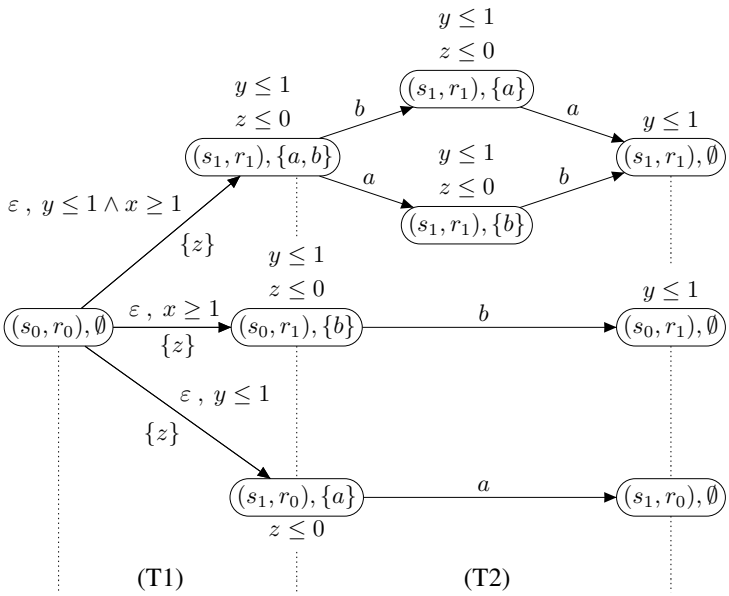


Figure 4. Part of the icTA $\mathcal{B}_\mathcal{D}$ for the DTA $\mathcal{D}$ from Figure 1

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ be a DTA where $\mathcal{A}_p = (S_p, \Sigma_p, \mathcal{Z}_p, \delta_p, I_p, \iota_p, F_p)$. We associate with $\mathcal{D}$ an icTA $\mathcal{B}_\mathcal{D} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi')$ as follows. Its set of states is $S = (\prod_{p \in Proc} S_p) \times 2^\Sigma$ where $\Sigma = \biguplus_{p \in Proc} \Sigma_p$. The first component of a state collects the current local state of every process. The intuitive meaning of the second component is the following. Being in a state $(s, \emptyset)$, the icTA guesses a set of transitions that the DTA may fire simultaneously. It then collects corresponding actions (except ε) into the set A, going into some state (s', A) where s' contains the target states of all transitions that had been selected. Now, as long as $A \neq \emptyset$, we use a clock invariant to ensure that the icTA can simulate all these actions, in any order, without time elapsing in between. In doing so, the icTA will eventually enter $(s', \emptyset)$. To ensure that the steps that are taken when $A \neq \emptyset$ occur instantaneously, we add an extra clock $z \notin \bigcup_{p \in Proc} \mathcal{Z}_p$ and we set $\mathcal{Z} = \{z\} \uplus \bigcup_{p \in Proc} \mathcal{Z}_p$. Thus, with our assumption stating that if time elapses, then it must elapse in all processes, we can assume z to belong to any process, i.e, $\pi(z)$ to be arbitrary.

For $s = (s_p)_{p \in Proc} \in \prod_{p \in Proc} S_p$ and $A \subseteq \Sigma$ with $A \neq \emptyset$, we define the state invariants $I(s, \emptyset) = \bigwedge_{p \in Proc} I_p(s_p)$ and $I(s, A) = z \le 0 \wedge \bigwedge_{p \in Proc} I_p(s_p)$. Moreover, we set $\iota = ((\iota_p)_{p \in Proc}, \emptyset)$, and $F = (\prod_{p \in Proc} F_p) \times \{\emptyset\}$. Then, the transitions in $\mathcal{B}_\mathcal{D}$ are of two types:

(T1) The first type is an ε-move, which guesses the set of processes of the DTA that will move next and the transitions that each of them would perform. In addition, it checks the guard that each of them must satisfy and resets the clocks as well. Thus, $((s, \emptyset), \varepsilon, \varphi, R, (s', A)) \in \delta$ if there are some non-empty set $P \subseteq Proc$ and transitions $(\tilde{s}_p, a_p, \varphi_p, R_p, \tilde{s}'_p) \in \delta_p$, $p \in P$, such that

- $s_p = \tilde{s}_p$ and $s'_p = \tilde{s}'_p$ for all $p \in P$, and $s_q = s'_q$ for all $q \in Proc \setminus P$,
- $\varphi = \bigwedge_{p \in P} \varphi_p$, $R = \bigcup_{p \in P} R_p \cup \{z\}$, and $A = \{a_p \mid p \in P\} \setminus \{\varepsilon\}$.

(T2) This move performs an action from its guessed set A and then removes it: $((s, A), a, \mathrm{true}, \emptyset, (s, A \setminus \{a\})) \in \delta$ for all $s \in \prod_{p \in Proc} S_p$, $A \subseteq \Sigma$, and $a \in A$.

This completes our definition of the icTA $\mathcal{B}_\mathcal{D}$ associated to a DTA $\mathcal{D}$.

Note that several different semantics in terms of icTA are conceivable here, and that their choice might depend on the concrete application or property to check. For example, one might require that the set of actions $P \subseteq Proc$ in condition (T1) be maximal so that a maximal coalition of processes fires simultaneously. Both our positive and negative results, however, do not depend on this choice. The positive result holds for icTA anyway. The undecidability result relies on a minimalistic encoding of the PCP in terms of a DTA, in which the given semantics as well as the maximal-step semantics coincide.

Example 2.3. The Figure 4 illustrates how this construction works on the DTA $\mathcal{D}$ from Figure 1. The picture illustrates how each move of $\mathcal{D}$ is in fact split into two phases. In the first phase, from a product of local states of the DTA, transition (T1) chooses the set of transitions that may fire next. Then in the second phase, depending on this set we have a sequence of (T2) transitions. Indeed, the second phase occurs without time elapse since clock z is reset at (T1) and checked to be zero by means of invariants in intermediate states.

Now we can define the language(s) of a DTA.

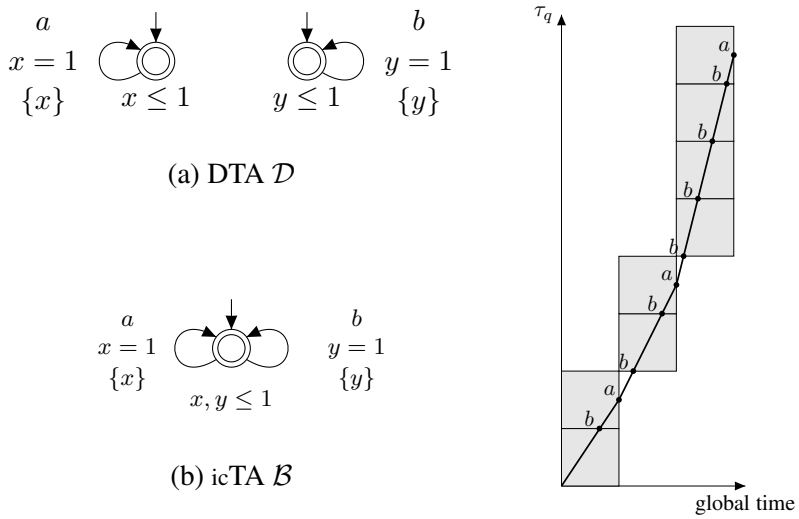
Definition 2.4. For a DTA $\mathcal{D}$ and $\tau \in Rates$, we set $L(\mathcal{D}, \tau) = L(\mathcal{B}_\mathcal{D}, \tau)$ to be the *language* of $\mathcal{D}$ wrt. τ, and we define $L_\exists(\mathcal{D}) = L_\exists(\mathcal{B}_\mathcal{D})$ as well as $L_\forall(\mathcal{D}) = L_\forall(\mathcal{B}_\mathcal{D})$ to obtain its existential and universal semantics, respectively.

Example 2.4. For the DTA $\mathcal{D}$ that is given in Figure 1, we can now formalize what we had described intuitively: $L(\mathcal{D}, \mathrm{id}) = \mathit{Pref}(\{aab, aba, baa\})$, $L_\exists(\mathcal{D}) = \mathit{Pref}(\{aab, abab, baab\})$, and $L_\forall(\mathcal{D}) = \mathit{Pref}(\{aa\})$ where, for $L \subseteq \Sigma^*$, $\mathit{Pref}(L)$ is the set of prefixes of words in L.

It is worthwhile to observe that $L(\mathcal{D}, \tau)$ can, in general, have bizarre (non-regular) behavior, if τ is itself a "weird" function. This is one more reason to look at the existential and universal semantics. Let us illustrate this with an example.

Example 2.5. Consider the simple DTA $\mathcal{D}$ in Figure 5 over $Proc = \{p, q\}$, where $\Sigma = \{a, b\}$, $\pi(x) = p$ and $\pi(y) = q$. The icTA $\mathcal{B}$ depicted in Figure 5 is a simplified version of $\mathcal{B}_\mathcal{D}$ where all the intermediate states and transitions have been removed. Indeed $L(\mathcal{B}, \tau) = L(\mathcal{B}_\mathcal{D}, \tau) = L(\mathcal{D}, \tau)$ for any $\tau \in Rates$. Now, let $\tau = (\mathrm{id}_p, \tau_q)$, where τ_q is any continuous, strictly increasing function such that $\tau_q(0) = 0$ and $\tau_q(n) = 2^n - 0.5$ for all $n \geq 1$. This is seen in the adjoining graph in Figure 5. Then, an a occurs at every local time unit of p (which is the same as a unit of global time), and a b occurs at every local time unit of q. Thus, $L(\mathcal{B}, \tau) = L(\mathcal{D}, \tau)$ is the set of finite prefixes of the infinite word $bab^2ab^4ab^8ab^{16}a\ldots$, which is not a regular language.

We end this section by noting that icTAs, in fact, capture the behavior of several variants of a shared-memory model and their semantics. For instance, in the spirit of *asynchronous automata* [19], we could have considered a distributed timed automaton to be a tuple $\big((S_p)_p, (\Sigma_p)_p, \mathcal{Z}, (\delta_a)_a, (I_p)_p, \iota, (F_p)_p, \pi\big)$, where the alphabets Σ_p are not necessarily disjoint, a ranges over $\Sigma = \bigcup_{p \in Proc} \Sigma_p$, $\iota \in \prod_{p \in Proc} S_p$, and



(a) DTA $\mathcal{D}$

(b) icTA $\mathcal{B}$

Figure 5. An example of "weird" behaviour

$\pi : \mathcal{Z} \to Proc$. This models a shared-memory system: executing an action $a \in \Sigma$ does not only affect one single process but rather involves each process from $proc(a) = \{p \in Proc \mid a \in \Sigma_p\}$. Therefore,

$$\delta_a \subseteq S_a \times \mathrm{Form}(\mathcal{Z}) \times 2^{\mathcal{Z}_a} \times S_a$$

where $S_a = \prod_{p \in proc(a)} S_p$ and $\mathcal{Z}_a = \bigcup_{p \in proc(a)} \pi^{-1}(p)$. Such a model of asynchronous DTA can be easily translated to our icTA, by additionally ensuring that on shared actions all processes sharing the action evolve simultaneously. The model from [4] corresponds to such an asynchronous automaton except for two differences: (1) clocks are local in the sense that they can only be read by those processes to which they belong, and (2) each process comes with a distinguished clock that is never reset; a synchronizing transition from δ_a can then be performed only if the special clocks that are associated with processes from $proc(a)$ exhibit the same value.

3. The existential semantics and the region abstraction

Given an icTA $\mathcal{B}$ (which might arise from some DTA $\mathcal{D}$) and a set Bad of undesired behaviors, it is natural to ask if $\mathcal{B}$ is robust against the (unknown) relative clock speeds and faithfully avoids executing action sequences from Bad. This corresponds to checking if $L_\exists(\mathcal{B}) \cap Bad = \emptyset$. In this section, we show that this question is indeed decidable, given that Bad is a regular language. To this aim, we define a partitioning of clock valuations into finitely many equivalence classes and generalize the well-known region construction for timed automata [2].

But first, we give an alternate view of the icTA semantics as an infinite-state transition system. Given an icTA $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$, we associate the transition system $TS(\mathcal{B})$ as follows: A state of $TS(\mathcal{B})$ is a tuple (s, ν), where $s \in S$ and $\nu : \mathcal{Z} \to \mathbb{R}_{\geq 0}$. Then, for $a \in \Sigma_\varepsilon$, $(s, \nu) \xrightarrow{a} (s', \nu')$ is a

transition in $TS(\mathcal{B})$ if there exist $t \in \mathbb{R}_{\geq 0}$, $\tau \in Rates$, $\varphi \in \mathrm{Form}(\mathcal{Z})$, and $R \subseteq \mathcal{Z}$ such that:

$$(s, a, \varphi, R, s') \in \delta \tag{6}$$
$$\nu + \tau(t') \models I(s) \qquad \text{for each } t' \in [0, t] \tag{7}$$
$$\nu + \tau(t) \models \varphi \tag{8}$$
$$\nu' = (\nu + \tau(t))[R] \tag{9}$$
$$\nu' \models I(s') \tag{10}$$

The initial state is (ι, ν_0) with $\nu_0(x) = 0$ for all $x \in \mathcal{Z}$. A state (s, ν) is final if $s \in F$. A run of $TS(\mathcal{B})$ on $w = a_1 \dots a_n \in \Sigma^*$ is a sequence of transitions,

$$(s_0, \nu_0) \xrightarrow{a_1} (s_1, \nu_1) \xrightarrow{a_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n} (s_n, \nu_n)$$

where $n \geq 0$. It is accepting if $s_n \in F$ and in this case we say $w \in L(TS(\mathcal{B}))$.

Proposition 3.1. $L(TS(\mathcal{B})) = L_{\exists}(\mathcal{B})$.

Proof:
Consider $w \in L_{\exists}(\mathcal{B})$. Then $w \in L(\mathcal{B}, \tau)$ for some τ and we find an accepting τ-run of $\mathcal{B}$:

$$(s_0, \nu_0) \xrightarrow{a_1, t_1} (s_1, \nu_1) \xrightarrow{a_2, t_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n, t_n} (s_n, \nu_n)$$

with $a_i \in \Sigma_\varepsilon$, $w = a_1 \dots a_n$ and such that (1–5) hold for some $\varphi_i \in \mathrm{Form}(\mathcal{Z})$ and $R_i \subseteq \mathcal{Z}$. We show that, abstracting away from the t_i's, we obtain an accepting run of $TS(\mathcal{B})$. For $1 \leq i \leq n$, we define $\hat{t}_i = t_i - t_{i-1}$ (with $t_0 = 0$) and τ_i by $\tau_i(t) = \tau(t_{i-1} + t) - \tau(t_{i-1})$. From (2–4), we obtain

$$\nu_{i-1} + \tau_i(t') \models I(s_{i-1}) \qquad \text{for each } t' \in [0, \hat{t}_i] \tag{11}$$
$$\nu_{i-1} + \tau_i(\hat{t}_i) \models \varphi_i \tag{12}$$
$$\nu_i = (\nu_{i-1} + \tau_i(\hat{t}_i))[R_i] \tag{13}$$

Therefore, $(s_0, \nu_0) \xrightarrow{a_1} (s_1, \nu_1) \xrightarrow{a_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n} (s_n, \nu_n)$ is an accepting run of $TS(\mathcal{B})$ for w.

Conversely, let $w = a_1 \dots a_n \in L(TS(\mathcal{B}))$ and let

$$(s_0, \nu_0) \xrightarrow{a_1} (s_1, \nu_1) \xrightarrow{a_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n} (s_n, \nu_n)$$

be an accepting run of $TS(\mathcal{B})$ for w. By definition, for each $1 \leq i \leq n$, we find $\hat{t}_i \geq 0$, τ_i, φ_i, and R_i such that (1,5,11–13) are satisfied. We define now by induction the non-decreasing sequence $(t_i)_{0 \leq i \leq n}$ by $t_0 = 0$ and $t_i = t_{i-1} + \hat{t}_i$ for $1 \leq i \leq n$. We also define τ in order to obtain a τ-run of $\mathcal{B}$: for $1 \leq i \leq n$ and $t \in [t_{i-1}, t_i]$, we let $\tau(t) = \tau(t_{i-1}) + \tau_i(t - t_{i-1})$; and for $t \geq t_n$, we let $\tau(t) = \tau(t_n) + \mathrm{id}(t - t_n)$. Then, we can easily check using (11–13) that (2–4) are satisfied. Therefore,

$$(s_0, \nu_0) \xrightarrow{a_1, t_1} (s_1, \nu_1) \xrightarrow{a_2, t_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n, t_n} (s_n, \nu_n)$$

is an accepting τ-run of $\mathcal{B}$. □

In order to prove that $L_\exists(\mathcal{B})$ is regular, we define on $TS(\mathcal{B})$ a bisimulation of finite index which preserves final states. In this way, we obtain as a quotient a finite automaton accepting $L(TS(\mathcal{B})) = L_\exists(\mathcal{B})$. As one may expect, this bisimulation is based on clock regions that we define below.

For each clock $x \in \mathcal{Z}$, let C_x be the largest constant clock x is compared with in guards or invariants. Let $p \in Proc$. As before, $\pi^{-1}(p) = \{z \in \mathcal{Z} \mid \pi(z) = p\}$ denotes the set of clocks owned by p. Given a clock valuation ν over $\mathcal{Z}$, define its p-restriction $\nu_p : \pi^{-1}(p) \to \mathbb{R}_{\geq 0}$ by $\nu_p(x) = \nu(x)$ for all $x \in \pi^{-1}(p)$. Then, from the classical region construction for timed automata, we obtain a notion of equivalence $\sim_p$ between two such valuations. That is, we say that $\nu_p \sim_p \nu'_p$ if the following hold:

1. for each $x \in \pi^{-1}(p)$, $\nu_p(x) > C_x$ if and only if $\nu'_p(x) > C_x$,
2. for each $x \in \pi^{-1}(p)$, $\nu_p(x) \leq C_x$ implies both $\lfloor \nu_p(x) \rfloor = \lfloor \nu'_p(x) \rfloor$ and $fract(\nu_p(x)) = 0$ if and only if $fract(\nu'_p(x)) = 0$, and
3. for each pair $x, y \in \pi^{-1}(p)$ such that $\nu_p(x) \leq C_x$ and $\nu_p(y) \leq C_y$, we have $fract(\nu_p(x)) \leq fract(\nu_p(y))$ if and only if $fract(\nu'_p(x)) \leq fract(\nu'_p(y))$. Note that this constraint only applies to clocks that belong to the same process.

From the result on timed automata [2], it follows that each $\sim_p$ is an equivalence relation and also a *time-abstract bisimulation*, i.e, if $\nu_p \sim_p \nu'_p$, then for all $t \in \mathbb{R}_{>0}$, there exists $t' \in \mathbb{R}_{>0}$ such that $\nu_p + t \sim_p \nu'_p + t'$.

Now, we say that two clock valuations ν and ν' over $\mathcal{Z}$ are *equivalent*, denoted $\nu \sim \nu'$ if they are equivalent when restricted to each process, i.e, $\nu_p \sim_p \nu'_p$ for all $p \in Proc$. An equivalence class of a clock valuation is called a *clock region* (of $\mathcal{B}$). For a valuation ν, $[\nu]$ denotes the clock region that contains ν. The set of clock regions of $\mathcal{B}$ is denoted by $Regions(\mathcal{B})$.

Fact 3.1. The number of clock regions is finite:

$$|Regions(\mathcal{B})| \leq (2\,C + 2)^{|\mathcal{Z}|} \cdot |\mathcal{Z}|!$$

where C is the largest constant that a clock is compared with in $\mathcal{B}$.

Clearly, equivalent valuations satisfy the same guards and invariants: if $\nu \sim \nu'$ then $\nu \models \varphi$ if and only if $\nu' \models \varphi$ for all $\varphi \in \mathrm{Form}(\mathcal{Z})$. Moreover,

Lemma 3.1. (Time-abstract bisimulation)
If $\nu \sim \nu'$, then for all $t \in \mathbb{R}_{>0}^{Proc}$, there exists $t' \in \mathbb{R}_{>0}^{Proc}$ such that $\nu + t \sim \nu' + t'$.

Proof:
Let $\nu, \nu' : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ and $t = (t_p)_{p \in Proc}$. Then $\nu \sim \nu'$ implies that $\nu_p \sim_p \nu'_p$ for each $p \in Proc$. Then, since each $\sim_p$ is a time-abstract bisimulation, for each $p \in Proc$, there exists $t'_p \in \mathbb{R}_{>0}$ such that $\nu_p + t_p \sim \nu'_p + t'_p$. Thus, defining $t' = (t'_p)_{p \in Proc}$ we obtain $\nu + t \sim \nu' + t'$. □

This equivalence can be naturally extended to states of $TS(\mathcal{B})$ by $(s, \nu) \sim (s', \nu')$ if $s = s'$ and $\nu \sim \nu'$. In order to show that this defines a bisimulation on $TS(\mathcal{B})$ (Proposition 3.2), we first introduce the successor relation on regions.

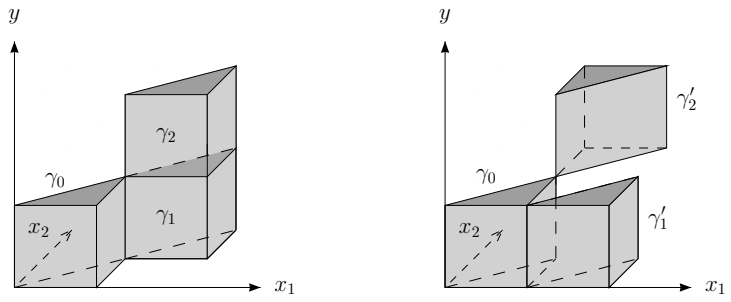


Figure 6. Accessible and non-accessible regions

Let γ and γ' be two clock regions. We say that γ' is *accessible* from γ, written $\gamma \preceq \gamma'$, if either $\gamma = \gamma'$ or there are $\nu \in \gamma, \nu' \in \gamma', t \in \mathbb{R}_{>0}^{Proc}$ such that $\nu' = \nu + t$. Note that $\preceq$ is a partial-order relation. The *successor* relation, written $\gamma \prec \gamma'$, is as usual defined by $\gamma \preceq \gamma'$, $\gamma \neq \gamma'$, and $\gamma'' = \gamma$ or $\gamma'' = \gamma'$ for all clock regions γ'' with $\gamma \preceq \gamma'' \preceq \gamma'$.

Example 3.1. The accessible-regions relation is illustrated in Figure 6. Suppose we deal with two processes, one owning clocks x_1 and x_2, the other owning a single clock y. Suppose furthermore that, in the icTA at hand, all clocks are compared to the constant 2. Consider the prisms $\gamma_0, \gamma_1, \gamma_2, \gamma_1', \gamma_2'$, each representing a non-border clock region, which are given by the following clock constraints:

$$\gamma_0 = (0 < x_2 < x_1 < 1) \wedge (0 < y < 1)$$

$$\gamma_1 = (1 < x_2 < x_1 < 2) \wedge (0 < y < 1)$$

$$\gamma_2 = (1 < x_2 < x_1 < 2) \wedge (1 < y < 2)$$

$$\gamma_1' = (0 < x_2 < x_1 - 1 < 1) \wedge (0 < y < 1)$$

$$\gamma_2' = (1 < x_1 < x_2 < 2) \wedge (1 < y < 2)$$

We have $\gamma_0 \preceq \gamma_1 \preceq \gamma_2$. However, $\gamma_0 \not\preceq \gamma_1'$ and $\gamma_0 \not\preceq \gamma_2'$.

Proposition 3.2. (Bisimulation)
If $(s, \nu) \sim (s, \hat{\nu})$ and $(s, \nu) \xrightarrow{a} (s', \nu')$ then $(s, \hat{\nu}) \xrightarrow{a} (s', \hat{\nu}')$ for some $\hat{\nu}' \sim \nu'$.

Proof:
Assume that $(s, \nu) \xrightarrow{a} (s', \nu')$. Let $t \in \mathbb{R}_{\geq 0}$, $\tau \in \mathit{Rates}$, $\varphi \in \mathrm{Form}(\mathcal{Z})$ and $R \subseteq \mathcal{Z}$ such that (6–10) hold. Consider the successive regions $\gamma_0 \prec \gamma_1 \prec \cdots \prec \gamma_n$ visited along $\nu + \tau[0 \ldots t]$: there is $0 = t_0 < t_1 < \cdots < t_n = t$ such that, for $0 \leq i \leq n$, we have $\gamma_i = [\nu_i]$ with $\nu_i = \nu + \tau(t_i)$; and moreover, for any $1 \leq i \leq n$ and all $t_{i-1} < t' < t_i$ we have $\nu + \tau(t') \in \gamma_{i-1} \cup \gamma_i$.

Assume now in addition that $(s, \nu) \sim (s, \hat{\nu})$. We construct $\hat{\tau}$ such that, for each $0 \leq i \leq n$, we have $P(i) : \hat{\nu}_i = \hat{\nu} + \hat{\tau}(t_i) \sim \nu_i$. We start with $\hat{\tau}(0) = 0$ so that $P(0)$ holds. Let now $1 \leq i \leq n$ and assume we have constructed $\hat{\tau}$ up to t_{i-1} with $P(i-1)$. Using Lemma 3.1 we find $\hat{t} \in \mathbb{R}_{>0}^{Proc}$ such that $\hat{\nu}_{i-1} + \hat{t} \sim \nu_{i-1} + \tau(t_i) - \tau(t_{i-1}) = \nu_i$. Then, define $\hat{\tau}$ on the interval $[t_{i-1}, t_i]$ using a linear

interpolation so that $\hat{\tau}(t_i) = \hat{\tau}(t_{i-1}) + \hat{t}$. We obtain $\hat{\nu}_{i-1} + \hat{t} = \hat{\nu} + \hat{\tau}(t_i) = \hat{\nu}_i$ and $P(i)$ also holds. Finally, for $t' \geq t_n = t$, we let $\hat{\tau}(t') = \hat{\tau}(t_n) + \mathrm{id}(t' - t_n)$.

For any $1 \leq i \leq n$ and all $t_{i-1} < t' < t_i$ we have

$$\gamma_{i-1} = [\hat{\nu}_{i-1}] \preceq [\hat{\nu} + \hat{\tau}(t')] \preceq [\hat{\nu}_i] = \gamma_i$$

and since $\gamma_{i-1} \prec \gamma_i$ we obtain $\hat{\nu} + \hat{\tau}(t') \in \gamma_{i-1} \cup \gamma_i$. Therefore, $\hat{\nu} + \hat{\tau}(t') \models I(s)$ for all $t' \in [0, t]$ and $\hat{\nu}_n = \hat{\nu} + \hat{\tau}(t) \models \varphi$. We let $\hat{\nu}' = \hat{\nu}_n[R] \sim \nu_n[R] = \nu'$. We have $\hat{\nu}' \models I(s')$ and we deduce that $(s, \hat{\nu}) \xrightarrow{a} (s', \hat{\nu}')$ in $TS(\mathcal{B})$. □

To obtain the main result of this section, it remains to consider the finite quotient $TS(\mathcal{B})/\!\sim = (S'', \Sigma, \delta'', \iota'', F'')$. A state in S'' is an equivalence class $[(s, \nu)]$ and we have a transition $[(s, \nu)] \xrightarrow{a} [(s', \nu')] \in \delta''$ whenever $(s, \nu) \xrightarrow{a} (s', \nu')$ is a transition of $TS(\mathcal{B})$. The initial state is indeed $\iota'' = [(\iota, \nu_0)]$ where (ι, ν_0) is the initial state of $TS(\mathcal{B})$. Moreover, a state $[(s, \nu)]$ is final if and only if $s \in F$. Since the bisimulation equivalence relation $\sim$ on $TS(\mathcal{B})$ preserves final states, we obtain by standard (and easy) arguments:

Corollary 3.1. $L(TS(\mathcal{B})/\!\sim) = L(TS(\mathcal{B}))$.

The finite quotient $TS(\mathcal{B})/\!\sim$ is not exactly what is usually called the *region automaton* in the classical theory of timed automata. The main difference is that, in the region automaton, transitions are decomposed into time-elapse ε-transitions from a region to a successor region, and discrete transitions with no time-elapse. The other minor difference is that we use as set of states $S' = S \times \mathit{Regions}(\mathcal{B})$ which is indeed isomorphic to S''. In particular, $\iota' = (\iota, [\nu_0])$ and $F' = F \times \mathit{Regions}(\mathcal{B})$. The region automaton associated with $\mathcal{B}$ is therefore $\mathcal{R}_\mathcal{B} = (S', \Sigma, \delta', \iota', F')$ where, for $a \in \Sigma_\varepsilon$ and $s, s' \in S$ and $\gamma, \gamma' \in \mathit{Regions}(\mathcal{B})$, δ' contains $(s, \gamma) \xrightarrow{a} (s', \gamma')$ if

- $a = \varepsilon$, $s = s'$, $\gamma \prec \gamma'$, and $\nu' \models I(s)$ for some $\nu' \in \gamma'$
  (we then call $(s, \gamma) \xrightarrow{\varepsilon} (s, \gamma')$ a *time-elapse transition*), or
- there are $\nu \in \gamma$ and $(s, a, \varphi, R, s') \in \delta$ such that $\nu \models \varphi \wedge I(s)$, $\nu[R] \models I(s')$, and $\nu[R] \in \gamma'$ (we then call $(s, \gamma) \xrightarrow{a} (s', \gamma')$ a *discrete transition*).

A part of the region automaton for the icTA from Figure 3 is shown in Figure 15.

It is easy to see that a sequence of time-elapse transitions followed by a discrete transition of $\mathcal{R}_\mathcal{B}$ is a transition of $TS(\mathcal{B})/\!\sim$. Conversely, any transition of $TS(\mathcal{B})/\!\sim$ can be decomposed into a sequence of time-elapse transitions followed by a discrete transition of $\mathcal{R}_\mathcal{B}$. Therefore:

Theorem 3.1. Let $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ be an icTA and let C be the largest constant a clock is compared with in $\mathcal{B}$. Then, the number of states of $TS(\mathcal{B})/\!\sim$ and of $\mathcal{R}_\mathcal{B}$ is bounded by $|S| \cdot (2\,C + 2)^{|\mathcal{Z}|} \cdot |\mathcal{Z}|!$ and we have

$$L(\mathcal{R}_\mathcal{B}) = L(TS(\mathcal{B})/\!\sim) = L(TS(\mathcal{B})) = L_\exists(\mathcal{B})$$

which is therefore a regular word language.

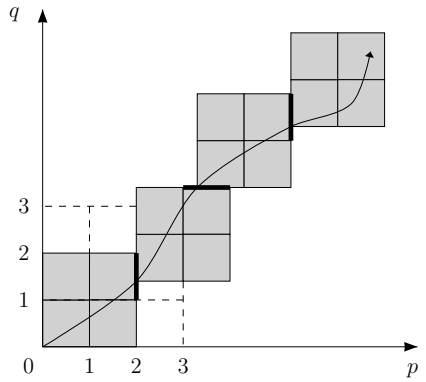


Figure 7. $dir(\tau) = 010\ldots$

Now, suppose we are given a negative specification as a regular set Bad. Then, since $L_\exists(\mathcal{B})$ is a regular word language, so is $L_\exists(\mathcal{B}) \cap Bad$ and hence we can check emptiness and so on. Thus, model checking icTAs wrt. regular negative specifications is decidable:

Corollary 3.2. The questions $L_\exists(\mathcal{B}) \cap R$? and $R \subseteq L_\exists(\mathcal{B})$? are decidable for given icTA $\mathcal{B}$ and regular language R.

4. The universal semantics

While the existential semantics allows us to verify negative specifications, the universal semantics is natural when we want to check if our system has some good behavior. By good we mean a behavior that is robust against clock variations. We show in this section that emptiness and universality are unfortunately undecidable for the universal semantics. This is shown for icTAs first and then will be extended to DTAs. Therefore, it is undecidable if, for a positive specification $Good$ containing the behaviors that a system *must* exhibit and a DTA $\mathcal{D}$, we have $Good \subseteq L_\forall(\mathcal{D})$.

Theorem 4.1. The following problem is undecidable if $|Proc| \geq 2$: Given an icTA $\mathcal{B}$ over $Proc$, does $L_\forall(\mathcal{B}) \neq \emptyset$ hold?

Proof:
The proof is by reduction from Post's correspondence problem (PCP). An instance $Inst$ of the PCP consists of an alphabet A and two morphisms f and g from A^+ to $\{0,1\}^+$. A solution of $Inst$ is a word $w \in A^+$ such that $f(w) = g(w)$.

Suppose $Proc = \{p, q\}$ and let $\tau \in Rates$. One may associate with τ two sequences $t\text{-}dir(\tau) = t_1 t_2 \ldots \in (\mathbb{R}_{\geq 0})^\omega$ of time instances and $dir(\tau) = d_1 d_2 \ldots \in \{0,1,2\}^\omega$ of *directions* as follows: for $i \geq 1$, we let first (assuming $t_0 = 0$) $t_i = \min\{t > t_{i-1} \mid \tau_r(t) - \tau_r(t_{i-1}) = 2 \text{ for some } r \in Proc\}$.

With this, we set

$$d_i = \begin{cases} 0 & \text{if } \tau_p(t_i) - \tau_p(t_{i-1}) = 2 \text{ and } 1 < \tau_q(t_i) - \tau_q(t_{i-1}) < 2 \\ 1 & \text{if } \tau_q(t_i) - \tau_q(t_{i-1}) = 2 \text{ and } 1 < \tau_p(t_i) - \tau_p(t_{i-1}) < 2 \\ 2 & \text{otherwise} \end{cases}$$

The construction of $dir(\tau)$ is illustrated in Figure 7. The idea is to allow the shape of the relative time-rate function (from τ) to encode a word in $\{0,1,2\}^\omega$. We do this using 2×2-square regions, each consisting of 4 sub-squares as shown. If the rate function leaves this region by the upper boundary or right boundary of the right-upper sub-square, then we write 1 or 0, respectively. If it leaves by any other boundary or by end-points of any sub-square, then we write 2. A new square region is started at the point where the rate function left the old one. Thus, the direction sequences partition the space of time rates.

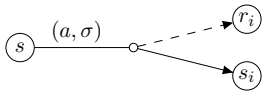
Roughly speaking, a word is accepted universally by an icTA if and only if it is accepted for all directions. Our trick will be to define an icTA such that the PCP instance has a solution w if and only if the word wb is accepted by the icTA for all directions. Thus, if there is no solution to the PCP, there will be some direction sequence (respectively, local time rates) for which the icTA does not accept.

Let an instance $Inst$ of the PCP be given by an alphabet $A = \{a_1, \ldots, a_k\}$ with $k \geq 1$ and two corresponding morphisms f and g. We will construct an icTA $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ over the set of processes $Proc = \{p, q\}$ and $\Sigma = \{a_1, \ldots, a_k, b\}$ such that $L_\forall(\mathcal{B}) = \{wb \mid w \in A^+ \text{ and } f(w) = g(w)\}$. First, let $\mathcal{Z} = \{x, y\}$ with $\pi(x) = p$ and $\pi(y) = q$. For $d \in \{0,1,2\}$, we set

$$guard(d) = \begin{cases} x = 2 \ \wedge \ 1 < y < 2 & \text{if } d = 0 \\ y = 2 \ \wedge \ 1 < x < 2 & \text{if } d = 1 \\ ((x \leq 1 \ \vee \ x = 2) \ \wedge y = 2) \ \vee \ (y \leq 1 \ \wedge \ x = 2) & \text{if } d = 2 \end{cases}$$

Moreover, let $\overline{guard}(d) = \bigvee_{d' \in \{0,1,2\} \setminus \{d\}} guard(d')$.

The encoding of the given PCP instance in terms of the icTA is given by Figure 9. Hereby, given $a \in A$, $\sigma = d_1 \ldots d_n \in \{0,1,2\}^+$ (with $d_j \in \{0,1,2\}$ for any $j \in \{1, \ldots, n\}$) and $i \in \{1,2\}$, a transition of the form



will actually stand for the sequence of transitions that is depicted in Figure 8, say, with intermediate states $s_{(i,a,\sigma,1)}, \ldots, s_{(i,a,\sigma,n-1)}$.

Example 4.1. Consider the PCP instance $Inst$ given by $A = \{a_1, a_2\}$, $f(a_1) = 101$, $g(a_1) = 1$, $f(a_2) = 1$, $g(a_2) = 01110$ with the solution $w = a_1a_2a_1$. One can check that $a_1a_2a_1b \in L_\forall(\mathcal{B})$. This is illustrated in Figure 10. In the tree depicted, any path corresponds to a finite prefix (of length $|f(w)|+1$) of some sequence of directions. The edges are labeled by this sequence, where a left-edge is 0, downward is 2 and right-edge is 1. Thus, intuitively, a word wb is in the universal language if and only if all paths of the tree correspond to accepting runs in $\mathcal{B}$. Now, let us verify that the word wb is accepted by $\mathcal{B}$. If clock rate τ is such that $dir(\tau) \in f(w) \cdot d \cdot \{0,1,2\}^\omega$ with $d \in \{0,1\}$, then the accepting run of $\mathcal{B}$ is the path

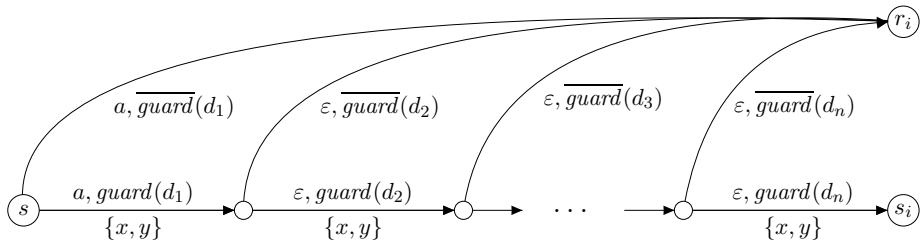


Figure 8. Transition macro

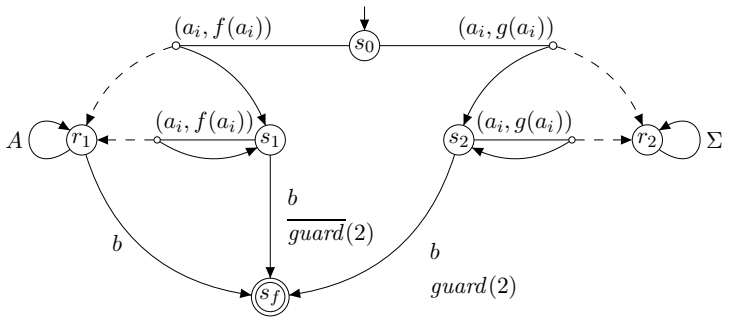


Figure 9. Encoding of PCP: icTA $\mathcal{B}$

shown in the left figure, which assigns states s_1 to nodes of the tree and finishes at s_f. If $d = 2$, then the accepting run of $\mathcal{B}$ is the path in the figure on right, which assigns states s_2 appropriately, crucially using the fact that $f(w) = g(w)$, and finally ends at s_f. If the clock rate τ has $dir(\tau)$ different from the above cases, it is easy to see that there is an accepting run in which $\mathcal{B}$ reaches state s_f by passing through state r_1.

Thus, for a given rate τ, the left branch of the automaton given in Figure 9 accepts all words wb such that $f(w) \cdot 2 \not\leq dir(\tau)$, while the right branch accepts words wb such that $g(w) \cdot 2 \leq dir(\tau)$. So if w is a solution, i.e., $f(w) = g(w)$, then wb is accepted for every possible τ, hence $wb \in L_\forall(\mathcal{B})$. And if w is not a solution, i.e., $f(w) \neq g(w)$, then for some τ such that $f(w) \cdot 2 \leq dir(\tau)$, wb will not be accepted, hence $w \notin L_\forall(\mathcal{B})$, which gives our desired reduction.

Let us now show that our reduction is formally correct. In the following, let $\leq$ denote the usual prefix relation on words. We begin by observing that if a τ-run starting from s_0 on $\mathcal{B}$ satisfies the guards given by $guard(d)$ (and thus avoids states r_1 and r_2), then the time stamps of the run are exactly the ones given by $dir(\tau)$.

Fact 4.1. Let $\tau \in Rates$ and let $t\text{-}dir(\tau) = t_1 t_2 \ldots \in (\mathbb{R}_{\geq 0})^\omega$ and $dir(\tau) = d_1 d_2 \ldots \in \{0, 1, 2\}^\omega$ be the associated sequences. In addition, we set $t_0 = 0$. Consider a τ-run

$$(s_0, \nu_0) \xrightarrow{a_1, t'_1} (u_1, \nu_1) \ldots (u_{n-1}, \nu_{n-1}) \xrightarrow{a_n, t'_n} (u_n, \nu_n)$$

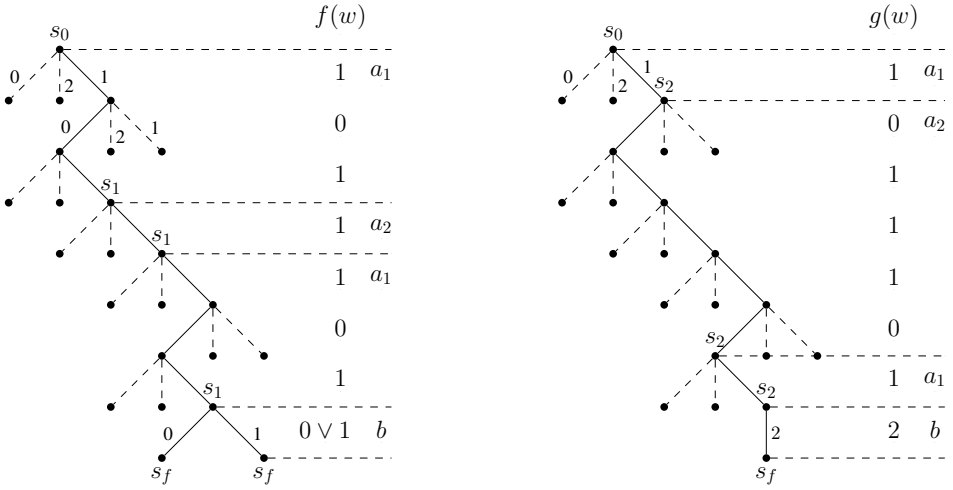


Figure 10. The tree generated by $w = a_1a_2a_1b$ with respect to f and g.

of $\mathcal{B}$ with $a_i \in \Sigma_\varepsilon$ and where $u_{n-1} \notin \{r_1, r_2, s_f\}$. Assume moreover $d'_1, \ldots, d'_n$ are the (unique) elements from $\{0, 1, 2\}$ such that $\nu_{i-1} + \tau(t'_i) - \tau(t'_{i-1}) \models \mathit{guard}(d'_i)$ for all $i \in \{1, \ldots, n\}$. Then, $t'_i = t_i$ and $d'_i = d_i$ for all $1 \leq i \leq n$, and $\nu_i = \nu_0$ for all $1 \leq i < n$.

Proof:
We proceed by induction. Assume $t'_{i-1} = t_{i-1}$ and $\nu_{i-1}(x) = \nu_{i-1}(y) = 0$ for some $1 \leq i \leq n$ (note that this is the case for $i = 1$). By assumption, $\tau(t'_i) - \tau(t_{i-1}) \models \mathit{guard}(d'_i)$. Here we consider the former expression as a valuation by considering the p-component of the pair as the clock value of x and the q-value as that of y. Now assume $d'_i = 0$ (the cases $d'_i \in \{1, 2\}$ are analogous). Then, $\tau_p(t'_i) - \tau_p(t_{i-1}) = 2$ and $1 < \tau_q(t'_i) - \tau_q(t_{i-1}) < 2$. Hence, $t'_i = \min\{t > t_{i-1} \mid \tau_r(t) - \tau_r(t_{i-1}) = 2 \text{ for some } r \in \mathit{Proc}\}$. We deduce $t'_i = t_i$ and $d'_i = d_i$. Moreover, $\nu_i(x) = \nu_i(y) = 0$ if $i < n$. □

Fact 4.2. For $\tau \in \mathit{Rates}$ and $w \in A^+$, the following hold (recall that $\leq$ denotes the prefix relation):

(1) $f(w) \leq \mathit{dir}(\tau)$ iff $(\mathcal{B}, \tau) : s_0 \xrightarrow{w} s_1$

(2) $g(w) \leq \mathit{dir}(\tau)$ iff $(\mathcal{B}, \tau) : s_0 \xrightarrow{w} s_2$

(3) $f(w) \not\leq \mathit{dir}(\tau)$ iff $(\mathcal{B}, \tau) : s_0 \xrightarrow{w} r_1$

Proof:
First note that (1) and (2) can be shown along the same lines. Moreover, their "if"- directions follow from Claim 4.1.

Let $\tau \in \mathit{Rates}$ and let $t\text{-}\mathit{dir}(\tau) = t_1t_2 \ldots \in (\mathbb{R}_{\geq 0})^\omega$ and $\mathit{dir}(\tau) = d_1d_2 \ldots \in \{0, 1, 2\}^\omega$ be the associated sequences. We also set $t_0 = 0$. Suppose $w = a_1 \ldots a_n$ where $a_i \in A$ for all $i \in \{1, \ldots, n\}$. For $i \in \{1, \ldots, n\}$, let moreover ℓ_i denote the length of $f(a_1 \ldots a_i)$. Finally, ν is the valuation with $\nu(x) = \nu(y) = 0$.

(1) Suppose $f(w) \leq dir(\tau)$. We have $f(w) = d_1 \dots d_{\ell_n}$. Let us check that

$$\begin{array}{lllll}
(s_0,\nu) & \xrightarrow{(a_1,t_1)} & (s_{(1,a_1,f(a_1),1)},\nu) & \xrightarrow{(\varepsilon,t_2)} \quad \dots \quad & \xrightarrow{(\varepsilon,t_{\ell_1})} (s_1,\nu) \\
& \xrightarrow{(a_2,t_{\ell_1+1})} & (s_{(1,a_2,f(a_2),1)},\nu) & \xrightarrow{(\varepsilon,t_{\ell_1+2})} \quad \dots \quad & \xrightarrow{(\varepsilon,t_{\ell_2})} (s_1,\nu) \\
& & \vdots & & \\
& \xrightarrow{(a_n,t_{\ell_{n-1}+1})} & (s_{(1,a_n,f(a_n),1)},\nu) & \xrightarrow{(\varepsilon,t_{\ell_{n-1}+2})} \quad \dots \quad & \xrightarrow{(\varepsilon,t_{\ell_n})} (s_1,\nu)
\end{array}$$

is a τ-run of $\mathcal{B}$, which implies $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_1$. We need to show that, for all $i \in \{1,\dots,\ell_n\}$, $\tau(t_i) - \tau(t_{i-1}) \models guard(d_i)$. So let $i \in \{1,\dots,\ell_n\}$. Assuming $d_i = 0$, we indeed have $\tau(t_i) - \tau(t_{i-1}) \models x = 2 \wedge 1 < y < 2$, whereas $d_i = 1$ implies $\tau(t_i) - \tau(t_{i-1}) \models y = 2 \wedge 1 < x < 2$.

(3) Let us assume $f(w) \not\leq dir(\tau)$. There is $j \in \{1\dots,\ell_n\}$ such that we have $f(w) = d_1 \dots d_{j-1} d'_j \dots d'_{\ell_n}$ for some $d'_j,\dots,d'_{\ell_n} \in \{0,1\}$ with $d'_j \neq d_j$. We construct a τ-run that coincides with the run that we constructed above until the $(j-1)$-th transition. Now, $\tau(t_j) - \tau(t_{j-1}) \models guard(d_j)$ and since $d'_j \neq d_j$, the j-th transition leads to state r_1. Once in r_1, we stay in r_1 under any timing. Thus, $(\mathcal{B},\tau) : s_0 \xrightarrow{w} r_1$.

Conversely, assume that $(\mathcal{B},\tau) : s_0 \xrightarrow{w} r_1$ and consider a τ-run

$$(s_0,\nu_0) \xrightarrow{a'_1,t'_1} (u_1,\nu_1) \xrightarrow{(a'_2,t'_2)} \cdots \xrightarrow{(a'_{j-1},t'_{j-1})} (u_{j-1},\nu_{j-1}) \xrightarrow{a'_j,t'_j} (r_1,\nu')$$

with $a'_1 \cdots a'_j \leq w$ and $u_{j-1} \neq r_1$. Let moreover $d'_1,\dots,d'_j$ be the (unique) elements from $\{0,1\}$ such that $\nu_{i-1} + \tau(t'_i) - \tau(t'_{i-1}) \models guard(d'_i)$ for all $1 \leq i \leq j$. By Claim 4.1, we deduce that $t_i = t'_i$ and $d'_i = d_i$ for all $1 \leq i \leq j$. Since the last transition reaches state r_1 from $u_{j-1} \neq r_1$, we deduce that $d_1 \cdots d_{j-1} \leq f(w)$ but d_j differs from the j-th letter of $f(w)$. Therefore, $f(w) \not\leq dir(\tau)$. □

With Claim 4.2, we can now show both directions of the correctness of the construction of $\mathcal{B}$, i.e., $L_\forall(\mathcal{B}) = \{wb \mid w \in A^+ \text{ and } f(w) = g(w)\}$.

Let $w \in A^+$ with $f(w) = g(w)$ and let $\tau \in Rates$. We distinguish three cases. If $dir(\tau) \in f(w)\cdot\{0,1\}\cdot\{0,1,2\}^\omega$, then $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_1 \xrightarrow{b} s_f$ by Claim 4.2 (1). If $dir(\tau) \in f(w)\cdot 2\cdot\{0,1,2\}^\omega$, then $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_2 \xrightarrow{b} s_f$ by Claim 4.2 (2), since $g(w) = f(w)$. If $f(w) \not\leq dir(\tau)$, then $(\mathcal{B},\tau) : s_0 \xrightarrow{w} r_1 \xrightarrow{b} s_f$ by Claim 4.2 (3). Hence, $wb \in L_\forall(\mathcal{B})$.

Conversely, let $w \in A^+$ and suppose $wb \in L_\forall(\mathcal{B})$. Pick $\tau \in Rates$ such that $dir(\tau) \in f(w)\cdot 2\cdot\{0,1,2\}^\omega$. As $f(w) \leq dir(\tau)$, we have $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_1$ and $(\mathcal{B},\tau) : s_0 \not\xrightarrow{w} r_1$ by Claim 4.2 (1,3), and since $f(w)\cdot 2 \leq dir(\tau)$ we deduce that $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_1 \not\xrightarrow{b}$. Thus, $(\mathcal{B},\tau) : s_0 \xrightarrow{w} s_2 \xrightarrow{b} s_f$. Hence, $g(w)\cdot 2 \leq dir(\tau)$ by Claim 4.2 (2). As $f(w), g(w) \in \{0,1\}^*$ and both $f(w)\cdot 2 \leq dir(\tau)$ and $g(w)\cdot 2 \leq dir(\tau)$, we deduce $f(w) = g(w)$. □

Thus, we have shown that it is undecidable to check the emptiness of the universal semantics. We also show that the universality problem for this case is undecidable.

Theorem 4.2. Suppose that $|Proc| \geq 2$. For icTA $\mathcal{B}$ over $Proc$, it is undecidable to know if $L_\forall(\mathcal{B}) = \Sigma^*$ (where Σ is the set of actions of $\mathcal{B}$).

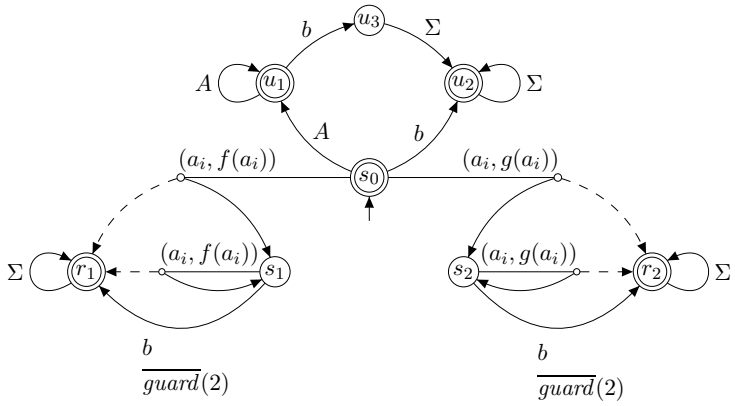


Figure 11. Encoding of PCP: icTA $\widetilde{\mathcal{B}}$

Proof:
As before, the reduction is from the PCP problem. The construction of the corresponding automaton is obtained by a slight modification of the automaton in the previous proof as we shall see below. We consider, as before, an instance of the PCP. We will then construct an icTA $\widetilde{\mathcal{B}}$ over $Proc = \{p, q\}$ and $\Sigma = A \cup \{b\}$, where $A = \{a_1, \ldots, a_k\}$, such that

$$L_\forall(\widetilde{\mathcal{B}}) = \Sigma^* \setminus \{wb \mid w \in A^+ \text{ and } f(w) = g(w)\}.$$

Thus, the PCP instance has a solution if and only if $L_\forall(\widetilde{\mathcal{B}}) \neq \Sigma^*$.

Define icTA $\widetilde{\mathcal{B}}$ as in Figure 11 where the transition macros are as defined in Figure 8. This automaton is almost the same as $\mathcal{B}$ in Figure 9, except for two differences. One is that we have switched the final states and b-transitions leading to them. Further, we have a 3-state gadget on the top of the previous automaton which does not use any clocks. In fact, this gadget is just the automaton which accepts the language $\Sigma^* \setminus A^+b$. Thus, if we have a word which is not in A^+b, it gets nondeterministically accepted by this gadget. If the word is in A^+b then it can only be accepted in states r_1 or r_2.

Fact 4.3. For $\tau \in Rates$ and $w \in A^+$, the following hold:

(a) $f(w) \cdot 2 \not\leq dir(\tau)$ iff $(\widetilde{\mathcal{B}}, \tau) : s_0 \xrightarrow{wb} r_1$

(b) $g(w) \cdot 2 \not\leq dir(\tau)$ iff $(\widetilde{\mathcal{B}}, \tau) : s_0 \xrightarrow{wb} r_2$

Proof:
The proof of both these statements follows that from Claim 4.2 which itself depends on Claim 4.1. But, since we have only fiddled with the final states/transitions, these claims still hold. □

We resume with the proof of Theorem 4.2. First, suppose the PCP has a solution, say $w \in A^+$, then consider the word wb and choose τ such that $dir(\tau) \in f(w) \cdot 2 \cdot \{0, 1, 2\}^\omega$. Then, by Claim 4.3, we have $(\widetilde{\mathcal{B}}, \tau) : s_0 \not\xrightarrow{wb} r_1$. Now, since w is assumed to be a solution, we have $f(w) = g(w)$ and

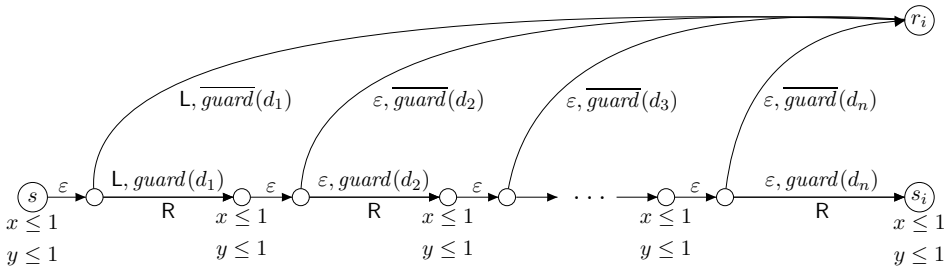


Figure 12. Transition macro for the distributed setting

$dir(\tau) \in g(w) \cdot 2 \cdot \{0,1,2\}^\omega$ and so again from Claim 4.3 we have $(\widetilde{\mathcal{B}}, \tau) : s_0 \xrightarrow{wb} r_2$. Further, on reading wb, $\widetilde{\mathcal{B}}$ can only (nondeterministically) reach the reject state u_3, not the accepting states u_1 or u_2 or s_0. Therefore, $wb \notin L(\widetilde{\mathcal{B}}, \tau)$ and so $L_\forall(\widetilde{\mathcal{B}}) \neq \Sigma^*$.

Conversely, suppose $L_\forall(\widetilde{\mathcal{B}}) \neq \Sigma^*$. Let $w' \in \Sigma^*$ such that $w' \notin L_\forall(\widetilde{\mathcal{B}})$. Necessarily, $w' = wb$ with $w \in A^+$, since otherwise it would be accepted by the gadget for all $\tau \in Rates$. Moreover, there exists τ such that $w' \notin L(\widetilde{\mathcal{B}}, \tau)$, i.e., after reading $w' = wb$, $\widetilde{\mathcal{B}}$ does not reach the accepting states r_1 or r_2. By Claim 4.3, we can now conclude that $f(w) \cdot 2 \leq dir(\tau)$ and $g(w) \cdot 2 \leq dir(\tau)$, which finally implies that $f(w) = g(w)$ and thus the PCP has a solution, namely w. □

In fact, the above proof also demonstrates that checking if the existential and universal semantics coincide is undecidable.

Corollary 4.1. Suppose that $|Proc| \geq 2$. For icTA $\mathcal{B}$ over $Proc$, it is undecidable to check if $L_\forall(\mathcal{B}) = L_\exists(\mathcal{B})$.

Proof:
Consider the icTA $\widetilde{\mathcal{B}}$ in Figure 11 constructed for the above proof. Then $L_\exists(\widetilde{\mathcal{B}}) = \Sigma^*$ and thus checking if $L_\forall(\widetilde{\mathcal{B}}) = L_\exists(\widetilde{\mathcal{B}})$ is the same as checking if $L_\forall(\widetilde{\mathcal{B}}) = \Sigma^*$. But this is undecidable by Theorem 4.2. □

These results can be strengthened and extended to the distributed setting as follows:

Theorem 4.3. Suppose $|Proc| \geq 2$. For DTAs $\mathcal{D}$ over $Proc$, the emptiness and universality of $L_\forall(\mathcal{D})$ are undecidable.

Proof:
We fix $Proc = \{p, q\}$ and the clock distribution $\pi(x) = p$ and $\pi(y) = q$. Each process will be a copy of the automaton $\mathcal{B}$ that is depicted in Figure 9 for emptiness (respectively, $\widetilde{\mathcal{B}}$ in Figure 11 for universality), except for one difference: in the copy $\mathcal{A}_p$ for process p, the transition macro from Figure 8 is replaced with that from Figure 12 where L is the letter $a \in A$ and R is the singleton set $\{x\}$; and in the copy $\mathcal{A}_q$ for process q, we use the same new macro, but now we have $\mathsf{L} = \varepsilon$ and $\mathsf{R} = \{y\}$.

The main difficulty is to make sure that transitions with $guard(d_j)$ or $\overline{guard}(d_j)$ are taken simultaneously in the two copies $\mathcal{A}_p$ and $\mathcal{A}_q$. If this is the case, then clock y is always reset synchronously with clock x and $\mathcal{A}_p$ faithfully simulates the icTA $\mathcal{B}$ (or $\widetilde{\mathcal{B}}$) with the slight difference induced by the additional

ε-transitions from the states with invariant $x \leq 1 \wedge y \leq 1$. Therefore, the proof of Theorems 4.1 or 4.2 can be carried out similarly.

We explain now how we make sure that transitions with guards are taken simultaneously. We have splitted each state of the transition macro described in Figure 8 (except s_i and r_i) into two. The combination of the guards and the invariants ensure that both clocks have been reset simultaneously.

Let us examine this in more detail. Being in two identical copies of a state with an invariant, the ε-transitions might indeed be taken asynchronously by $\mathcal{A}_p$ and $\mathcal{A}_q$. However, the following transitions will be performed synchronously. Assume first that p follows a transition of the form $(s_p, a, guard(d), \{x\}, s'_p)$ before process q moves. As $guard(d)$, where $d \in \{0, 1\}$, is satisfied when p goes to s'_p, the value of both clocks exceeds 1. But as x is reset at the same time whereas y is not, the invariant associated with s'_p is violated, which is a contradiction. Thus, q has to take the corresponding transition, which is of the form $(s_q, a, guard(d), \{y\}, s'_q)$, simultaneously. This explains why we use 2×2-squares as in Figure 7 and corresponding guards. In the DTA $\mathcal{D}$, they allow us to check when one clock has been reset and the other has not. Now consider the case where p performs a transition of the form $(s_p, a, \overline{guard}(d), \emptyset, s'_p)$. When p executes its transition, at least one clock has reached the value 2. As this clock cannot be reset anymore, q is obliged to follow instantaneously the corresponding transition of the form $(s_q, a, \overline{guard}(d), \emptyset, s'_q)$, to reach a final state. □

We obtain that model checking regular positive specifications is undecidable for DTA:

Corollary 4.2. The questions $L_\forall(\mathcal{D}) \cap R\,?$ and $R \subseteq L_\forall(\mathcal{D})\,?$ are undecidable for given DTA $\mathcal{D}$ and regular language R.

5. Playing with local time rates

In the previous sections, we considered the set of behaviours when the local-time rates are arbitrarily chosen (existential) or arbitrarily enforced (universal). It is then natural to ask what would happen if there are some restrictions on the way these rates are chosen or enforced. In this section, we consider some such questions. Broadly, we examine two types of restrictions on the local-time rates. In the first case, we try to bound the clock drifts between processes, while in the second case, we restrict to a natural sub-class of local-time rate functions.

5.1. Bounding the clock drifts

Our first attempt is to try to bound the way the clocks drift on different processes, thus curtailing the independence of the local-time rates. We consider two sub-cases here, where we insist (1) the ratio or (2) the difference of local times in different processes is always bounded by a constant. We might expect that such a strong restriction could lead to a decidability result for the universal semantics. However, it turns out that we can strengthen the undecidability proof in Section 4 to show that emptiness and universality of the universal semantics is already undecidable in this restricted case.

Let us formalize this. We will restrict to two processes, $Proc = \{p, q\}$. We note however that the following definitions (and results) can easily be generalized to more processes. For a rational number $k \geq 1$, we define $Rates_{\text{rat}}(k) = \{\tau = (\tau_p, \tau_q) \in Rates \mid \frac{1}{k} \leq \frac{\tau_p(t)}{\tau_q(t)} \leq k \text{ for all } t \in \mathbb{R}_{>0}\}$. This is the set of all rate-function tuples such that the ratio of the local times in the two processes are always bounded

Figure 13. Bounding the clock drifts by ratio and difference

by fixed rationals. Thus, when we plot the local times in the two components of the rate-function tuple against each other, this function lies completely within the shaded region in Figure 13 (a).

Further, for a rational number $\ell \geq 0$, $Rates_{\text{diff}}(\ell) = \{\tau = (\tau_p, \tau_q) \in Rates \mid |\tau_p(t) - \tau_q(t)| \leq \ell$ for all $t \in \mathbb{R}_{\geq 0}\}$. These are the rate-function tuples for which the difference between the local times in the two processes are bounded by some constant. Thus again, any function in the shaded region in Figure 13 (b) describes such a bounded rate-function tuple.

Accordingly, for an icTA or a DTA $\mathcal{B}$, we define

- $L_{\exists}^{\text{rat},k}(\mathcal{B}) = \bigcup_{\tau \in Rates_{\text{rat}}(k)} L(\mathcal{B}, \tau)$, $L_{\forall}^{\text{rat},k}(\mathcal{B}) = \bigcap_{\tau \in Rates_{\text{rat}}(k)} L(\mathcal{B}, \tau)$,
- $L_{\exists}^{\text{diff},\ell}(\mathcal{B}) = \bigcup_{\tau \in Rates_{\text{diff}}(\ell)} L(\mathcal{B}, \tau)$, $L_{\forall}^{\text{diff},\ell}(\mathcal{B}) = \bigcap_{\tau \in Rates_{\text{diff}}(\ell)} L(\mathcal{B}, \tau)$.

Theorem 5.1. For icTAs or DTAs $\mathcal{B}$ over $Proc = \{p, q\}$,

1. checking emptiness of $L_{\forall}^{\text{rat},1}(\mathcal{B}) = L_{\forall}^{\text{diff},0}(\mathcal{B})$ is decidable while checking universality is undecidable.
2. checking emptiness and universality of $L_{\forall}^{\text{rat},k}(\mathcal{B})$ are undecidable for every rational $k > 1$.
3. checking emptiness and universality of $L_{\forall}^{\text{diff},\ell}(\mathcal{B})$ are undecidable for every rational $\ell > 0$.

Proof:
For $k = 1$ or $\ell = 0$, the sets $Rates_{\text{rat}}(k)$ and $Rates_{\text{diff}}(\ell)$ consist of exactly the tuples in which time evolves at the same rate in both processes. Thus, the sets $L_{\forall}^{\text{rat},1}(\mathcal{B})$ and $L_{\forall}^{\text{diff},0}(\mathcal{B})$ are identical and correspond to the language of an ordinary timed automaton. Hence, checking emptiness is decidable, while checking universality is undecidable. This proves Part (1) of the theorem.

To prove the remaining parts of the theorem, we need the following lemma.

Lemma 5.1. Let $k > 1$, $\ell > 0$ be some fixed rationals. For all $\sigma \in \{0, 1, 2\}^*$, there exists $\tau \in Rates_{\text{rat}}(k) \cap Rates_{\text{diff}}(\ell)$ such that σ is a prefix of $dir(\tau)$.

Proof:
Let $\sigma = d_1 d_2 \dots d_n \in \{0, 1, 2\}^*$ be of length n. We define τ (in terms of $n + 1$ points) as follows: τ_p is the piecewise linear function with $\tau_p(2i) = x_i$ for $i \in \{0, \dots, n\}$ and $\tau_p(2n + t) = x_n + t$ for all

$t \in \mathbb{R}_{\geq 0}$. Similarly, τ_q is defined as the piecewise linear function with $\tau_q(2i) = y_i$ for $i = \{0, \ldots, n\}$ and $\tau_q(2n+t) = y_n + t$ for $t \in \mathbb{R}_{\geq 0}$. The points (x_i, y_i) are defined by $x_0 = y_0 = 0$ and, for $i \in \{1, \ldots, n\}$, $x_i = 2i - \alpha|d_1 \ldots d_i|_1$ and $y_i = 2i - \alpha|d_1 \ldots d_i|_0$ ($|\sigma'|_d$ denoting the number of occurrences of d in σ'), where α is a rational parameter to be fixed.

With the above definition, we observe that, for all i, we have $|x_i - y_i| \leq i\alpha$, and, for $i > 0$, we have $1 - \frac{\alpha}{2} \leq \frac{x_i}{y_i} \leq \frac{1}{1-\alpha/2}$. Thus, by choosing $\alpha = \min\{\frac{1}{2}, \frac{\ell}{n}, 2(1 - \frac{1}{k})\}$, we can check that $\tau \in Rates_{\text{rat}}(k) \cap Rates_{\text{diff}}(\ell)$.

Finally, we show that $dir(\tau) = \sigma \cdot 2^\omega$. This is done by induction. Assume that $d_1 \cdots d_{i-1} \leq dir(\tau)$ for some $1 \leq i \leq n$. If $d_i = 2$ then $x_i - x_{i-1} = 2 = y_i - y_{i-1}$ and we deduce that $d_1 \cdots d_{i-1} d_i \leq dir(\tau)$. If now $d_i = 0$ then $x_i - x_{i-1} = 2$ and $y_i - y_{i-1} = 2 - \alpha$. Since $0 < \alpha < 1$, we deduce that $d_1 \cdots d_{i-1} d_i \leq dir(\tau)$. The proof is similar when $d_i = 1$. Hence, $\sigma = d_1 \cdots d_n \leq dir(\tau)$. Since after $t = 2n$ clocks x and y are synchronous, we obtain $dir(\tau) = \sigma \cdot 2^\omega$. □

With the above lemma, we now prove Parts (2) and (3) of the theorem. Let $k > 1$ and $\ell > 0$. First, we will show that checking emptiness is undecidable. Given a PCP instance as before, we again consider the icTA (or DTA) $\mathcal{B}$ from Figure 9. We want to show that $w \in A^+$ is solution if and only if $wb \in L_\forall(\mathcal{B}) = L_\forall^{\text{rat},k}(\mathcal{B}) = L_\forall^{\text{diff},\ell}(\mathcal{B})$. One direction is trivial. If, for $w \in A^+$, we have $f(w) = g(w)$, then $wb \in L_\forall(\mathcal{B})$, and this implies that $wb \in L_\forall^{\text{rat},k}(\mathcal{B})$ and $wb \in L_\forall^{\text{diff},\ell}(\mathcal{B})$. On the other hand, if $wb \in L_\forall^{\text{rat},k}(\mathcal{B})$ or $wb \in L_\forall^{\text{diff},\ell}(\mathcal{B})$, then, by Lemma 5.1, we pick $\tau \in Rates_{\text{rat}}(k) \cap Rates_{\text{diff}}(\ell)$ such that $dir(\tau) = f(w) \cdot 2 \cdot 2^\omega$, and the remaining part of the proof follows as before.

Now, to show that the universality is undecidable, we consider icTA $\widetilde{\mathcal{B}}$ from Figure 11 and show that $w \in A^+$ is a solution iff $wb \notin L_\forall^{\text{rat},k}(\widetilde{\mathcal{B}})$, and that w is a solution iff $wb \notin L_\forall^{\text{diff},\ell}(\widetilde{\mathcal{B}})$. One direction is again easy. If $wb \notin L_\forall^{\text{rat},k}(\widetilde{\mathcal{B}})$ or $wb \notin L_\forall^{\text{diff},\ell}(\widetilde{\mathcal{B}})$ then $wb \notin L_\forall(\widetilde{\mathcal{B}})$ and since $wb \in A^+b$, we deduce that $f(w) = g(w)$ as in the proof of Theorem 4.2. On the other hand, assume w is a solution so that we have $f(w) = g(w)$. Again by Lemma 5.1, we pick $\tau \in Rates_{\text{rat}}(k) \cap Rates_{\text{diff}}(\ell)$ such that $dir(\tau) = f(w) \cdot 2 \cdot 2^\omega$ and so by Claim 4.3, $(\widetilde{\mathcal{B}}, \tau) : s_0 \overset{wb}{\not\longrightarrow} r_1$ and $(\widetilde{\mathcal{B}}, \tau) : s_0 \overset{wb}{\not\longrightarrow} r_2$. We also have that after wb, states s_0, u_1,u_2 cannot be reached and so $wb \notin L_\forall^{\text{rat},k}(\widetilde{\mathcal{B}})$ and $wb \notin L_\forall^{\text{diff},\ell}(\widetilde{\mathcal{B}})$.

This completes the proof of the whole theorem.

□

As a related question, we could also ask if the existential semantics still describes a regular set of behaviours, i.e, for each $k \geq 1$ and $\ell \geq 0$, are $L_\exists^{\text{rat},k}(\mathcal{B}), L_\exists^{\text{diff},\ell}(\mathcal{B})$ regular? We leave this as an open question. We note however, that this does not immediately follow from the region construction, since the restriction induced by the bounds may not result in classical zones (union of regions).

5.2. Restricting to fixed slopes

In this subsection, we restrict the behaviour by considering a selected subclass of local-time rate functions, rather than all of them. In particular, we restrict to the class of local-time rate functions that have fixed and constant (rational) slopes. Surprisingly, even checking emptiness of the existential semantics turns out to be undecidable with this restriction.

In fact, we show that checking emptiness of the existential semantics is undecidable even in a slightly weaker setting, where the local-time rate functions have a fixed slope with respect to each other. To see this, let us define $Rates_{\text{fix}} = \{\tau \in Rates \mid$ for each pair $p, q \in Proc$, there is a constant $\alpha_{pq} \in$

$\mathbb{Q}_{\geq 0}$ such that $\tau_p(t) = \alpha_{pq} \cdot \tau_q(t)$ for all $t \in \mathbb{R}_{\geq 0}\}$. Again, for an icTA or DTA $\mathcal{B}$, we define $L^{\text{fix}}_{\exists}(\mathcal{B}) = \bigcup_{\tau \in Rates_{\text{fix}}} L(\mathcal{B}, \tau)$ and $L^{\text{fix}}_{\forall}(\mathcal{B}) = \bigcap_{\tau \in Rates_{\text{fix}}} L(\mathcal{B}, \tau)$. Now, we can state the theorem formally:

Theorem 5.2. For icTAs or DTAs $\mathcal{B}$, checking emptiness of $L^{\text{fix}}_{\exists}(\mathcal{B})$ is undecidable.

The proof is by reducing to the above problem a certain "unknown-sampling rate discrete-time reachability problem" for timed automata [3, 8], which was proved in [8] to be undecidable. The idea is that the fixed constant β between the local-time rates of two processes can be used to simulate a β-sampled run of a timed automaton. To make this clear, we need to define sampled runs and the results known about them.

A timed run $\sigma = (a_1, t_1) \dots (a_n, t_n)$, with $a_i \in \Sigma_\varepsilon, t_i \in \mathbb{R}_{\geq 0}$, of a timed automaton $\mathcal{A} = (S, \Sigma, \mathcal{Z}, \delta, I, i, F)$ can be seen as an alternating sequence of discrete (or ε) moves and time elapse moves (see, e.g., [3] for details). Then, σ is called a *β-sampled run* for some $\beta \in \mathbb{Q}_{\geq 0}$, if each time elapse is exactly β, i.e., $t_i - t_{i-1} = \beta$ for all $i \in \{1, \dots, n\}$, $t_0 = 0$. A state s of $\mathcal{A}$ is reachable in the β-sampled semantics of $\mathcal{A}$, if there is a β-sampled run of $\mathcal{A}$ that reaches it. It is easy to see that this definition coincides with the one in [8] from which we have the following theorem:

Theorem 5.3. (cf. [8])
The following problem is undecidable: Given a timed automaton $\mathcal{A}$ and a state s of $\mathcal{A}$, does there exist $\beta \in \mathbb{Q}_{\geq 0}$ such that s is reachable in the β-sampled semantics of $\mathcal{A}$.

Proof:
[of Theorem 5.2] Given a timed automaton $\mathcal{A}$ and a state s of $\mathcal{A}$, we will construct a DTA $\mathcal{D}$ such that $L^{\text{fix}}_{\exists}(\mathcal{D}) \neq \emptyset$ if and only if there is a β such that s is reachable in the β-sampled semantics of $\mathcal{A}$. Thus, from the undecidability result of the theorem above, it follows that checking emptiness of $L^{\text{fix}}_{\exists}(\mathcal{D})$ is undecidable.

The DTA we construct will consist of two components/processes. The broad idea is that the first component just measures its local time by making a transition at every clock tick. The second component simulates the timed automaton $\mathcal{A}$ and in addition checks a clock of the first component, to ensure that its transitions occur exactly at clock ticks of the first component. Thus, if the relative local-time rate is β, then a run of the automaton in the second component corresponds to a β-sampled run of the timed automaton.

There is a point of subtlety here. The automaton in the second component must make a transition every clock tick and cannot be allowed to remain at a state as time passes. It turns out that straightforward use of state invariants is not enough. Our construction below describes one way to handle these concerns.

Let $\mathcal{A} = (S_\mathcal{A}, \Sigma_\mathcal{A}, \mathcal{Z}_\mathcal{A}, \delta_\mathcal{A}, I_\mathcal{A}, \iota_\mathcal{A}, F_\mathcal{A})$ be the timed automaton and $s_\mathcal{A} \in S_\mathcal{A}$. We set $Proc = \{p, q\}$ and define DTA $\mathcal{D} = ((\mathcal{A}_p, \mathcal{A}_q), \pi)$ as follows.

The p-component uses and owns a single clock x (which occurs in both components, as will be seen). Thus, $\mathcal{Z}_p = \{x\}$ and $\pi(x) = p$. This automaton $\mathcal{A}_p$ is described in Figure 14.

The second component consists of two copies of $\mathcal{A}$ with transitions alternating between them, defined as follows: $\mathcal{A}_q = (S_q, \Sigma_q, \mathcal{Z}_q, \delta_q, I_q, \iota_q, F_q)$, where

- $S_q = S_\mathcal{A} \times \{0, 1\}$,
- $\Sigma_q = \Sigma_\mathcal{A}$,

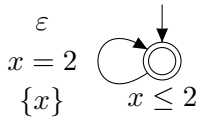


Figure 14. The timed automaton $\mathcal{A}_p$

- $\mathcal{Z}_q = \mathcal{Z}_\mathcal{A} \uplus \{x\}$ and $\pi(y) = q$ for all $y \in \mathcal{Z}_\mathcal{A}$,
- if $(s, a, \varphi, R, s') \in \delta_\mathcal{A}$, then $((s,0), a, \varphi, R, (s',1)), ((s,1), a, \varphi, R, (s',0)) \in \delta_q$,
- $I_q((s,0)) = I_\mathcal{A}(s) \wedge (0 \leq x \leq 1)$ and $I_q((s,1)) = I_\mathcal{A}(s) \wedge (1 \leq x \leq 2)$,
- $\iota_q = (\iota_\mathcal{A}, 0)$, and
- $F_q = \{(s_\mathcal{A}, 0), (s_\mathcal{A}, 1)\}$.

Thus, in the above construction of $\mathcal{A}_q$, in any state of the first copy, the value of clock x must be between 0 and 1. Similarly, in any state of the second copy, the clock has a value between 1 and 2. Further, transitions can only occur from one copy to another (never within a copy). These properties ensure that transitions (of $\mathcal{A}_q$) occur at every clock tick (of clock x) and there must occur a transition at every clock tick (else a state invariant is violated). But clock x measures its local time, i.e, the local time of process p. Thus, if β is the ratio between the fixed local-time rates of the two processes $\tau = (\tau_p, \tau_q) \in \mathit{Rates}_{\text{fix}}$, then each transition of $\mathcal{A}_q$ (which simulates $\mathcal{A}$) occurs at β-intervals, thus simulating a β-sampled run of $\mathcal{A}$. Finally, a run of $\mathcal{D}$ under this local-time rate function τ is accepting, if and only if the corresponding β-sampled run reaches $s_\mathcal{A}$. Thus, the existential fixed-slope semantics of $\mathcal{D}$ is non-empty if and only if there is a β such that $s_\mathcal{A}$ is reachable in the β-sampled semantics. This completes the proof of the theorem. □

We could also consider the emptiness problem for the universal semantics, i.e., is $L^{\text{fix}}_\forall(\mathcal{D}) = \emptyset$? From the above construction, we have $L^{\text{fix}}_\forall(\mathcal{D}) = \emptyset$ if and only if there is a β such that $s_\mathcal{A}$ is *not* reachable in the β-sampled semantics of $\mathcal{A}$. Thus if the safety problem for β-sampled semantics is undecidable (to the best of our knowledge, the status of this problem is open), then this would imply undecidability here as well.

6. The reactive semantics

The goal of this section is to tackle the undecidability results that come along with the universal semantics so that we are still in a position to check a given system against positive specifications. In order to do this, we come up with a non-trivial under-approximation of the system behaviors that is also an under-approximation of (or is contained in the) universal semantics. Thus, if the set of behaviors given by the positive specification is contained in this set, then we are certain that it is satisfied under all possible choices of clock rates. We will show that this set is always regular thus allowing us to effectively check it against regular positive specifications.

We start by taking a closer look at the roots of the undecidability result. To explain the undecidability, it is useful to consider both semantics, the existential one and the universal one, as a game. In the existential semantics, there is one player, controlling both, system transitions and time. Indeed, a behavior is accepted if it is executable under some arbitrary time rates. It is then left to the only player to find transitions and rates that get a given behavior accepted. In the universal semantics, there are two players: Player 0 controls the transitions, and Player 1 controls the environment, i.e., time. Actually, this game has only two rounds, since Player 1 has to provide the entire time rates in advance without knowing the transitions that Player 0 will choose. Given this first "blind move" by Player 1, Player 0 can then choose transitions freely. A behavior is accepted if Player 0 has a winning strategy, i.e., if a final state can be reached no matter what the initial move by Player 1 was. Note that, in game frameworks, the lack of information of one player is often the reason for undecidability (see, e.g., [15, 16]). Based on this observation, we introduce a third semantics: The *reactive semantics* considers our system model to be a (more) turn-based, perfect-information two-player game. It allows Player 1 to react upon the moves made by Player 0. As, then, Player 1 is granted more power, the reactive semantics contains fewer behaviors than the universal semantics. Therefore, it is an under-approximation of the system behavior, too. However, we will show that, thanks to perfect information, it is always regular and, thus, a suitable tool to check a system against positive specifications.

Let us illustrate the reactive semantics through an example. Consider the icTA $\mathcal{B}$ from Figure 3. We have that ab is in the universal language since Player 0 has a winning strategy for ab: For any time rates (any move by Player 1), we can find an accepting run. In such an accepting run on ab, we start at s_0 and we can move to either s_1 or s_2. This choice of which state to move to now (i.e., at s_0 with clock values $x < 1, y < 1$) depends on how the local time rates behave in the future, an information that is given to Player 0 in advance. In particular, if a region will be reached with $x = 1$ and $0 < y \leq 1$, then we move to s_1 so that we can safely execute b. If, on the other hand, we know that the region with $0 < x < 1$ and $y = 1$ will be reached, then we go to s_2. Thus, the universal semantics works by assuming that Player 1 (the environment, or, time) provides the whole future time rate function at any point without knowing anything about Player 0 (blind move). In a turn-based, perfect-information game, on the other hand, Player 1 reveals the time evolution only piecewise and can adapt its strategy according to the system moves. This can be seen pictorially from Figure 15. In the case of ab, the environment simply waits for the first transition chosen by Player 0 and then chooses a time evolution that spoils the goal to reach a final state while reading ab: Initially, Player 1 chooses the region $0 < x, y < 1$ and gives control to Player 0. If Player 0 goes to s_1, Player 1 moves to the time region with $0 < x < 1$ and $y = 1$. Otherwise, it may move to $x = y = 1$. In both cases, b cannot be executed anymore. Thus, ab is not in the reactive semantics. Indeed, we will show that the reactive semantics is always contained in the universal one.

An advantage of the reactive semantics is that it can be refined in terms of a bounded look-ahead, which still yields a regular language: We can grant Player 0 more power by requiring that Player 1 has to reveal a larger (but bounded) amount of information of his future moves.

There is another subtle but powerful motivation for looking at the reactive semantics. To check if the system satisfies a positive specification, we would like to design a controller which does this check. For this, the universal semantics is inadequate, in the sense that, to choose a correct run in the system, we might need to know the future time rates. For instance, in the icTA from Example 2.2 (Figure 3), ab is in its universal language. However, as seen from Figure 15, to accept ab, the automaton, at state s_0, must guess how the time rates will evolve after this step and accordingly go to either s_1 or s_2. If we are designing a controller, then we do not know how the time rates will change later. Thus, we do not

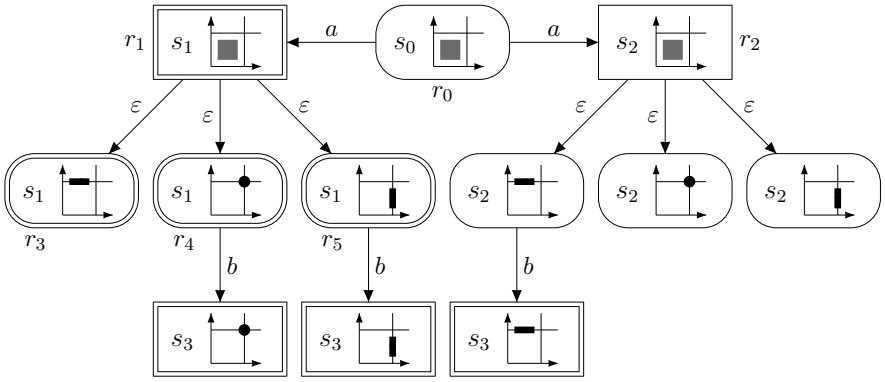


Figure 15. Part of the region/alternating automaton for the icTA from Figure 3

know a priori whether we will reach this region or the other, and so the controller actually has no way of deciding which transition to take, to ensure an accepting run. Thus, in Figure 15, though ab is in the universal language, there is no step-by-step, "practically constructible" accepting run on ab. The reactive semantics disallows such words and hence can be potentially used for controller synthesis.

Formally, we will describe the reactive semantics using an alternating automaton, which is based on the region automaton introduced in Section 3. Intuitively, *time-elapse* transitions are controlled by the environment (Player 1), and *discrete* transitions are controlled by the system (Player 0), which aims at exhibiting some behavior. Actually, this *game* is not turn-based in a strict sense because the system should be able to execute several discrete transitions while staying in the same region. After moving from some region to a successor region, the environment hands over the control to the system so that the system always has a chance to execute some discrete transition. On the other hand, after executing some discrete transition, the system may either keep the control or hand it over to the environment.

Since icTAs or DTAs have ε-transitions, we define an *alternating automaton with ε-transitions* (ε-AA) as a tuple $\mathcal{A} = (S, \Sigma, \delta, \iota, F)$ where S is a finite set of *states*, $\iota \in S$ is the *initial state*, $F \subseteq S$ is the set of *final states*, and $\delta : S \times \Sigma_\varepsilon \to \mathbb{B}^+(S)$ is the *alternating transition function*. Here, $\mathbb{B}^+(S)$ denotes positive boolean combinations of states from S.

As usual, a run of an ε-AA will be a (doubly) labeled finite tree. We assume the reader to be familiar with the notion of trees and only mention that we deal with structures (V, σ, μ) where V is the finite set of nodes with a distinguished root, and both σ and μ are node-labeling functions. Given a node $u \in V$, the set of children of u is denoted $\mathrm{children}(u)$. Let $w = a_1 \dots a_{|w|} \in \Sigma^*$ be a finite word. A run of $\mathcal{A}$ on w is a doubly labeled finite tree $\rho = (V, \sigma, \mu)$ where $\sigma : V \to S$ is the *state-labeling* function and $\mu : V \to \{0, \dots, |w|\}$ is the *position-labeling* function such that, for each node $u \in V$, the following hold:

- if u is the root, then $\sigma(u) = \iota$ and $\mu(u) = 0$ (we start in the initial state at the beginning of the word),
- if u is not a leaf (i.e., $\mathrm{children}(u) \neq \emptyset$), then we have

- either $\mu(u') = \mu(u)$ for all $u' \in \text{children}(u)$ and in this case $\{\sigma(u') \mid u' \in \text{children}(u)\} \models \delta(\sigma(u), \varepsilon)$
- or $\mu(u') = \mu(u) + 1 = i \leq n$ for all $u' \in \text{children}(u)$ and in this case $\{\sigma(u') \mid u' \in \text{children}(u)\} \models \delta(\sigma(u), a_i)$.

The run is accepting if all leaves are labeled with $F \times \{|w|\}$. The set of words from Σ^* that come with an accepting run is denoted by $L(\mathcal{A})$. This set is indeed regular since ε-AAs are special cases of two-way alternating automata (2-AA) which accept only regular languages.

Lemma 6.1. (cf. [5])
Given a 2-AA $\mathcal{A}$ with n states, one can construct a non-deterministic finite automaton with $2^{\mathcal{O}(n^2)}$ states that recognizes $L(\mathcal{A})$.

Let $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ be an icTA. We associate with $\mathcal{B}$ an ε-AA $\mathcal{A}_\mathcal{B} = (S', \Sigma, \delta', \iota', F')$ as follows: First, let $S' = S \times \mathit{Regions}(\mathcal{B}) \times \{0, 1\}$. Intuitively, tag 0 is for *system positions* (Player 0) while tag 1 is for *environment positions* (Player 1). Recall that the environment controls how time elapses whereas the system wants to accept some word. Then, $\iota' = (\iota, [\nu], 0)$ where $\nu(x) = 0$ for each $x \in \mathcal{Z}$, and $F' = F \times \mathit{Regions}(\mathcal{B}) \times \{0, 1\}$. Finally, for $(s, \gamma) \in S \times \mathit{Regions}(\mathcal{B})$ and $a \in \Sigma_\varepsilon$, we let

$$\delta'((s,\gamma,1),a) \;=\; \begin{cases} \text{False} & \text{if } a \neq \varepsilon \text{ or } \gamma \text{ maximal} \\ \bigwedge\{(s,\gamma',0) \mid \gamma \prec \gamma'\} & \text{otherwise} \end{cases}$$

$$\delta'((s,\gamma,0),a) \;=\; \begin{cases} \bigvee\{(s',\gamma',0) \mid (s,\gamma) \xrightarrow{a}_{\mathrm{d}} (s',\gamma')\} & \text{if } a \neq \varepsilon \text{ or } \gamma \text{ maximal} \\ (s,\gamma,1) \vee \bigvee\{(s',\gamma',0) \mid (s,\gamma) \xrightarrow{\varepsilon}_{\mathrm{d}} (s',\gamma')\} & \text{otherwise} \end{cases}$$

where $\xrightarrow{a|\varepsilon}_{\mathrm{d}}$ denotes a discrete transition of the region automaton $\mathcal{R}_\mathcal{B}$ (Section 3).

Definition 6.1. For an icTA $\mathcal{B}$, let $L_{react}(\mathcal{B}) = L(\mathcal{A}_\mathcal{B})$ be the *reactive semantics* of $\mathcal{B}$. Moreover, for a DTA $\mathcal{D}$, $L_{react}(\mathcal{D}) = L_{react}(\mathcal{B}_\mathcal{D})$ is the *reactive semantics* of $\mathcal{D}$.

Example 6.1. Consider, again, the icTA $\mathcal{B}$ from Figure 3. A part of its ε-AA $\mathcal{A}_\mathcal{B}$ is shown in Figure 15. States with tag 0 are depicted as ovals and are existential (non-deterministic) states and states with tag 1 are depicted as rectangles and are universal states. We have, e.g., $\delta'(r_1, \varepsilon) = r_3 \wedge r_4 \wedge r_5$. Note, however, that a transition from an oval to a rectangles should actually be split into two transitions, which is omitted in the picture. For example, there is a state r_1' between r_0 and r_1 which resembles r_1 but is tagged 0. Similarly, there is another state r_2' between r_0 and r_2, and we have $\delta'(r_0, a) = r_1' \vee r_2'$. Then, as mentioned in the beginning of the section, under the reactive semantics, the language of this automaton contains a but does not contain ab. Thus, $L_{react}(\mathcal{B}) = \{a\}$.

The following theorem follows from Lemma 6.1:

Theorem 6.1. Let $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ be an icTA and let n be the number of states of $\mathcal{R}_\mathcal{B}$ (which is bounded by $|S| \cdot (2\,C + 2)^{|\mathcal{Z}|} \cdot |\mathcal{Z}|!$ where C is the largest constant a clock is compared with in $\mathcal{B}$). Then, $L_{react}(\mathcal{B})$ is regular and one can compute a non-deterministic finite automaton with $2^{\mathcal{O}(n^2)}$ states that recognizes $L_{react}(\mathcal{B})$.

As expected, the reactive semantics is more restricted than the universal semantics, so we get the inclusion of Proposition 6.1. It can also be considered as an underapproximation of the universal semantics. Therefore, we can safely use the reactive semantics to check an icTA for a positive specification $Good$ containing the behaviors that a system *must* exhibit. If $Good \subseteq L_{react}(\mathcal{B})$ then the system has a strategy to ensure each *good* behaviors robustly against all possible clock variations.

Proposition 6.1. For any icTA $\mathcal{B}$, $L_{react}(\mathcal{B}) \subseteq L_{\forall}(\mathcal{B})$.

Proof:
Assume $\mathcal{B} = (S, \Sigma, \mathcal{Z}, \delta, I, \iota, F, \pi)$ to be an icTA and let $\mathcal{A}_{\mathcal{B}} = (S', \Sigma, \delta', \iota', F')$ be the associated ε-AA, and let $w \in L_{react}(\mathcal{B}) = L(\mathcal{A}_{\mathcal{B}})$. Let furthermore $\rho = (V, \sigma, \mu)$ be an accepting run of $\mathcal{A}_{\mathcal{B}}$ on w. We pick $\tau \in Rates$.

We construct inductively a maximal branch $u_0 u_1 \dots u_n \in V^*$ in ρ and two sequences $t_0, t_1, \dots, t_n$ and $\nu_0, \nu_1, \dots, \nu_n$ as follows. First, we let $u_0 = \varepsilon$, $t_0 = 0$ and $\nu_0(x) = 0$ for all $x \in \mathcal{Z}$. Note that $\sigma(u_0) = (\iota, [\nu_0], 0)$. Assume that the sequences have been constructed up to k and that $\sigma(u_k) = (s_k, [\nu_k], pl_k)$. If u_k is a leaf, the construction is over and $n = k$. Otherwise, there are three cases. First, assume that $pl_k = 1$. Let $t_{k+1} > t_k$ be such that $[\nu_k] \prec [\nu_{k+1}]$ with $\nu_{k+1} = \nu_k + \tau(t_{k+1}) - \tau(t_k)$. By definition of δ', there exists a child u_{k+1} of u_k such that $\sigma(u_{k+1}) = (s_k, [\nu_{k+1}], 0)$. Assume now that $pl_k = 0$. Choose u_{k+1} in $\mathrm{children}(u_k)$. Either $\sigma(u_{k+1}) = (s_k, [\nu_k], 1)$ and we let $t_{k+1} = t_k$ and $\nu_{k+1} = \nu_k$ in this second case. Otherwise, the move from u_k to u_{k+1} corresponds to some discrete transition of $\mathcal{R}_{\mathcal{B}}$ with label $a_{k+1} \in \Sigma_\varepsilon$ and some reset set $R \subseteq \mathcal{Z}$. In this third case, we let $t_{k+1} = t_k$ and $\nu_{k+1} = \nu_k[R]$ so that we have $\sigma(u_{k+1}) = (s_{k+1}, [\nu_{k+1}], 0)$.

The discrete moves along the constructed branch correspond to the sequence $0 < i_1 < \dots < i_\ell \leq n$ of indices k such that $pl_{k-1} = pl_k = 0$. As ρ is an accepting run for w, we have $w = a_{i_1} \cdot a_{i_2} \cdot \dots \cdot a_{i_\ell}$ and $s_{i_\ell} = s_n \in F$. It is now easy to verify that the sequence

$$(s_0, \nu_0) \xrightarrow{a_{i_1}, t_{i_1}} (s_{i_1}, \nu_{i_1}) \xrightarrow{a_{i_2}, t_{i_2}} \dots \xrightarrow{a_{i_\ell}, t_{i_\ell}} (s_{i_\ell}, \nu_{i_\ell})$$

is a τ-run of $\mathcal{B}$ so that $w \in L(\mathcal{B}, \tau)$. □

To summarize, we have the following strict hierarchy of our semantics.

Proposition 6.2. Suppose that $|Proc| \geq 2$. There are some DTA $\mathcal{D}$ over $Proc$ and some $\tau \in Rates$ such that $L_{react}(\mathcal{D}) \subsetneq L_{\forall}(\mathcal{D}) \subsetneq L(\mathcal{D}, \tau) \subsetneq L_{\exists}(\mathcal{D})$.

Proof:
Consider the icTA $\mathcal{B}$ from Figure 3. Recall that $L_{react}(\mathcal{B}) = \{a\}$, $L_{\forall}(\mathcal{B}) = \{a, ab\}$, $L(\mathcal{B}, \mathrm{id}) = \{a, ab, b\}$, and $L_{\exists}(\mathcal{B}) = \{a, ab, b, c\}$. As $\mathcal{B}$ does not employ any reset, we may view it as a DTA where $\mathcal{B}$ models a process owning clock x, and where a second process, owning clock y, does nothing, but is in a local accepting state. □

7. Conclusion

We provided a framework for the analysis of distributed timed systems where each clock belongs to a local process and may evolve independently of clocks of other processes. The analysis is based on a regular under- or over-approximation of the system behavior, depending on the property at hand.

As future work, it remains to investigate the expressive power of DTAs and, in particular, the *synthesis problem*: Given a (global) specification $Spec$, can we generate a DTA D (over a given system architecture) such that $L_{react}(\mathcal{D}) = Spec$? Here, the system architecture describes the clock accessibility relation between processes by specifying constraints like "process p can only read clocks of process q." A similar synthesis problem has been studied in [11] in the framework of untimed distributed channel systems. There, additional messages are employed to achieve a given global behavior. In this context, it would be favorable to have partial-order based specification languages and a partial-order semantics for DTAs (see, for example, [14]). It would also be worthwhile to study to which extent partial-order methods [4, 14] can be applied to make our model-checking approach more efficient.

Acknowledgments

We thank the anonymous referees for their useful remarks and interesting comments on this paper.

References

[1] Akshay, S., Bollig, B., Gastin, P., Mukund, M., Narayan Kumar, K.: Distributed Timed Automata with Independently Evolving Clocks, *Proc. of CONCUR*, 5201, Springer, 2008.

[2] Alur, R., Dill, D. L.: A Theory of Timed Automata., *Theoretical Computer Science*, **126**(2), 1994, 183–235.

[3] Alur, R., Madhusudan, P.: Decision Problems for Timed Automata: A Survey, *Formal Methods for the Design of Real-Time Systems*, Springer, 2004.

[4] Bengtsson, J., Jonsson, B., Lilius, J., Yi, W.: Partial Order Reductions for Timed Systems, *Proc. of CONCUR*, 1466, Springer, 1998.

[5] Birget, J.-C.: State-complexity of finite-state devices, state compressibility and incompressibility, *Mathematical Systems Theory*, **26**(3), 1993, 237–269.

[6] Bouyer, P., Haddad, S., Reynier, P.-A.: Timed Unfoldings for Networks of Timed Automata, *Proc. of ATVA*, 4218, Springer, 2006.

[7] Cassez, F., Chatain, T., Jard, C.: Symbolic Unfoldings For Networks of Timed Automata, *Proc. of ATVA*, 4218, Springer, 2006.

[8] Cassez, F., Henzinger, T. A., Raskin, J.-F.: A Comparison of Control Problems for Timed and Hybrid Systems, *Proc. of HSCC*, 2289, Springer, 2002.

[9] De Wulf, M., Doyen, L., Markey, N., Raskin, J.-F.: Robustness and Implementability of Timed Automata, *Proc. of FORMATS and FTRTFT*, 3253, Springer, 2004.

[10] Dima, C., Lanotte, R.: Distributed Time-Asynchronous Automata, *Proc. of ICTAC*, 4711, Springer, 2007.

[11] Genest, B.: On Implementation of Global Concurrent Systems with Local Asynchronous Controllers., *Proc. of CONCUR*, 3653, Springer, 2005.

[12] Henzinger, T. A.: The theory of hybrid automata, *Proc. of LICS*, IEEE Computer Society, 1996.

[13] Larsen, K. G., Pettersson, P., Yi, W.: Compositional and symbolic model-checking of real-time systems, *Proc. of RTSS*, IEEE Computer Society, 1995.

[14] Lugiez, D., Niebert, P., Zennou, S.: A partial order semantics approach to the clock explosion problem of timed automata, *Theoretical Computer Science*, **345**(1), 2005, 27–59.

[15] Peterson, G., Reif, J.: Multiple-Person Alternation, *Proc. of FOCS*, 1979.

[16] Peterson, G., Reif, J., Azhar, S.: Lower Bounds for Multiplayer Noncooperative Games of Incomplete Information, *Comput. Math. Appl.*, **41**, 2001, 957–992.

[17] Puri, A.: Dynamical Properties of Timed Automata, *Discrete Event Dynamic Systems*, **10**(1-2), 2000, 87–113, ISSN 0924-6703.

[18] Swaminathan, M., Fränzle, M., Katoen, J.-P.: The surprising robustness of (closed) timed automata against clock-drift, *Proc. of IFIP-TCS*, 273, Springer, 2008.

[19] Zielonka, W.: Notes on finite asynchronous automata, *R.A.I.R.O. — Informatique Théorique et Applications*, **21**, 1987, 99–135.