Theoretical Computer Science 625 (2016) 85–124

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Timed recursive state machines: Expressiveness and complexity ☆

Massimo Benerecetti *, Adriano Peron

Department of Electrical Engineering and Information Technologies, Università di Napoli "Federico II", Italy

ARTICLE INFO

Article history:
Received 5 August 2014
Received in revised form 7 January 2016
Accepted 15 February 2016
Communicated by P. Aziz Abdulla

Keywords:
Formal languages
Formal models of computing
Timed automata
Real-time systems
Pushdown systems
Recursive state machines

ABSTRACT

The paper proposes a temporal extension of Recursive State Machines (RSMs), called Timed RSMs (TRSMs), which consists of an indexed collection of Timed Automata, called components. Each component can invoke other components in a potentially recursive manner. Besides being able to model procedure calls and recursion, TRSMs are equipped with the ability to suspend the evolution of time within a component when another component is invoked and to recover it when control is given back at return time. This mechanism is realized by storing clock valuations into an implicit stack at invocation time and restoring them upon return. Indeed, TRSMs can be related to an extension of Pushdown Timed Automata, called EPTAs, where an additional stack, coupled with the standard control stack, is used to store temporal valuations of clocks. The expressiveness and computational properties of the resulting model are analyzed, showing that it can be used to recognize timed languages exhibiting context-free properties not only in the untimed "control" part, but also in the associated temporal dimension. The reachability problem for both TRSMs and EPTAs is investigated, showing that the problem is undecidable in the general case. However, the problem becomes decidable for two meaningful subclasses, called I-TRSM and L-TRSM, obtained by suitably constraining the set of clocks to reset at invocation time and to restore at return time. The considered subclasses are compared with the corresponding EPTAs subclasses through bisimulation of their timed LTSs. The complexity of the reachability problem for L-TRSM and I-TRSM is proved to be **EXPTIME**-complete and **PSPACE**-complete, respectively. Moreover, we prove that such classes strictly enhance the expressive power of Timed Automata and of Pushdown Timed Automata, forming a proper hierarchy.

1. Introduction

The formalism of *Recursive State Machines* (RSMs) has been introduced in [5] to model control flow in typical sequential imperative programming languages with recursive procedure calls. A RSM is an indexed collection of finite state machines, called components. Components enhance the power of ordinary state machines with special nodes, called *boxes*, which correspond to invocations of other components in a potentially recursive manner. As shown in [5], RSMs are closely related to

☆ Partially supported by Italian MIUR Project "Integrating automated reasoning in model checking: towards push-button formal verification of large-scale and infinite-state systems", PRIN-20079E5KM8.

* Corresponding author.
E-mail addresses: massimo.benerecetti@unina.it (M. Benerecetti), adrperon@unina.it (A. Peron).

http://dx.doi.org/10.1016/j.tcs.2016.02.021
0304-3975/

Pushdown Systems (PDSs). While PDSs have been widely studied in the literature within the field of program verification, RSMs seem to be more appropriate for visual modeling. In the context of state-transition formalisms, Timed Automata [2], TAs for short, have received a lot of attention in the past twenty years and have become the reference framework for modeling real-time systems. Their good computational properties, deriving from the decidability of emptiness and the reachability problems [2], have also led to the development of automated tools for the analysis of real-time systems [23,18]. In this paper we consider a real-time extension of RSMs, called *Timed RSMs* (TRSMs for short), which allows to model real-time recursive systems. Roughly speaking, a TRSM is an indexed collection of Timed Automata, referred to as *timed components*, with the additional feature of providing boxes corresponding to invocations of other timed components. All the timed components can refer to a common set of clocks used to constrain their behavior. Within a timed component, the only explicit update of clocks is the standard operation of *clock reset*. In addition, clock updates occur when invocations of other components or returns from components are performed. In particular, at invocation time one can choose to reset clocks, and at return time some clocks may be restored to the values they had at invocation time.

The idea of extending formalisms, which implicitly allow to model recursive systems (e.g., PDSs), with real-time features has already been proposed in the literature (e.g., [10–12,16,15]). In particular, [10,15,16] introduce and study Pushdown Timed Automata (PTAs), namely Timed Automata augmented with a pushdown control stack. The work in [10,11] considers the reachability problem, namely reachability of a given control location, in PTAs (or, equivalently, Context-Free Timed Systems), reducing it to the reachability problem in standard PDSs. The work in [15,16], on the other hand, considers the more general problem of binary reachability in PTAs, namely reachability between configurations, showing that the problem is decidable as well.

Notice, however, that besides that fact that TRSMs, similarly to RSMs, are more appropriate than PTAs for visual modeling, PTAs are not expressive enough to account for storing and restoring clock values. Indeed, the control stack of PTAs can trace the history of component invocation, but cannot be exploited to record the history of clock values stored at invocation time and needed at time of the matching return for restoration. In particular, the clock store/restore mechanism allows to model in a very natural way a notion of time spent by the local computation of a timed component. In other words, in TRSMs one can predicate on the evolution of time within a single component, abstracting away the time elapsed within the invoked components. Since the number of recursive invocation can be unbounded, this notion of local time does not seem amenable to modeling by means of PTAs without employing an unbounded number of clocks.

The correspondence between RSMs and PDSs, as introduced in [5], can be lifted to the timed setting. Since, however, PTAs are not expressive enough to account for TRSMs, we propose Extended PTAs (EPTAs), which augment PTAs with an additional stack used to store clock valuations. The two stacks are independent, in the sense that the control stack is used to save control symbols and the valuation stack is used only for storing clock valuations. As a consequence, EPTAs, besides the standard clock reset operations, also allows for Store and Restore operations on clock valuations. We shall prove that TRSMs are equivalent (via timed bisimulation) to a syntactic restriction of EPTAs, where the operations on the two stacks are synchronized. EPTAs offer a suitable framework to study timed languages exhibiting context-free properties both in the untimed "control" part and in the associated "timestamps". For instance, we provide an example of a context-free timed language with mirror distribution of symbols and of temporal delays between consecutive symbols.

In the paper we identify and study meaningful subclasses of TRSMs and corresponding (via bisimulation) subclasses of EPTAs. The first subclass, called Local TRSM (L-TRSM), restricts TRSMs so as to allow for a local use of clocks. This is obtained by requiring that all the clocks get restored when returning from a component invocation. Therefore, the temporal information may flow from the calling environment to the invoked component but cannot flow in the other direction from the callee back to the caller. The smaller class, called Initialized TRSM (I-TRSM), completely isolates the temporal context of an invoked component, by further requiring that all the clocks are reset (initialized) at each invocation. In a sense, the temporal dimension of each called component of a I-TRSM proceeds independently from the temporal evolution of the calling environment, since the temporal information in I-TRSM neither flows from the calling environment to the invoked component nor from the callee back to the caller.

The expressiveness, decidability and complexity results are the main technical contributions of the paper. A complete comparison of the expressive power (as acceptors of timed languages) of these classes is presented. Decidability and complexity results for the reachability problem are provided for all the classes as well. In particular, we show that, from the expressiveness viewpoint TAs, I-TRSM, L-TRSM and TRSMs form a strictly increasing hierarchy. Unfortunately, we show that I-TRSM and L-TRSM are not closed under intersection, even when intersected with TAs. From the reachability viewpoint, we show that the problem is undecidable for the general classes of TRSMs and EPTAs. However, decidability can be recovered for their subclasses I-TRSM, L-TRSM (and, correspondingly, for I-EPTA and L-EPTA), for which the complexity of the problem is established. In particular, we prove that, despite the fact that the class I-TRSM (and I-EPTA) is more expressive than TAs, the complexity of the reachability problem remains unchanged, namely **PSPACE**-complete. The reachability problem for L-TRSM (and L-EPTA) turns out to be **EXPTIME**-complete.

Related work The relation with the basic traditional formalisms of Pushdown Systems, Recursive State Machines [5], Pushdown Timed Automata [15] has been already outlined in the initial part of the introduction. Even if a precise comparison goes beyond the goals of the paper, a qualitative relation with the formalism of Stopwatch Automata (SWAs for short) [14] is also outlined. The distinguishing feature of SWAs is the ability to freeze/unfreeze the temporal progress of clocks. The invocation/return mechanism introduced in TRSMs, together with the ability of freely choosing the set of clocks to restore upon

return, can simulate the basic mechanism of a SWA. In fact, both SWAs and the general class of TRSMs can simulate, by means of suitable operations on clocks, the increment and decrement operations of 2-Counter Machines [21], which suffices to make the reachability problem undecidable in both formalisms. However, SWAs, as opposed to TRSMs, do not seem to be able to express context-free properties both in the timed and untimed component of the language. Conversely, SWAs can be simulated in the more general class of TRSMs as discussed at the end of Section 5. Analogous considerations also apply to the similarly defined formalism of Suspension Automata [20]. Interrupt Automata [9], another formalism related to SWA, allow for freezing and unfreezing operations on clocks. They are strictly less expressive that SWA and are incomparable with Timed Automata, being able to capture a fragment of the regular timed languages together with some non-regular ones. In this formalism, decidability of the reachability problem is obtained by requiring that each automaton only use one evolving clock at a time, while all the others remain frozen.

A preliminary version of this work has been published in [6], where the basic ideas of TRSMs and EPTAs and their subclasses were introduced. The present work significantly revises the semantics of the formalisms, provides new expressiveness results as well as the results on the complexity for the reachability problem for the subclasses of TRSMs and EPTAs.

A model similar to [6] has been independently formulated in [22] and published almost simultaneously. Recursive Timed Automata are syntactically similar to TRSMs, being collections of Timed Automata extended with special states that correspond to (potentially recursive) invocations of other Timed Automata. When an automaton is invoked, the clock values can be passed to the invoked automaton using either a pass-by-value or a pass-by-reference mechanism. These mechanisms can be easily simulated by resetting/restoring clocks at invocation/return time in TRSMs. In particular, a pass by value (resp. a call by reference) of a clock can be simulated in the general class of TRSMs by a call without reset at invocation time and a restore at return time (resp. without a restore at invocation time). Conversely, Recursive Timed Automata can easily simulate the restoring mechanism of TRSMs but, in absence of silent moves, they seems not to be able to simulate the reset of a subset of clocks possibly performed at invocation time. Therefore, it seems that behaviors of TRSMs where a clock is reset at invocation time and restored at return time cannot be directly simulated by Recursive Timed Automata. Since this kind of behavior is admissible in both the proper considered subclasses of TRSMs (i.e. L-TRSM and I-TRSM) a precise correspondence between the two formalisms or their subclasses cannot be immediately assessed. For this formalism, [22] also provides complexity results similar to the ones given in [6] for TRSMs.

Recently, a novel timed pushdown model, namely Dense Timed Pushdown Automata (DTPDAs), has been proposed in [1]. In a DTPDA each symbol in the stack is equipped with local clocks named *ages*. All ages in the stack are kept aligned with the progress of time. Indeed, differently from TRSMs, local clocks in the stack are updated according to the elapsing of time. The reachability problem of DTPDAs is shown to be in **EXPTIME**.

In [19] another timed pushdown model, called Nested Timed Automata (NeTAs), is introduced which is strictly related to DTPDAs. A NeTA is a pushdown system whose stack symbols are Timed Automata. The automaton behaves as the Timed Automaton on top of the stack. It can switch from one TA to another by pushing, popping, or changing the TA on top of the stack. Differently from TRSMs, as time passes all the clocks of the TAs stored in the stack progress uniformly. As shown in [19], NeTAs can be encoded into DTPDAs, preserving control state reachability.

Structure of the paper The syntax and the semantics of TRSMs are presented in Section 2, where the two subclasses I-TRSM and L-TRSM are also introduced. Section 3 defines the pushdown automata analogue of TRSMs, namely EPTAs, and the corresponding subclasses I-EPTA and L-EPTA. The behavioral correspondence between TRSMs and EPTAs (and their subclasses) is established in Section 4, thus providing the expressive equivalence between the two formalisms. The expressiveness properties of the identified classes of EPTAs is considered in Section 5, where EPTAs are also compared with Timed Automata and PTAs. In Section 6 the reachability problem for EPTAs and TRSMs is considered. The problem is proved to be undecidable for the general class, while decidability is assessed for the weaker subclasses and the precise complexity is provided. Some conclusions are finally drawn in Section 7.

2. Timed Recursive State Machines

A TRSM is an indexed collection of Timed Automata, with the additional ability of distinguishing ordinary states (nodes) and states corresponding to invocation of other timed components (boxes). We shall first recall some standard notions of Timed Automata.

A dense *clock* is a variable over a dense domain $\mathcal{D}^{\geq 0}$ (e.g., non-negative reals $\mathbb{R}^{\geq 0}$ or rationals $\mathbb{Q}^{\geq 0}$). A *clock valuation* is a function $\mathbf{v} : X \to \mathcal{D}^{\geq 0}$, assigning non-negative values to clocks. For a set of clocks X, the set of *clock constraints* $\mathcal{C}(X)$ is defined by the following grammar:

$$\varphi := x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi$$

where $x, y \in X$, $c \in \mathbb{Q}^{\geq 0}$ and $\sim \in \{<, \leq, =, \neq, \geq, >\}$.

Following [2], we write $\mathbf{v} \in \varphi$ when the clock valuation $\mathbf{v}$ satisfies the clock constraint $\varphi \in \mathcal{C}(X)$. The notion of constraint satisfaction by a clock valuation is defined in the natural way:

$$\begin{aligned} \mathbf{v} \in x \sim c \quad &\text{if} \quad \mathbf{v}(x) \sim c \\ \mathbf{v} \in x - y \sim c \quad &\text{if} \quad \mathbf{v}(x) - \mathbf{v}(y) \sim c \\ \mathbf{v} \in \varphi_1 \wedge \varphi_2 \quad &\text{if} \quad \mathbf{v} \in \varphi_1 \text{ and } \mathbf{v} \in \varphi_2. \end{aligned}$$

For $t \in \mathcal{D}^{\geq 0}$, $\mathbf{v}+t$ denotes the valuation $\mathbf{v}'$ such that $\mathbf{v}'(x) = \mathbf{v}(x)+t$, for all $x \in X$ (*clock progress*). Given $r \subseteq X$, $\mathbf{v}\downarrow_r$ denotes the valuation resulting from the reset of the clocks in r, namely:

$$(\mathbf{v}\downarrow_r)(x) = \begin{cases} 0 & \text{if } x \in r \\ \mathbf{v}(x) & \text{otherwise.} \end{cases}$$

Moreover, given the valuations $\mathbf{v}, \overline{\mathbf{v}}$ and a set $r \subseteq X$, $\mathbf{v}\uparrow_{(r,\overline{\mathbf{v}})}$ denotes the valuation resulting from the update of the clocks in r, according to $\overline{\mathbf{v}}$, namely:

$$(\mathbf{v}\uparrow_{(r,\overline{\mathbf{v}})})(x) = \begin{cases} \overline{\mathbf{v}}(x) & \text{if } x \in r \\ \mathbf{v}(x) & \text{otherwise.} \end{cases}$$

Note that the reset operation is a special case of the restore operation. In fact, it holds that $\mathbf{v}\downarrow_r = \mathbf{v}\uparrow_{(r,\mathbf{v}_0)}$, where $\mathbf{v}_0$ is the valuation assigning value 0 to all the clocks.

We can now define Timed Recursive State Machines (TRSMs). A TRSM is an indexed collection of components which share a set of clocks X. The set of states of each component is partitioned into a set of nodes and a set of boxes. Boxes correspond to invocation and are associated with the index of some component. There are three kinds of transitions: (i) *internal transitions* connecting nodes of the same component; (ii) *call transitions* which leads to an entry node of a box, corresponding to the invoked component; (iii) *Return transitions* which lead to exit nodes of a component and trigger the return to the callee. Transitions are decorated with clock constraints as in TAs. Internal transitions can only reset clocks as in TAs. Call transitions can reset a subset of clocks at invocation time. Return transitions can update a subset of clocks by restoring the values they had at invocation time, and reset a, possibly different, subset of clocks. Transitions of a TRSM are triggered by external stimuli provided by the environment (*external symbols*). As usual, we shall denote with Σ the alphabet of external symbols.

Definition 1. Given an alphabet Σ, a TRSM over Σ is a tuple $\langle A_1, \ldots, A_n, X\rangle$ where, for all $1 \leq i \leq n$, A_i is a *component* and X is a finite set of clocks. A component A_i is a tuple $\langle N_i \bigcup B_i, Y_i, En_i, Ex_i, \delta_i\rangle$, where:

- N_i and B_i are the disjoint sets of *nodes* and *boxes*, respectively, and $B_i \cap B_j = \emptyset$, for $i \neq j$;
- $Y_i : B_i \to \{1, \ldots, n\}$ assigns to every box the index of a component (the invoked component);
- $En_i \subseteq N_i$ and $Ex_i \subseteq N_i$ are the sets of *entry* and *exit* nodes, respectively;
- $\delta_i \subseteq (N_i \cup Retns_i) \times \Sigma \times C(X) \times C(X) \times 2^X \times 2^X \times (N_i \cup Calls_i)$ is the transition relation, where $Calls_i = \{(b, en) : b \in B_i \text{ and } en \in En_{Y_i(b)}\}$, $Retns_i = \{(b, ex) : b \in B_i \text{ and } ex \in Ex_{Y_i(b)}\}$.
  For each transition $\langle u_1, \sigma, \varphi_c, \varphi_s, r_1, r_2, u_2\rangle \in \delta_i$:
  - u_1 (resp.: u_2) are the source (resp.: target) of the transition;
  - $\sigma \in \Sigma$ is an input symbol (the trigger);
  - $\varphi_c, \varphi_s \in C(X)$ are constraints over the clocks in X. The *current constraint* φ_c is the guard evaluated w.r.t. the current clock valuation, while the *return constraint* φ_s is evaluated w.r.t. the clock valuation restored after a Return transition;
  - r_1 and r_2 denote the subset of clocks to restore and reset, respectively. □

Component A_1 is assumed to be the initial component, *i.e.*, the component initially called from the external environment. Notice that for a component A_i: (i) *internal transitions* have source in the set $N_i \cup Retns_i$ and target in the set $(N_i \setminus Ex_i)$; (ii) *Call transitions* have target in the set $Calls_i$; (iii) *Return transitions* have target in the set Ex_i. For the sake of readability, from now on we shall use the more convenient notation $u_1 \xrightarrow[r_1,r_2]{\sigma,\varphi_1,\varphi_2} u_2$ to represent transitions, where triggers and guards decorate the top side of the arrow, while updates decorate the bottom side.

A TRSM is called *well-formed* if all internal and Call transitions are not allowed to restore clocks and have return constraint $\varphi_s = True$. Formally, for all i, if $u_1 \xrightarrow[r_1,r_2]{\sigma,\varphi_c,\varphi_s} u_2 \in \delta_i$, $u_2 \in (N_i \setminus Retns_i) \cup Calls_i$, then $r_1 = \emptyset$ and $\varphi_s = True$.

We use the following abbreviations: $N = \bigcup_i N_i$, $B = \bigcup_i B_i$, $Calls = \bigcup_i Calls_i$, $Retns = \bigcup_i Retns_i$, $En = \bigcup_i En_i$ and $Ex = \bigcup_i Ex_i$.

We shall now give the semantics of TRMSs in terms of timed Labeled Transition Systems (timed LTSs). Let $b_{init} \notin B$ be an additional box name denoting the calling external environment. A *global state* of a TRSM $\mathcal{T} = \langle A_1, \ldots, A_n, X\rangle$ is a tuple $gs = \langle b_1 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}\rangle$, where: (i) $b_1 \cdots b_s$ is a sequence of boxes belonging to the language $(b_{init} \cdot B^*) \cup \{\epsilon\}$; (ii) u is either a control node or a return node of the form (b, ex); (iii) $\mathbf{v}_1 \cdots \mathbf{v}_s$ is a sequence of valuations; and (iv) $\mathbf{v}$ is a valuation. Intuitively, the sequence $b_1 \cdots b_s$ represents the history of the invoked components, $\mathbf{v}_1 \cdots \mathbf{v}_s$ is the history of the valuations stored at the invocation times of the corresponding components, u is the current control node and $\mathbf{v}$ is the current clock valuation. For all global states with a non-empty invocation history, the first box b_1 in the history is always equal to b_{init}, a fresh box name intuitively corresponding to the external environment's name for the initial component A_1.

Let us consider a global state $gs = \langle b_1 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}\rangle$, with $u \in N_j \cup Retns_j$. Let $b_i \in B_{j_i}$, for $2 \leq i \leq s$. Intuitively, j_i is the index of the component containing the i-th box b_i in the invocation history. We say that gs is *well-formed* if one of the following conditions holds:

- $s = 0$ and $u = (b_{init}, ex)$, for some $ex \in Ex_1$ and $b_{init} \notin B$;
- $s = 1$ and $j = 1$ (no invocation has occurred yet and state u belongs to A_1);
- $s > 1$, $j_2 = 1$, $j = Y_{j_s}(b_s)$ and $j_i = Y_{j_{i-1}}(b_{i-1})$, for $2 < i < s$.

The first condition describes terminating global states, where the computation has reached an exit state of the initial component A_1. The second condition takes care of the case where only the invocation of the initial component A_1 from the external environment has occurred. Finally, the last condition deals with the general case and ensures that : *(i)* box b_2 (the first actual invocation in the history) belongs to the initial component A_1 (i.e., $j_2 = 1$); *(ii)* the current state u belongs to the component (with index j) corresponding to the last invoked box b_s (i.e., $Y_{j_s}(b_s)$); and *(iii)* the correct caller-callee relationship holds between (the components of) adjacent boxes in the history. In particular, each entered box b_i, with $i > 2$, must belong to the component with index $j_i = Y_{j_{i-1}}(b_{i-1})$, corresponding to the box b_{i-1} preceding b_i in the history. The set of well-defined global states is denoted by GS.

The semantics of a TRSM over the alphabet Σ is given by the timed Labeled Transition System (timed LTS) $\langle GS, GS_0, \Sigma^{\mathcal{D}^{\geq 0}}, \Delta \rangle$, where:

- $GS_0 \subseteq GS$ is the set of states of the form $\langle b_{init}, u, \mathbf{v}_0, \mathbf{v}_0 \rangle$, with $u \in En_1$ and $\mathbf{v}_0(x) = 0$, for all $x \in X$;
- $\Sigma^{\mathcal{D}^{\geq 0}}$ is the set $\Sigma \cup \{\tau(t) : t \in \mathcal{D}^{\geq 0}\}$, where Σ and $\mathcal{D}^{\geq 0}$ are disjoint;
- $\Delta \subseteq GS \times \Sigma^{\mathcal{D}^{\geq 0}} \times GS$ is the transition relation. For $gs = \langle b_1 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v} \rangle$, with $u \in N_j$ and $b_s \in B_m$, $\langle gs, \sigma, gs' \rangle \in \Delta$ (notationally, $gs \xrightarrow{\sigma} gs'$) whenever one of the following holds:
  1. **Progress transition**: $\sigma = \tau(t)$ and $gs' = \langle b_1 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}' \rangle$, with $\mathbf{v}' = \mathbf{v} + t$ and $t \in \mathcal{D}^{\geq 0}$;
  2. **Reset transition**: if $u \xrightarrow[\emptyset, r_2]{\sigma, \varphi_c, True} u' \in \delta_j$, with $u' \in (N_j \setminus Ex_j)$, and $\mathbf{v} \in \varphi_c$, then $gs' = \langle b_1 \cdots b_s, u', \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}' \rangle$, with $\mathbf{v}' = \mathbf{v}\downarrow_{r_2}$;
  3. **Call transition**: if $u \xrightarrow[\emptyset, r_2]{\sigma, \varphi_c, True} (b', en) \in \delta_j$, with $(b', en) \in Calls_j$ and $\mathbf{v} \in \varphi_c$, then $gs' = \langle b_1 \cdots b_s\, b', en, \mathbf{v}_1 \cdots \mathbf{v}_s\, \mathbf{v}, \mathbf{v}' \rangle$, with $\mathbf{v}' = \mathbf{v}\downarrow_{r_2}$;
  4. **Return transition**: if $u \xrightarrow[r_1, r_2]{\sigma, \varphi_c, \varphi_s} u' \in \delta_j$, with $u' \in Ex_j$, and $\mathbf{v} \in \varphi_c$ and $\mathbf{v}\uparrow_{(\mathbf{v}_s, r_1)} \in \varphi_s$, then $gs' = \langle b_1 \cdots b_{s-1}, (b_s, u'), \mathbf{v}_1 \cdots \mathbf{v}_{s-1}, \mathbf{v}' \rangle$, with $\mathbf{v}' = (\mathbf{v}\uparrow_{(r_1, \mathbf{v}_s)})\downarrow_{r_2}$.

A *Progress transition* occurs when the control remains in the same node and only a time progress occurs. *Reset transition* corresponds to internal transition of a component and can reset a subset of clocks. When a box is entered and its corresponding component is invoked, a *Call transition* occurs. In this case the current clock valuation is stored and a subset of clocks are possibly reset afterwards. Finally, when an exit node is entered a *Return transition* occurs, which returns the control to the invoking component. In this case, the current constraint φ_c of the transition must be satisfied by the current clock valuation, a subset of clocks (those specified in the restore set decorating the transition) are restored to their original values at invocation time. The transition can be taken if the resulting restored valuation also satisfies the return constraint φ_s. If this is the case, the new current valuation is obtained from the restored valuation by resetting all the clocks in the reset set of the transition.

Notice that the timed LTS defined above, besides the symbols in Σ, also has symbols representing the lapse of time (e.g. $\tau(t)$, for $t \in \mathcal{D}^{\geq 0}$), and satisfies the usual properties of temporal determinism, time additivity and 0-delay as defined in [13] and reported below:

- *Temporal Determinism*: for all states $gs, gs', gs'' \in GS$ and $t \in \mathcal{D}^{\geq 0}$, if $gs \xrightarrow{\tau(t)} gs'$ and $gs \xrightarrow{\tau(t)} gs''$, then $gs' = gs''$;
- *Time Additivity*: for all states $gs, gs' \in GS$ and $t_1, t_2 \in \mathcal{D}^{\geq 0}$, if $gs \xrightarrow{\tau(t_1 + t_2)} gs'$, then $gs \xrightarrow{\tau(t_1)} gs''$ and $gs'' \xrightarrow{\tau(t_2)} gs'$, for some $gs'' \in GS$;
- *0-Delay*: for all states $gs, gs' \in GS$, $gs \xrightarrow{\tau(0)} gs'$ if and only if $gs = gs'$.

A run λ of $\mathcal{T}$ is a path in the LTS $\langle GS, GS_0, \Sigma^{\mathcal{D}^{\geq 0}}, \Delta \rangle$ for $\mathcal{T}$ having the form

$$\lambda = gs_0 \xrightarrow{\sigma_0} gs_1 \xrightarrow{\sigma_1} gs_2 \cdots gs_{k-1} \xrightarrow{\sigma_{k-1}} gs_k$$

where $gs_i \xrightarrow{\sigma_i} gs_{i+1} \in \Delta$, for all $0 \leq i < k$. It is an *initial* run, if $gs_0 \in GS_0$. By $\Lambda_{\mathcal{T}}$ we denote the set of all runs of $\mathcal{T}$. For any two global states $gs_i, gs_j \in GS$, we say that gs_j is *reachable* from gs_i, written $gs_i \rightarrow^* gs_j$, if there is a run $\lambda = gs_i \xrightarrow{\sigma_i} gs_{i+1} \xrightarrow{\sigma_{i+1}} gs_{i+2} \cdots gs_{i+k-1} \xrightarrow{\sigma_{i+k-1}} gs_{i+k}$ with $gs_{i+k} = gs_j$.

An (initial) run

$$\lambda = gs_0 \xrightarrow{\sigma_0} gs_1 \xrightarrow{\sigma_1} gs_2 \cdots gs_{k-1} \xrightarrow{\sigma_{k-1}} gs_k$$

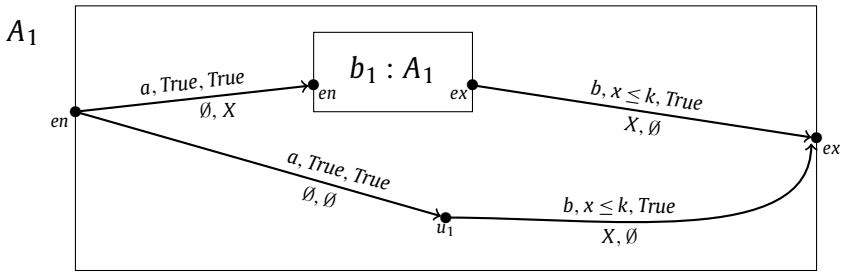


Fig. 1. The TRSM of Example 1.

induces a *timed sequence* $tseq(\lambda) \in (\Sigma \times \mathcal{D}^{\geq 0})^\star$ of the form $(a_0, t_0)(a_1, t_1) \dots (a_{m-1}, t_{m-1})$, with $m \leq k$, where for some sequence of indexes $j_0 < j_1 < \cdots < j_{m-1}$ the following hold:

1. for all $0 \leq h < k$, $\sigma_h \in \Sigma$ iff $h = j_i$, for some $0 \leq i < m$;
2. $a_i = \sigma_{j_i}$, for all $0 \leq i < m$;
3. $t_i = \sum_{h < j_i, \sigma_h = \tau(t_h)} t_h$.

An (initial) run $gs_0 \xrightarrow{\sigma_0} gs_1 \xrightarrow{\sigma_1} gs_2 \cdots \xrightarrow{\sigma_{k-1}} gs_k$ is *accepting* if gs_k is of the form $\langle \epsilon, (b_{init}, ex), \epsilon, \mathbf{v} \rangle$, with $ex \in Ex_1$. That is, an accepting run is a run which leads to an exit node of the initial component A_1. A timed sequence $\alpha \in (\Sigma \times \mathcal{D}^{\geq 0})^\star$ is *accepted* by a TRSM $\mathcal{T}$, if there exists a accepting run λ of $\mathcal{T}$, such that $tseq(\lambda) = \alpha$. The language accepted by a TRSM $\mathcal{T}$ is the set of timed sequences accepted by $\mathcal{T}$ and will be denoted by $\mathcal{L}(\mathcal{T})$. For a run λ with $tseq(\lambda) = (a_0, t_0)(a_1, t_1) \dots (a_m, t_m)$, the *untimed sequence*, $untseq(\lambda)$, induced by λ is the sequence obtained by dropping the timestamps associated with the symbols in $tseq(\lambda)$, namely $a_0 a_1 \dots a_m$.

Remark Both the syntax and the semantics of TRSMs presented here slightly differ from the ones proposed originally in [6] which, in turn, generalize those proposed in [5]. The main difference is that in the present formulation Return transitions belong to the callee instead of the caller. Indeed, a Return transition leads from a node to an exit node of the callee, while in [6] and [5] they lead from boxes to nodes of the caller. Therefore, in the present formulation the exit mechanism implicit in the semantics, moving the computation from the callee back to the caller, is performed by the callee instead of the caller. While this choice has no relevant impact on the properties of the framework, we believe this interpretation to be closer to the intuitive semantics of a return operation.

An additional benefit of this formulation is that, differently from [6], the introduction of silent τ-transitions can be avoided. Silent transitions are known to have no effect on the expressive power of Pushdown Automata (and, therefore, of RSMs) and can be eliminated. However, this is not the case in general when dense time is involved: Timed Automata with τ-transitions are, indeed, more expressive than standard Timed Automata (see [7,8]). Hence, preventing τ-transitions allows for a more direct investigation of the intrinsic expressive power of the clock valuation store/restore mechanism of TRSMs in Section 5.

Example 1. In Fig. 1 we show a TRSM $\mathcal{T} = \langle A_1, X \rangle$, with a single clock x (i.e., $X = \{x\}$) and $Y_1(b_1) = 1$, accepting the timed language

$$\mathcal{L}(\mathcal{T}) = \{(a, t_1) \cdots (a, t_n)(b, t_{n+1}) \cdots (b, t_{2n}) : n \geq 1 \text{ and } \Delta_{2n-i+1} + \Delta_i \leq k, 1 \leq i \leq n\},$$

where $\Delta_i = t_i - t_{i-1}$, taking $t_0 = 0$, and k is some fixed constant value. An intuition of the time constraint of the language is given in Fig. 2.

Notice that the untimed version of the language accepted by the TRSM above is precisely the context-free language $L = \{a^n b^n : n \geq 1\}$.

The TRSM contains a single component A_1, which may be invoked recursively. The local behavior inside the component consists of transitions triggered by the symbols a and b. The transitions triggered by b may be delayed by recursive invocations, performed by entering box b_1. Clock x is reset at each invocation and restored on exiting the component. The enforced requirement is, therefore, that the local behavior, abstracting away the time possibly spent by the recursive computation, is performed within time k. It is meaningful to observe that even though a PTA can model invocations and returns by using its control stack, it could not guarantee the temporal requirement above since, in order to check $\Delta_{2n-i+1} + \Delta_i \leq k$, it would require an unbounded number of clocks, one for each recursive call. □

We introduce now two subclasses of TRSMs obtained by suitably constraining the ability of resetting and restoring clocks associated with transitions. These subclasses will be compared in Sections 5 and 6 with respect to expressive power, decidability and complexity properties.

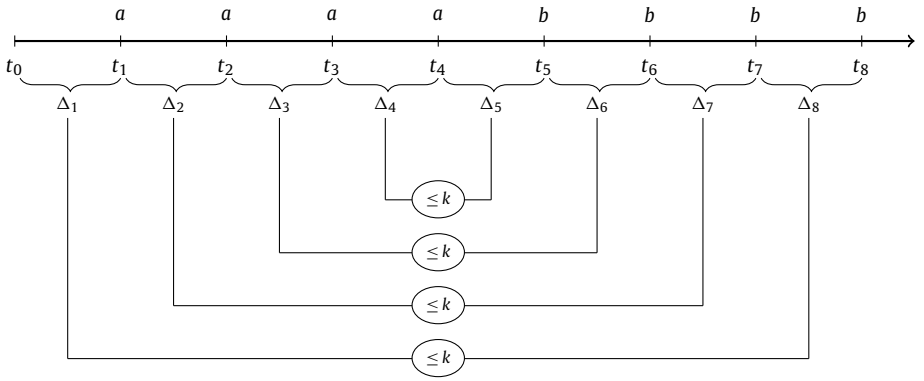


Fig. 2. An example of timed word of length 8 (and $n = 4$) accepted by the TRSM of Example 1.

Local TRSM (L-TRSM) is the class of TRSMs, where Return transitions are of the form

$$u \xrightarrow[X,r]{\sigma,\varphi_c,\varphi_s} u'.$$

This restriction forces the whole set of clocks to be restored at each Return transition.

Initialized TRSM (I-TRSM) is the class of L-TRSM, where Call transitions are of the form

$$u \xrightarrow[\emptyset,X]{\sigma,\varphi_c,True} (b, en).$$

This restriction forces the whole set of clocks to be reset at each Call transition.

Unlike the general class TRSMs, where subsets of clock values can be restored, in Local TRSM all the clocks are restored on exiting an invoked component. Therefore, the clock valuations at return time are the same clock valuations at the corresponding invocation time. This allows to model in a very natural way a notion of time "local" to a component, which abstracts away the elapse of time within an invoked component. In the class I-TRSM the abstraction gets even stronger, since all clocks are reset at invocation time, hence invoked components are "initialized" with all the clocks set to 0. This implies that the temporal behavior of the invoked component is completely isolated from the caller component: neither the temporal information flows from the caller to the callee nor from the callee to the caller.

We end this section with an example of a TRSM which is (syntactically) neither in I-TRSM nor in L-TRSM.

Example 2. Let us consider a language of timed sequences having the form:

$$\mathcal{L} = \{(a, t_1) \cdots (a, t_n)(b, t_{n+1}) \cdots (b, t_{2n})(\square, t_{2n+1}) : n \geq 2 \text{ and } \Delta_{2n-i} = \Delta_i\},$$

where $\Delta_i = t_{i+1} - t_i$ and $\square$ is an additional termination symbol for each word in the language. Notice that the untimed version of the language above corresponds to the context-free language $L = \{a^n b^n \square : n \geq 1\}$. As for the timed part, the sequence is required to satisfy a mirror distribution of the delays between consecutive symbols. Fig. 4 shows the example of a timed word (whose untimed form is $a^4 b^4 \square$) accepted by the automaton and the corresponding mirror constraints on the time intervals. Notice that, the timed language above exhibits a context-free property both in the untimed part and in the temporal sequence of timestamps. This shows the main difference with respect to PTAs ([15]) where there is no means to check context-free properties on times. The considered timed language is accepted by the TRSM in Fig. 3.

The TRSM consists of two components A_1 and A_2 and uses two clocks x and y. Component A_1 takes care of the word with only one a and one b, which need not satisfy any time constraint. Component A_2, which is responsible for checking the time property characterizing $\mathcal{L}$, is called by A_1 if at least two a's occur in the timed word and invokes itself recursively on each additional a. Observe that, since the Return transitions leading to state ex_2 in A_2 restore a proper subsets of clocks, namely $\{x\}$, the TRSM neither belongs to the class L-TRSM nor to the class I-TRSM.

The entry node en_1 is the initial state, whereas each recursive invocation start from the entry node en_2 in A_2. Clock x is used in component A_2 to record the time interval Δ_i between two adjacent a's (see Fig. 4). Hence, once in component A_2, on reading a sequence of a's the component recursively invokes itself, saving the values Δ_i, contained in the valuations of clock x, in the history of valuations. After reading the first occurrence of b only Return transitions triggered by b can be taken (the first return being from node u_3, while all the following ones start from the return node (b_2, ex_2)). On reading the sequence of b's, clock y is used to record the value Δ_i for two adjacent occurrences of b. The value of clock y is checked for equality with respect to the time distance between the mirror pairs of a symbols, which is recovered in clock x by the restore operation (while the current value of y is retained). Notice that the equality constraints occur as restore constraints, thus ensuring that the current value of clock y is compared to the recovered value of clock x.

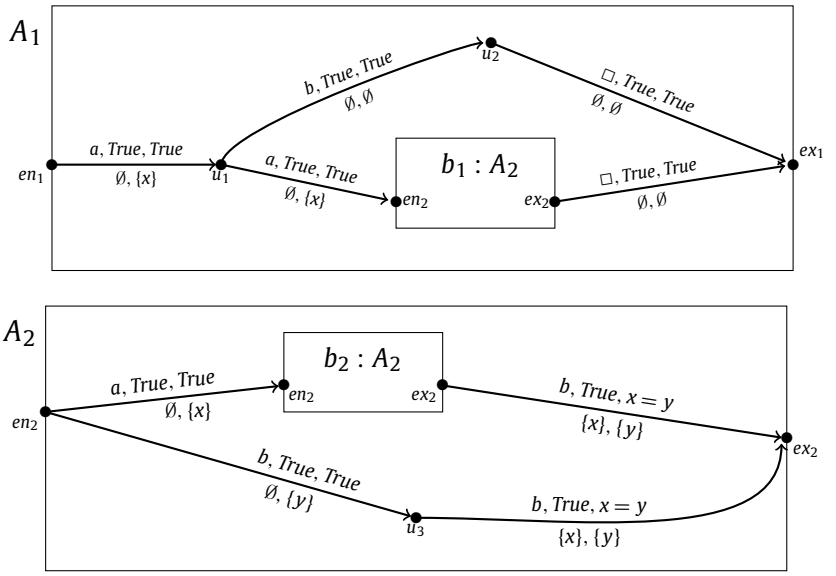


Fig. 3. The TRSM for Example 2.

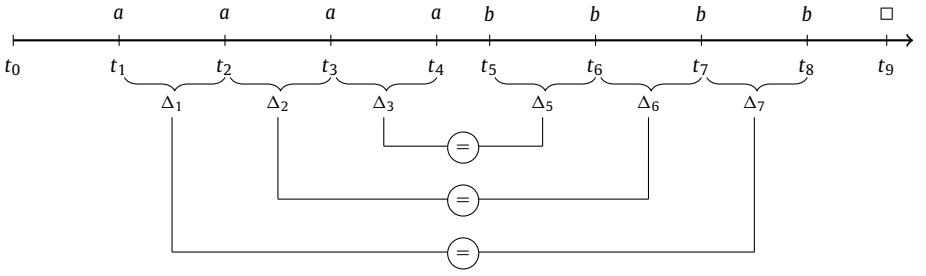


Fig. 4. An example timed word of length 9 (and $n = 4$) recognized by the TRSM of Example 2.

3. Extended Pushdown Timed Automata

Just like RSMs are tightly connected with PDAs [5], TRSMs can be set in relation with a suitable extension of PDAs, called *Extended Pushdown Timed Automata* (EPTAs), enriched with a set of clocks and with an additional stack used to store clock valuations.

We start by recalling the syntax and semantics of PDAs, as considered in [5]. A PDA $\mathcal{P}$ over an alphabet Σ is a tuple $\langle Q, Q_0, \Gamma, \bot, T\rangle$, where:

- Q is a (finite) set of states and $Q_0 \subseteq Q$ is the set of initial states;
- Γ is a (finite) set of stack symbols and $\bot \in \Gamma$ is the initial stack symbol;
- $T \subseteq Q \times \Sigma \times \Gamma \times \Gamma^{\leq 2} \times Op \times Q$ is the transition relation, where $Op = \{Swap, Push, Pop\}$ and $\Gamma^{\leq 2}$ denotes the set of (possibly empty) sequences of symbols in Γ of length at most 2.

A configuration of a PDA $\mathcal{P}$ is a tuple $\langle q, w\rangle$, where $q \in Q$ is the current control state and $w \in \Gamma^*$ is the content of the control stack. The set of all configurations is denoted by PC.

The semantics of PDAs is given by a LTS $\langle PC, PC_0, \Sigma, \Delta\rangle$, where:

- PC_0 is $\{\langle q, \bot\rangle : q \in Q_0\}$;
- $\Delta \subseteq PC \times \Sigma \times PC$ is the transition relation. For $pc = \langle q, w\rangle$, $\langle pc, \sigma, pc'\rangle \in \Delta$ whenever one of the following holds:
  1. **Swap transition**: if $q \xrightarrow[\gamma, Swap]{\sigma, a} q' \in T$, $w = a \cdot w''$, $\gamma \in \Gamma$ then $pc' = \langle q', \gamma \cdot w''\rangle$;
  2. **Push transition**: if $q \xrightarrow[\gamma, Push]{\sigma, a} q' \in T$, $w = a \cdot w''$, $\gamma \in \Gamma^2$, then $pc' = \langle q', \gamma \cdot w''\rangle$;
  3. **Pop transition**: if $q \xrightarrow[\epsilon, Pop]{\sigma, a} q' \in T$, with $w = a \cdot w''$, then $pc' = \langle q', w''\rangle$.

An EPTA is a Pushdown Automaton enriched with a set of clocks and with an additional stack used to store/restore clock valuations.

Definition 2. An EPTA $\mathcal{E}$ over Σ is a tuple $\langle Q, Q_0, X, \Gamma, \bot, T\rangle$, where:

- Q is a (finite) set of states and $Q_0 \subseteq Q$ is the set of initial states;
- X is a (finite) set of clocks and Γ is a finite stack alphabet;
- $\bot \in \Gamma$ is the initial stack symbol;
- $T \subseteq Q \times \Sigma \times \Gamma \times \Gamma^{\leq 2} \times \mathcal{C}(X) \times \mathcal{C}(X) \times 2^X \times 2^X \times Op \times Q$ is the transition relation, with $Op = \{Reset, Store, Restore\}$. □

Intuitively, for each transition $\langle q_1, \sigma, \gamma_1, \gamma_2, \varphi_c, \varphi_s, r_1, r_2, op, q_2\rangle \in T$ (denoted by $q_1 \xrightarrow{\gamma_1,\sigma,\varphi_c,\varphi_s}_{\gamma_2,op,r_1,r_2} q_2$ from now on), q_1 (resp.: q_2) is the source (resp.: the target) state; σ is the input symbol; γ_1 is the symbol on top of the control stack; γ_2 is the string (of length at most 2) which replaces the symbol on top of the control stack; $\varphi_c, \varphi_s \in \mathcal{C}(X)$ are clock constraints, respectively the *current constraint* and the *stack constraint*; op is the operation requested on the valuation stack. The set r_2 contains clocks to be reset at the end of the transition, while r_1 contains the clocks to restore from the valuation on top of the valuation stack. If $op = Store$, the current clock valuation is stored on top of the valuation stack, while if $op = Restore$ the current valuation is updated by restoring the clocks in r_1 with the values stored in the valuation on top of the valuation stack. Similarly to TRSMs, an EPTA is *well-formed* if each *Reset* and *Store* transition has $r_1 = \emptyset$ and $\varphi_s = True$.

Notice that, according to the definition above, the two stacks of an EPTA can be operated on independently from one another. Indeed, the behavior of the control stack is dictated by the length of the string γ_2 in each transition. If the length is 0, the transition performs a *Pop* operation on the stack, if the length is 1 the transition performs a *Swap* and if the length is 2 the performed operation is a *Push*. On the other hand, the valuation stack is operated on according to the labels *Reset*, *Store* and *Restore*.

A global configuration of an EPTA $\mathcal{E}$ is a tuple $\langle q, \mathbf{v}, w, \mathbf{d}\rangle$, where $q \in Q$ is the current control state, $\mathbf{v}$ is the current clock valuation, $w \in \Gamma^*$ is the content of the control stack, and $\mathbf{d} \in (X \to \mathcal{D}^{\geq 0})^*$ is the content of the valuations stack. The set of all global configurations is denoted by GC.

The semantics of EPTAs is given by a timed LTS $\langle GC, GC_0, \Sigma^{\mathcal{D}^{\geq 0}}, \Delta\rangle$, where:

- GC_0 is $\{\langle q, \mathbf{v}_0, \bot, \mathbf{v}_0\rangle : q \in Q_0\}$, where $\mathbf{v}_0(x) = 0$, for every $x \in X$;
- $\Delta \subseteq GC \times \Sigma^{\mathcal{D}^{\geq 0}} \times GC$ is the transition relation. For $gc = \langle q, \mathbf{v}, w, \mathbf{v}_s \cdot \mathbf{d}\rangle$, $\langle gc, \sigma, gc'\rangle \in \Delta$ whenever one of the following holds:
  1. **Progress transition**: $\sigma = \tau(t)$ and $gc' = \langle q, \mathbf{v}', w, \mathbf{v}_s \cdot \mathbf{d}\rangle$, with $\mathbf{v}' = \mathbf{v} + t$ and $t \in \mathcal{D}^{\geq 0}$;
  2. **Reset transition**: if $q \xrightarrow{a,\sigma,\varphi_c,True}_{\gamma,Reset,\emptyset,r_2} q' \in T$, with $\mathbf{v} \in \varphi_c$ and $w = a \cdot w''$, then $gc' = \langle q', \mathbf{v}', \gamma \cdot w'', \mathbf{v}_s \cdot \mathbf{d}\rangle$ with $\mathbf{v}' = \mathbf{v} \downarrow_{r_2}$;
  3. **Store transition**: if $q \xrightarrow{a,\sigma,\varphi_c,True}_{\gamma,Store,\emptyset,r_2} q' \in T$, with $\mathbf{v} \in \varphi_c$ and $w = a \cdot w''$, then $gc' = \langle q', \mathbf{v}', \gamma \cdot w'', \mathbf{v} \cdot \mathbf{v}_s \cdot \mathbf{d}\rangle$ with $\mathbf{v}' = \mathbf{v} \downarrow_{r_2}$;
  4. **Restore transition**: if $q \xrightarrow{a,\sigma,\varphi_c,\varphi_s}_{\gamma,Restore,r_1,r_2} q' \in T$, with $\mathbf{v} \in \varphi_c$, $\mathbf{v}\uparrow_{(r_1,\mathbf{v}_s)} \in \varphi_s$, $w = a \cdot w''$, then $gc' = \langle q', \mathbf{v}', \gamma \cdot w'', \mathbf{d}\rangle$, with $\mathbf{v}' = (\mathbf{v}\uparrow_{(r_1,\mathbf{v}_s)})\downarrow_{r_2}$.

When a Progress transition occurs, there is only a clock progress, the control remains in the same state and the two stacks are left unchanged. A Reset transition pops a symbol from the control stack while simultaneously pushing a string of symbols onto the control stack. In addition, the clocks contained in the reset set are reset, while the valuation stack is left unchanged. A Store transition behaves as a Reset transition except that, in addition, it pushes the current clock valuation onto the valuation stack. A Restore transition modifies the control stack similarly to a Reset transition and, in addition, pops a valuation from the valuation stack, restoring the specified subset of clock values, according to the popped clock valuation. Notice that for a Restore transition to be enabled both the current constraint φ_c, evaluated w.r.t. the current valuation, and the stack constraint φ_s, evaluated w.r.t the popped valuation, must hold.

The notions of (initial) run induced by a timed sequence, can be defined exactly as in the case of TRSMs. $\Lambda_{\mathcal{E}}$ denotes the set of runs of $\mathcal{E}$ and $gc \to^*_{\mathcal{E}} gc'$ denotes the reachability relation between global configurations. A finite run is *accepting* if the control stack in its last global configuration is empty. Acceptance in EPTAs is, therefore, defined by empty stack. As for TRSMs, a timed sequence α is accepted by an EPTA $\mathcal{E}$ if there is an accepting run over α. With $\mathcal{L}(\mathcal{E})$ we denote the language accepted by $\mathcal{E}$, i.e., the set of timed sequences accepted by $\mathcal{E}$.

We introduce three subclasses of EPTAs, defined by suitably constraining the operations associated with transitions. In Section 4 these classes will be naturally related with the classes of the TRSMs defined in the previous section. In all the restricted classes, the type of operations performed on the control stack are tightly coupled with the operations on the valuation stack, synchronizing the operations on the two stacks. Specifically, a *Reset* operation will be coupled with a *Swap* operation on the control stack, a *Store* operation will be coupled with a *Push* operation, and a *Restore* operation will be coupled with a *Pop* operation. This syntactic constraint precisely characterizes the wider subclass of Synchronized EPTA. The other two subclasses intuitively correspond to the class of Local TRSM and Initialized TRSM, respectively.

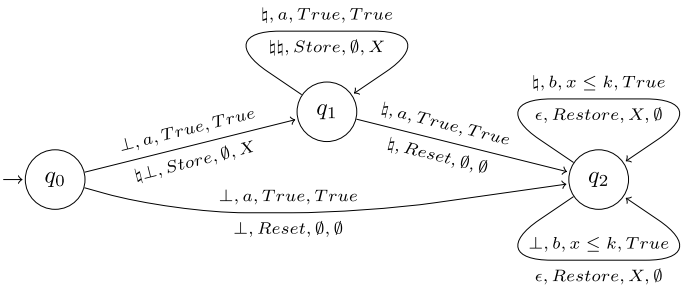


Fig. 5. The EPTA of Example 1.

Synchronized EPTA (S-EPTA) is the subclass of EPTAs where:

- Reset transitions have the form $q_1 \xrightarrow[a_2,Reset,\emptyset,r]{a_1,\sigma,\varphi_c,True} q_2$, with $a_1 \in \Gamma$, $a_2 \in \Gamma$ (reset operations can only swap a symbol at the top of the control stack);
- Store transitions have the form $q_1 \xrightarrow[\gamma,Store,\emptyset,r]{a,\sigma,\varphi_c,True} q_2$ with $a \in \Gamma$ and $\gamma \in \Gamma^2$ (store operations always perform a push on the control stack);
- Restore transitions have the form $q_1 \xrightarrow[\epsilon,Restore,r_1,r_2]{a,\sigma,\varphi_c,\varphi_s} q_2$, with $a \in \Gamma$ (restore operations always pop a symbol from the control stack);

Local EPTA (L-EPTA) is the subclass of S-EPTA where:

- restore transitions have the form $q_1 \xrightarrow[\epsilon,Restore,X,r]{a,\sigma,\varphi_c,\varphi_s} q_2$, with $a \in \Gamma$ (each restore operation restores the whole set of clocks);

Initialized EPTA (I-EPTA) is the subclass of L-EPTA where:

- store transitions have the form $q_1 \xrightarrow[\gamma,Store,\emptyset,X]{a,\sigma,\varphi_c,True} q_2$, with $a \in \Gamma$, $\gamma \in \Gamma^2$ (each store operation always resets the whole set of clocks).

An immediate consequence of the restrictions above is that, as opposed to EPTA, the control and valuation stacks of any global configuration of an S-EPTA (as well as L-EPTA and I-EPTA) always have the same length.

It is easy to see that PTAs, as introduced in [15], can be seen as a subclass of S-EPTA. PTAs are, indeed, Timed Automata enriched with a control stack. As a consequence, transitions of a PTA are guarded by a clock constraint, read symbols from the input string and the top of the (control) stack. Moreover, they can reset clocks, push new symbols onto the stack or pop a symbol from it. This class of automata can be embedded into S-EPTA by mapping each Push transition of the PTA into a suitable Store transition that pushes the same stack symbols and resets the same set of clocks; mapping each Pop transition t into a Restore transition, which does not restore any clocks, has the current constraint equal to the clock constraint of t, and a trivial stack constraint; and mapping each Reset transition of the PTA into a corresponding Reset transition of the S-EPTA. As we shall see in Section 5, PTAs are weaker than EPTAs, as there exist automata in I-EPTA that cannot be simulated by any PTA.

Before giving a formal account of the behavioral and expressive equivalence between TRSMs and EPTAs in the next section, the following two examples provide an insight on the relationship between the two formalisms by considering the context-free timed languages of Examples 1 and 2 and illustrating two EPTAs accepting them.

Example 3. Let us consider the language of Example 1, namely

$$\mathcal{L}(\mathcal{T}) = \{(a,t_1)\cdots(a,t_n)(b,t_{n+1})\cdots(b,t_{2n}) : n \geq 1 \text{ and } \Delta_{2n-i+1} + \Delta_i \leq k, 1 \leq i \leq n\},$$

where $\Delta_i = t_i - t_{i-1}$, taking $t_0 = 0$, and k is some fixed constant value. This language is accepted by the EPTA (with $\Gamma = \{\bot, \natural\}$ and $\natural$ a suitable additional stack symbol) in Fig. 5, which has a single clock x (i.e., $X = \{x\}$) used to record the time delay of each symbol a. The initial state q_0 guesses between a word with $n = 1$, taking the lower reset transition to state q_2, and a word with $n \geq 2$, by moving to state q_1. In this latter case, at each a read, the automaton stores the stack symbol $\natural$ onto the control stack, stores the current clock valuation on the valuation stack, and resets x. While in q_1 it also guesses the last a in the input word, by non-deterministically choosing to wait for another a or to move to state q_2 and start reading the matching b's. Once in state q_2, at each b the clock constraint $x \leq k$ is checked to ensure that the sum of the time delay of a b and its matching a in the input word does not exceed the constant quantity k. Moreover, clock x is restored, so that the resulting value of x always corresponds to the time delay since the a matching the next expected b. Notice that, since $X = \{x\}$, the EPTA does fulfill all the constraints of the class I-EPTA.

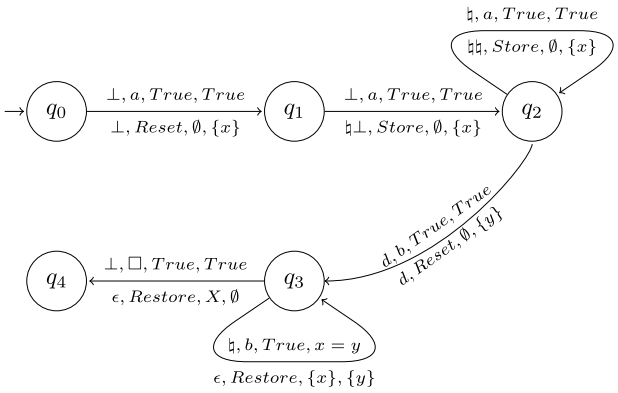


Fig. 6. The EPTA of Example 2.

Example 4. In Fig. 6 we show an EPTA, whose behavior is triggered by the same timed language introduced in Example 2, namely the language consisting of timed sequences having the form:

$$\mathcal{L} = \{(a, t_1) \cdots (a, t_n)(b, t_{n+1}) \cdots (b, t_{2n})(\Box, t_{2n+1}) : n \geq 2 \text{ and } \Delta_{2n-i} = \Delta_i\},$$

where $\Delta_i = t_{i+1} - t_i$.

The automaton starts in state q_0 with symbol $\bot$ on the control stack ($\Gamma = \{\bot, \natural\}$). At the first a read, the clock x is reset and the control moves to state q_1. In either state q_1 and q_2, at any successive read of symbol a the automaton pushes the symbol $\natural$ on top of the stack, stores the current valuation on the valuation stack, resets x and moves to q_2. Intuitively, at each reading of a, x records the time passed since the previous a (i.e. $\mathbf{v}(x) = \Delta_i$ at the time of reading the $(i+1)$-th a). Similarly, clock y will the be used to record the time spent between two successive b's. Indeed, on reading the first b from state q_2, the automaton resets clock y and moves to q_3. From there, at every b the automaton first restores the value of clock x from the valuation stack, checks that $x = y$ (notice that the value of y is not restored and contains the time since the last b) and finally resets clock y. It is immediate to verify that the EPTA defined above belongs to the class S-EPTA. □

4. Relationship between TRSMs and EPTAs

Similarly to RSMs and PDAs, TRSMs and EPTAs are clearly tightly related. To provide a formal account of this relationship, we shall resort to the standard notion of *bisimulation* between timed LTSs and show that the two formalisms, and the corresponding subclasses, are equally expressive in the sense that they accept the same languages.

Let us first recall the notion of bisimulation between timed LTSs. Let $\mathcal{LT} = \langle S, S_0, \Sigma^{\mathcal{D}^{\geq 0}}, \Delta\rangle$ and $\mathcal{LT}' = \langle S', S'_0, \Sigma^{\mathcal{D}^{\geq 0}}, \Delta'\rangle$ be two timed LTSs, we say that $\mathcal{LT}$ *simulates* $\mathcal{LT}'$ if there exists a *simulation relation* $\prec \subseteq S \times S'$ defined as follows:

- initialization: for all $s_0 \in S_0$, there is a $s'_0 \in S'_0$, with $s_0 \prec s'_0$;
- propagation: for all $\sigma \in \Sigma^{\mathcal{D}^{\geq 0}}$, if $s_1 \prec s'_1$ and $s_1 \xrightarrow{\sigma} s_2$, then there exists $s'_2 \in S'$ such that $s'_1 \xrightarrow{\sigma} s'_2$ and $s_2 \prec s'_2$.

If, in addition, the relation $\prec^{-1}$, defined as $s \prec^{-1} s' \Leftrightarrow s' \prec s$, is also a simulation relation, then $\prec$ is a called a *bisimulation relation*. Two LTSs $\mathcal{LT}$ and $\mathcal{LT}'$ are strongly bisimilar, written $\mathcal{LT} \equiv \mathcal{LT}'$, if there is a bisimulation relation between $\mathcal{LT}$ and $\mathcal{LT}'$. Clearly, given two automata A and B, whenever their corresponding LTSs are strongly bisimilar, then $\mathcal{L}(A) = \mathcal{L}(B)$.

The following result ensures the bisimulation equivalence between corresponding classes of TRSMs and EPTAs.

Theorem 1. *For any TRSM (resp.:* L-TRSM, I-TRSM*) $\mathcal{T}$, there exists a strongly bisimilar* S-EPTA *(resp.:* L-EPTA, I-EPTA*) $\mathcal{E}$ and vice versa.*

Proof. We shall first provide a uniform transformation mapping any TRSM into a strongly bisimilar S-EPTA. Given a TRSM $\mathcal{T} = \langle A_1, \ldots, A_n, X\rangle$ over an alphabet Σ, a uniform translation allows to build an S-EPTA $\mathcal{E}$ over the same alphabet that strongly bisimilar to $\mathcal{T}$. In the following we assume that A_i has the form $\langle N_i \bigcup B_i, Y_i, En_i, Ex_i, \delta_i\rangle$. The S-EPTA $\mathcal{E} = \langle Q, Q_0, X, \Gamma, \bot, T\rangle$ is defined as follows:

- $Q = \{q_u : u \in N \cup Retns\} \cup \{(\bot, ex) : ex \in Ex_1\}$;
- $Q_0 = \{q_u : u \in En_1\}$;
- $\Gamma = B \cup \{\bot\}$;
- T is defined as follows:

1. **[Reset transition]** if $u \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} u' \in \delta_j$, with $u' \notin Ex_j$, then $q_u \xrightarrow{\rho,\sigma,\varphi_c,True}_{\rho,Reset,\emptyset,r} q_{u'} \in T$, for every $\rho \in \Gamma$;
2. **[Call transition]** if $u \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} (b, en) \in \delta_j$, with $b \in B_j$, $Y_j(b) = k$ and $en \in En_k$, then $q_u \xrightarrow{\rho,\sigma,\varphi_c,True}_{b\cdot\rho,Store,\emptyset,r} q_{en} \in T$, for every $\rho \in \Gamma$;
3. **[Return transition]** if $u \xrightarrow{\sigma,\varphi_c,\varphi_s}_{r_1,r_2} ex \in \delta_j$, with $ex \in Ex_j$, then $q_u \xrightarrow{\rho,\sigma,\varphi_c,\varphi_s}_{\epsilon,Restore,r_1,r_2} q_{(\rho,ex)} \in T$, for every $\rho \in \Gamma$.

It is immediate that the following relation $\prec$, between global states of $\mathcal{LT}_\mathcal{T}$ and global configurations of $\mathcal{LT}_\mathcal{E}$, is indeed a strong bisimulation:

1. $\langle b_{init}, en, \mathbf{v}_0, \mathbf{v}_0\rangle \prec \langle q_{en}, \mathbf{v}_0, \bot, \mathbf{v}_0\rangle$, for all $en \in Ex_1$;
2. $\langle b_{init} \cdot b_2 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}\rangle \prec \langle q_u, \mathbf{v}, b_s \cdots b_2 \cdot \bot, \mathbf{v}_s \cdots \mathbf{v}_1\rangle$, $u \in N \cup Retns$ and $s \geq 1$, and
3. $\langle \epsilon, (b_{init}, ex), \epsilon, \mathbf{v}\rangle \prec \langle q_{(\bot,ex)}, \mathbf{v}, \epsilon, \epsilon\rangle$, for all $ex \in Ex_1$.

Let us now consider the other direction. The main difference between TRSMs and EPTAs is that the latter has a direct access to the control stack, being able to test the symbol currently on top of the stack on each transition and, possibly, replace it when performing a Reset or a Store transition. This is not allowed in TRSMs, where the control stack is left implicit in the semantics. In order to simulate an EPTA, then, we shall build a corresponding TRSM where the symbol on top of the control stack is always kept track of in the current node. The simulating TRSM shall have a single machine A_1, with entry nodes of the form en_q^ρ, where q is a state of the EPDA and $\rho \in \Gamma$ corresponds to the current symbol on top of the control stack. The second problem to solve is to recover the correct symbol on top of the stack after a Restore transition. This symbol corresponds to the symbol in the second position from the top of the stack placed by the matching Store transition previously performed (recall that Store transitions push on the stack a sequence in Γ^2). Therefore, we shall keep track of that symbol by recording it in the corresponding box name. In other words, boxes in the simulating TRSM are pairs (ρ, ρ') of control stack symbols, where $\rho' \cdot \rho$ is two-symbol sequence pushed onto the stack by the Store operation.

Formally, given the S-EPTA $\mathcal{E} = \langle Q, Q_0, X, \Gamma, \bot, T\rangle$ over the alphabet Σ, we can define the one-component TRSM $\mathcal{T} = \langle A_1, X\rangle$ over the alphabet Σ, which is strongly bisimilar to $\mathcal{E}$. The TRSM $A_1 = \langle N_1 \cup B_1, Y_1, En_1, Ex_1, \delta_1\rangle$ has entry nodes $En_1 = \{en_q^b : q \in Q \text{ and } b \in \Gamma\}$, exit nodes $Ex_1 = \{ex_q : q \in Q\}$, and $N_1 = En_1 \cup Ex_1$. The set of boxes is $B_1 = \{(\bot, \bot)\} \cup (\Gamma \times \Gamma \setminus \{\bot\})$, and all boxes are mapped to A_1 (i.e. $Y_1(b) = 1$, for all $b \in B_1$). The transitions in δ_1 are defined as follows:

1. **[Reset transition]** if $q \xrightarrow{\rho,\sigma,\varphi_c,True}_{\rho',Reset,\emptyset,r} q' \in T$, then the following transitions are in δ_1:
   - $en_q^\rho \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} en_{q'}^{\rho'}$;
   - $((\rho, \hat\rho), ex_q) \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} en_{q'}^{\rho'}$, for all $\hat\rho \in \Gamma$;
2. **[Store transition]** if $q \xrightarrow{\rho,\sigma,\varphi_c,True}_{\rho'\rho'',Store,\emptyset,r} q' \in T$, then the following transitions are in δ_1:
   - $en_q^\rho \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} ((\rho'', \rho'), en_{q'}^{\rho'})$;
   - $((\rho, \hat\rho), ex_q) \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} ((\rho'', \rho'), en_{q'}^{\rho'})$, for all $\hat\rho \in \Gamma$;
3. **[Restore transition]** if $q \xrightarrow{\rho,\sigma,\varphi_c,\varphi_s}_{\epsilon,Restore,r_1,r_2} q' \in T$, then the following transitions are in δ_1:
   - $en_q^\rho \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} ex_{q'}$;
   - $((\rho, \hat\rho), ex_q) \xrightarrow{\sigma,\varphi_c,True}_{\emptyset,r} ex_{q'}$, for all $\hat\rho \in \Gamma$.

Notice that, for all $q \in Q$ and $\rho, \hat\rho \in \Gamma$, there is a transition from en_q^ρ to some state $u \in N \cup Calls$ if and only if there is a transition form $((\rho, \hat\rho), ex_q)$ to u with identical decorations (triggers, constraints and clock updates).

The following relation $\prec$, between global configurations of $\mathcal{LT}_\mathcal{E}$ and global states of $\mathcal{LT}_\mathcal{T}$ turns out to be a strong bisimulation:

1. $\langle q, \mathbf{v}_0, \bot, \mathbf{v}_0\rangle \prec \langle b_{init}, en_q^\bot, \mathbf{v}_0, \mathbf{v}_0\rangle$, for all $q \in Q_0$;
2. $\langle q, \mathbf{v}, \rho_s \cdots \rho_1, \mathbf{v}_s \cdots \mathbf{v}_1\rangle \prec gs$, for $s \geq 1$, all $q \in Q$ and all gs of one of the following forms:
   - $\langle b_{init} \cdot b_2 \cdots b_s, en_q^{\rho_s}, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}\rangle$,
   - $\langle b_{init} \cdot b_2 \cdots b_s, ((\rho_s, \hat\rho), ex_q), \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}\rangle$,

   where $b_{init} = (\bot, \bot)$, $b_i = (\rho_{i-1}, \rho_i)$, for all $2 \leq i < s$ and taking $\rho_1 = \bot$, and $b_s = (\rho_{s-1}, \rho')$ for $\rho' \in \Gamma$;
3. $\langle q, \epsilon, \epsilon, \mathbf{v}\rangle \prec \langle \epsilon, (b_{init}, ex_q), \epsilon, \mathbf{v}\rangle$, for all $q \in Q$.

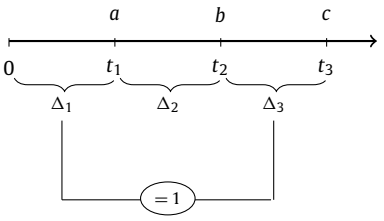


Fig. 7. The schema of timed sequences in $\mathcal{L}_0$.

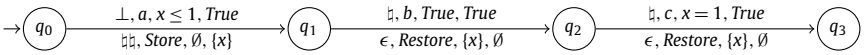


Fig. 8. The I-EPTA accepting $\mathcal{L}_0$.

It is immediate from the definitions of the two constructions above that they actually preserve the subclasses, namely they map TRSM into S-EPTA (I-TRSM into I-EPTA and L-TRSM into L-EPTA) and vice versa.

5. Expressiveness of EPTAs

In this section we discuss the expressive power of TRSMs and EPTAs and their subclasses with respect to their ability of accepting timed languages. The expressiveness results are stated for the classes of EPTA and the results are then lifted to TRSMs, by exploiting the behavioral bisimilarities stated in the previous section. When studying the expressiveness properties, it is useful to consider both the untimed and the timed component of the accepted languages. As we have shown in Examples 3 and 4, EPTAs can express non-regular properties both on their untimed and timed component. Since an EPTA exploits one stack for the control part and another stack for the clock valuations, it is not surprising that already the smallest class of EPTAs (namely, I-EPTA) is more expressive than TAs, since non-regular context free properties can be expressed in the untimed component of EPTA-languages, whereas the untimed component of a language accepted by a TA is always regular.

More interestingly, we show that there exist I-EPTA-languages, whose untimed component is indeed a regular language and which cannot be accepted by any TA, due to the temporal property they express. The proof of Theorem 2 show such a language. The same language can also be proved not to be expressible by PTAs either. In addition, we shall see that I-EPTA, L-EPTA and S-EPTA form a strict hierarchy of increasing expressive power with respect to the temporal properties of the languages they accept.

In the following we denote by $\mathcal{URe}g$(I-EPTA) the set of I-EPTA containing automata accepting some timed language $\mathcal{L}$ whose untimed version is regular, i.e., such that $\mathcal{U}nt(\mathcal{L}) \in \mathcal{Re}g\mathcal{L}ng$. In the following, we shall refer to such timed languages as *control-regular* EPTA-languages.

Let us start by comparing the expressive power of TAs and I-EPTA w.r.t. the properties they can enforce on the timed component of a language. To this end, we compare the class of control-regular I-EPTA-languages with timed regular languages, i.e. the timed languages accepted by TAs. The following theorem states that there exist control-regular I-EPTA-languages which cannot be accepted by any TA.

Theorem 2. *Timed Automata are strictly less expressive than* $\mathcal{URe}g$(I-EPTA)*.*

Proof. Since every TA is a special case of I-EPTA, every timed regular language can be accepted by some I-EPTA.

We provide a simple I-EPTA-language which cannot be accepted by any TA. Given a timed word $(a_1, t_1)(a_2, t_2)\cdots(a_n, t_n)$, let $\Delta_i = t_i - t_{i-1}$, for any $1 \leq i \leq n$. Let us consider the following timed language:

$$\mathcal{L}_0 = \{(a, t_1)(b, t_2)(c, t_3) : \Delta_1 + \Delta_3 = 1\}.$$

Fig. 7 shows the schema of timed sequences in $\mathcal{L}_0$, where Δ_2 can be any positive time interval and the sum of the intervals Δ_1 and Δ_3 must be equal to 1.

The automaton reported in Fig. 8 accepts this language and clearly belongs to the class I-EPTA. The automaton uses a single clock x. In the initial state q_0 it can read a if the value of x, which equals Δ_1, is less then or equal to 1. It then stores the current clock valuation onto the valuation stack, resets x and moves to q_1. When b is read in q_1, the automaton just restores the original clock valuation, so that the value of x is restored to Δ_1, and moves to q_2. Finally, on reading c in q_2, the automaton checks whether the value of x, which is precisely $\Delta_1 + \Delta_3$ at this time, is equal to 1. If so, it accepts the timed sequence by moving to the state q_3 with empty stack.

Assume there is a TA which accepts $\mathcal{L}_0$. Since all timed sequences of $\mathcal{L}_0$ have fixed length 3, we can assume, w.l.o.g., that the underlying graph of the automaton is acyclic and each path of the automaton leading to an accepting state contains precisely three transitions (one for each symbol) and 4 states. As a consequence, there can only be a finite number of paths

Table 1
Values of clock expressions on reading each symbol.

	a	b	c
x	Δ_1	$\Delta_1+\Delta_2$	$\Delta_1+\Delta_2+\Delta_3$
y	Δ_1	Δ_2	$\Delta_2+\Delta_3$
z	Δ_1	$\Delta_1+\Delta_2$	Δ_3
$x-y$	0	Δ_1	Δ_1
$x-z$	0	0	$\Delta_1+\Delta_2$
$y-z$	0	$-\Delta_1$	Δ_2

leading to an accepting state. By duplicating states shared by different paths, we can build an equivalent automaton with a finite number of states and transitions and whose paths do not share states.

Along each run induced by one of the paths clocks can be only reset on reading a symbol. On reading each symbol the set of clocks can, then, be partitioned into equivalence classes, where the clocks in each class evaluate to the same value. On reading a all the clocks belong to the same class. On reading b, they can be partitioned into two classes, those reset on reading a and those not reset. On reading c, there are three classes: the class of the clocks never reset, those reset only on reading a and those reset on reading b (notice that the clocks reset both on reading a and on reading b fall into the same equivalent class of those reset on reading b). Hence, we can assume, w.l.o.g., that the automaton has only three clocks x, y and z, and that every run of the automaton never resets x, resets y on reading a and resets z on reading b.

On reading a, all the clocks have the same value Δ_1. On reading b, $\mathbf{v}(x)=\mathbf{v}(z)=\Delta_1+\Delta_2$, $\mathbf{v}(y)=\Delta_2$ and $\mathbf{v}(x)-\mathbf{v}(y)=\mathbf{v}(z)-\mathbf{v}(y)=\Delta_1$. Finally, on reading c we have that $\mathbf{v}(x)=\Delta_1+\Delta_2+\Delta_3$, $\mathbf{v}(y)=\Delta_2+\Delta_3$, $\mathbf{v}(z)=\Delta_3$, $\mathbf{v}(x)-\mathbf{v}(y)=\Delta_1$, $\mathbf{v}(y)-\mathbf{v}(z)=\Delta_2$, $\mathbf{v}(x)-\mathbf{v}(z)=\Delta_1+\Delta_2$. Table 1 reports the correspondence between the expressions, possibly occurring in clock constraints, and the corresponding values.

Observe that any clock constraint on the transitions reading a and b can be expressed by an equivalent clock constraint on the transition reading c. For instance, the constraint $k_1\le x\le k_2$ (constraining Δ_1 to lie in the interval $[k_1,k_2]$) on reading a would be equivalently expressed by $k_1\le x-y\le k_1$ on reading c and, similarly, the constraints $k_1\le y\le k_2$ and $k_3\le x\le k_4$ (respectively, constraining Δ_2 and $\Delta_1+\Delta_2$) on reading b can be expressed equivalently by the constraints $k_1\le y-z\le k_2$ and $k_3\le x-z\le k_4$ on reading c, respectively. The automaton can then be transformed into an equivalent automaton where for each path: the clock constraints φ^a, guarding transitions reading a, and φ^b, guarding transitions reading b, are replaced by *True*; their equivalent versions (as described above) are conjoined to the current constraint of the last transition reading c along that path, thus obtaining the constraint φ^c.

Let π be a path of the above defined automaton and let $\mathcal{L}_\pi$ be the set of timed sequences accepted by the runs induced by π. If $\mathcal{L}_\pi=\emptyset$ then the path π can simply be deleted from the automaton without affecting the set of accepted timed sequences. Therefore, in the following we shall assume that every path in the automaton admits some accepting run. Then, the following claim holds.

Claim 1. *For every path π, $\mathcal{L}_\pi\subseteq\mathcal{L}_0$ if and only if the constraint φ^c, guarding the transition reading c in π, satisfies one of the following conditions:*

1. *there exists a constant $k\ge 1$ such that φ^c implies both $x=k$ and $y-z=k-1$;*
2. *there exists a constant $0\le\hat{k}\le 1$ such that φ^c implies both $x-y=\hat{k}$ and $z=1-\hat{k}$.*

The proof of the claim is reported in the appendix at the end of the paper. The claim ensures that every path π of the timed automaton must either fix the total time of an accepted timed sequence or fix the time instant when a is read. Clearly, there are infinitely many different values both for the total time of a time sequence in $\mathcal{L}$ and for the time instant where a is read. However, the automaton has only a finite number of paths. We can, then, conclude that the automaton cannot accept all and only the timed sequences in $\mathcal{L}_0$. □

A similar result can be proved to hold between I-EPTA and L-EPTA. Similarly to the previous result, we prove that L-EPTA are strictly more expressive than I-EPTA by showing an L-EPTA-language exhibiting temporal constraints which cannot be expressed by any I-EPTA.

Theorem 3. *The class* I-EPTA *is a proper subclass of* L-EPTA.

Proof. We shall show that the following L-EPTA-language

$$\mathcal{L}_1=\{(a,t_1)(b,t_2)(c,t_3):\Delta_1+\Delta_2=1 \text{ and } \Delta_1+\Delta_3=1\}$$

cannot be accepted by any automaton in I-EPTA. Fig. 9 shows the general schema of the timed sequences in $\mathcal{L}_1$.

It is easy to see that the L-EPTA automaton in Fig. 10 does accept $\mathcal{L}_1$. Notice that the only difference with the automaton in Fig. 8 is that the Store transition from q_0 to q_1 does not reset clocks and that the Restore transition from q_1 to q_2 can

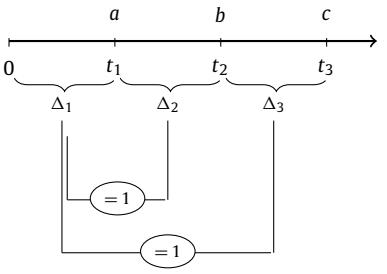


Fig. 9. The schema of timed sequences in $\mathcal{L}_1$.

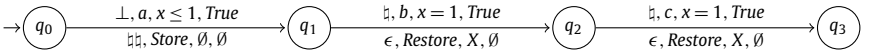


Fig. 10. The L-EPTA accepting $\mathcal{L}_1$.

Table 2
Values of clock expressions on reading each symbol for I-EPTA-path.

	a		b		c	
	φ_c^a	φ_s^a	φ_c^b	φ_s^b	φ_c^c	φ_s^c
x	Δ_1	0	Δ_2	Δ_1	$\Delta_1 + \Delta_3$	0
y	Δ_1	0	Δ_2	Δ_1	$\Delta_1 + \Delta_3$	0
z	Δ_1	0	Δ_2	0	Δ_3	0
$x - y$	0	0	0	0	0	0
$x - z$	0	0	0	Δ_1	Δ_1	0
$y - z$	0	0	0	Δ_1	Δ_1	0

then check that on reading b the value of x (which corresponds to $\Delta_1 + \Delta_2$) is actually equal to 1 just by checking the constraint $x = 1$.

Assume there is a I-EPTA automaton accepting $\mathcal{L}_1$. We proceed similarly to the proof of Theorem 2 and assume that all the paths of (the graph underlying) the automaton do not have common states. Along each path, besides Reset transitions, the automaton can also perform Store and Restore transitions. Notice that, for an induced run of a path π to be accepting, the last transition reading c in π must be a Restore transition. Therefore, a path π of the automaton can be of one of following two forms: either *(i)* both the transition reading a and the one reading b are Reset transitions (*timed path*), or *(ii)* the transition reading a is a Store transition and the one reading b is a Restore transition (*store path*).

Let π be a path of the above defined automaton and let $\mathcal{L}_\pi$ be the set of timed sequences accepted by the runs induced by π. Once again, we shall assume, w.l.o.g., that every path in the automaton admits some accepting run.

The following claim (whose proof is in the appendix) holds of any *timed path* π of the automaton.

Claim 2. *Let π be a* timed path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_1$ if and only if the current constraint φ_c^c of the transition reading c satisfies the following condition:*

1. *there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^c implies $x - z = 1$, $x - y = \hat{k}$ and $z = 1 - \hat{k}$.*

In *store paths*, no Reset transitions are allowed, however Store transitions do reset all the clocks, while Restore transitions may reset some clocks after restoring the clock valuation on top of the valuation stack. Hence, all the clocks have the same values up to the execution of the first restore transition on reading b and we assume, w.l.o.g., that on reading b only clock z is reset. Table 2 reports the abstract values of all the possible clock expressions that can occur in the guards of each Store/Restore transition reading the corresponding symbol. The two columns associated with each symbol $\sigma \in \{a, b, c\}$, named φ_c^σ and φ_s^σ, refer to the value of the clock expression when the current constraint φ_c of the transition reading σ is evaluated and when the stack constraint φ_s of that transition is evaluated, respectively. Therefore, we can also assume that both constraints φ_c^a and φ_s^a are *True* for the transition triggered by a, as well as both the stack constraints φ_s^b and φ_s^c. Indeed, on reading a all the clocks expressions evaluate either to 0 or to Δ_1 and they can be expressed by a suitable constraint φ_c^c. Similarly, for φ_s^b on reading b. Therefore, each path may have non-trivial current constraints φ_c only on the transitions reading b and c.

The following claim (see the appendix for a formal proof) holds of any *store path* of the automaton.

Claim 3. *Let π be a* store path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_1$ if and only if the constraints φ_c^b and φ_c^c satisfy the following condition:*

1. *there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^b implies $y = \hat{k}$ and φ_c^c implies both $x = 1$ and $x - z = 1 - \hat{k}$.*

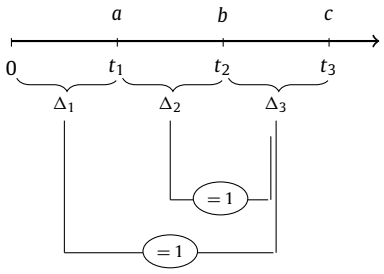


Fig. 11. The schema of timed sequences in $\mathcal{L}_2$.

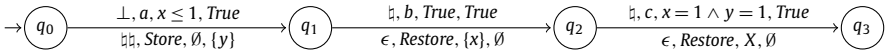


Fig. 12. The S-EPTA accepting $\mathcal{L}_2$.

The two claims above ensure that each path of the I-EPTA automaton accepts timed sequences having fixed constant durations for Δ_1 and Δ_3. However, since the number of paths is finite while there are infinitely many pairs of values for Δ_1 and Δ_3 compatible with timed sequences in $\mathcal{L}_1$, we obtain a contradiction. □

The following result completes the picture and shows that L-EPTA are strictly less expressive than S-EPTA. While this result also follows indirectly from the undecidability result for the reachability problem in S-EPTA and the decidability of the same problem in L-EPTA reported in Section 6, we believe that a direct proof of the result gives more insight on the expressive ability of the respective classes of automata.

Theorem 4. *The class* L-EPTA *is a proper subclass of* S-EPTA.

Proof. We shall prove that the following S-EPTA-language cannot be accepted by any L-EPTA.

$$\mathcal{L}_2 = \{(a, t_1)(b, t_2)(c, t_3) : \Delta_1 + \Delta_3 = 1 \text{ and } \Delta_2 + \Delta_3 = 1\}.$$

Fig. 11 shows the schema of the timed sequences in $\mathcal{L}_2$. The S-EPTA reported in Fig. 12 accepts this language.

Assume there exists an L-EPTA automaton accepting $\mathcal{L}_2$. The same transformation applied in the proofs of Theorems 2 and 3 gives us a L-EPTA automaton whose paths are disjoint (do not share states) and contain three transitions (one reading a, one reading b and the last one reading c). As in the case of the I-EPTA automaton of Theorem 3, the last transition reading c in any path must be a Restore transition. Hence, each path π of the automaton is either a *timed path* (no Store transition occurs) or a *store path*.

The following claim holds for any *timed path* π of the automaton (see the appendix the formal proof):

Claim 4. *Let* π *be a* timed path *of the automaton. Then* $\mathcal{L}_\pi \subseteq \mathcal{L}_2$ *if and only if the current constraint* φ_c^c *on the transition reading* c *satisfies the following condition:*

1. *there exists a constant* $0 \leq \hat{k} \leq 1$ *such that* φ_c^c *implies* $y = 1$, $x - y = \hat{k}$ *and* $z = 1 - \hat{k}$.

Store paths have a structure similar to *store paths* in the proof of Theorem 3 (i.e., a Store transition reading a followed by two Restore transition on reading b and c, respectively). However, differently from that case, Store transitions may reset a proper subset of the clocks. Therefore, all the clocks have the same values up to the execution of the Store transition on reading a. The clocks reset (resp., not reset) by that transition will have the same values on reading b. However, that transition restores all the clock and, possibly reset some of them. Hence, we can distinguish only three equivalence classes of clocks. Those never reset, those reset on reading a and those reset on reading b, and a single clock for each class will suffice. We can then assume, w.l.o.g., that clock x is never reset, clock y is reset on reading a and only clock z is reset on reading b. Table 3 reports the abstract values of all the possible clock expressions that can occur in the guards of each store/restore transition reading the corresponding symbol. The two columns associated with each symbol $\sigma \in \{a, b, c\}$ have the same interpretation as in Theorem 3. We can assume, w.l.o.g., that the constraints φ_c^a and φ_s^a are both *True*, as well as the constraints φ_s^b and φ_c^c, and, then, that each path may only have non-trivial constraints φ_c^b and φ_c^c.

The following claim (proved in the appendix) holds for *store paths* of the automaton.

Claim 5. *Let* π *be a* store path *of the automaton. Then* $\mathcal{L}_\pi \subseteq \mathcal{L}_2$ *if and only if the constraints* φ_c^b *and* φ_c^c *satisfy the following condition:*

1. *there exists a constant* $0 \leq \hat{k} \leq 1$ *such that* φ_c^b *implies* $y = \hat{k}$ *and* φ_c^c *implies both* $x = 1$ *and* $z = 1 - \hat{k}$.

Table 3
Values of clock expressions on reading each symbol.

	a		b		c	
	φ_c^a	φ_c^a	φ_c^b	φ_s^b	φ_c^c	φ_s^c
x	Δ_1	Δ_1	$\Delta_1 + \Delta_2$	Δ_1	$\Delta_1 + \Delta_3$	0
y	Δ_1	0	Δ_2	Δ_1	$\Delta_1 + \Delta_3$	0
z	Δ_1	0	Δ_2	0	Δ_3	0
$x - y$	0	Δ_1	Δ_1	0	0	0
$x - z$	0	Δ_1	Δ_1	Δ_1	Δ_1	0
$y - z$	0	0	0	0	Δ_1	0

The two claims above ensure that each path of the L-EPTA automaton accepts timed sequences having fixed constant durations for Δ_1, Δ_2 and Δ_3. However, there are infinitely many triples of values for those durations compatible with timed sequences in $\mathcal{L}_2$ but only a finite number of paths. Hence, we obtain a contradiction. □

As a consequence of Theorem 1, that establishes an effective equivalence of TRSMs and EPTAs, and of Theorems 3 and 4, we can state the following result.

Corollary 1. I-TRSM *is a proper subclass of* L-TRSM *and* L-TRSM *is a proper subclass of* TRSM.

Another interesting consequence of Theorem 3 and Theorem 4 is that I-EPTA and L-EPTA are not closed under intersection with timed regular languages.

For the case of I-EPTA, consider the timed language:

$$\mathcal{L}_4 = \{(a, t_1)(b, t_2)(c, t_3) : \Delta_1 + \Delta_2 = 1\}.$$

$\mathcal{L}_4$ is clearly a timed regular language accepted by some TA. On the other hand, notice that the language $\mathcal{L}_1$ of Theorem 3 is precisely the intersection of $\mathcal{L}_4$ and $\mathcal{L}_0$, the language accepted by a I-EPTA introduced in Theorem 2. As shown in the proof of Theorem 3, however, $\mathcal{L}_1$ cannot be accepted by any I-EPTA.

For the case of L-EPTA, consider the timed language $\mathcal{L}_5 = \{(a, t_1)(b, t_2)(c, t_3) : \Delta_2 + \Delta_3 = 1\}$, which is clearly a timed regular language accepted by some TA. Now, notice that $\mathcal{L}_2$ is the intersection of the timed regular language $\mathcal{L}_5$ and the L-EPTA-language $\mathcal{L}_0 = \{(a, t_1)(b, t_2)(c, t_3) : \Delta_1 + \Delta_3 = 1\}$. However, as shown in the proof of Theorem 4, $\mathcal{L}_2$ cannot be accepted by any L-EPTA.

In summary we have the following Corollary.

Corollary 2. *Neither of classes* I-EPTA *and* L-EPTA *(resp.* I-TRSM *and* L-TRSM*) are closed under intersection with TA.*

Notice, however, that by a standard construction one could easily prove that the class S-EPTA, and hence the class TRSM, are closed under intersection with TAs.

As already mentioned at the end of Section 3, PTAs correspond to a syntactic restriction of S-EPTA, where Restore transitions do not restore clocks. Moreover, by observing that PTAs can be viewed as Timed Automata with a control stack and that, as far as the time dimension is concerned, PTAs behaves exactly as TAs, the proof of Theorem 2 can easily be adapted to show that there are I-EPTA-languages that cannot be accepted by PTAs. In particular, there are no PTAs that can accept the language $\mathcal{L}_0$ defined above. Indeed, the same transformation applied to Timed Automata in the proof of Theorem 2 can also be applied to a PTA with acyclic paths, obtaining an automaton whose underling graph only contains disjoint paths. Since clock constraints and clock operations in PTAs are the same as in TAs, following the same argument as in the proof of Theorem 2, we conclude that no PTAs can express the temporal constraint characterizing the timed sequences in $\mathcal{L}_0$.

Theorem 5. *There exist* I-EPTA*-languages which are not PTA-languages.*

Clearly, being I-EPTA $\subset$ L-EPTA, the same holds of L-EPTA as well. However, since one cannot avoid to restore clocks in the classes I-EPTA and L-EPTA, it does not seem possible to easily simulate PTAs in either class and accept, e.g., context free timed languages requiring global time constraints on the duration of the timed sequences. An example of such a language is $\{(a, t_1) \cdots (a, t_n)(b, t_{n+1}) \cdots (b, t_{2n}) : n > 0, t_{2n} = k\}$. Indeed, we conjecture that PTAs are expressively not comparable to either I-EPTA or L-EPTA.

Table 4 summarizes the expressiveness results for EPTA (and TRSM) provided in this section.

Remark The clock valuations store-restore mechanism of EPTA (and the corresponding call-and-return mechanism in TRSM) also allows to simulate the distinguishing feature of SWAs, namely the ability to freeze/unfreeze the elapse of time for clocks [14]. In particular, if we allow for τ-transitions in EPTA, any SWA $\mathcal{W}$ can be simulated by a suitable S-EPTA $\mathcal{E}$. While

Table 4
Expressiveness results on TRSMs and EPTAs.

I-TRSM	$\subset$	L-TRSM	$\subset$	TRSM		
$\equiv$		$\equiv$		$\equiv$		
I-EPTA	$\subset$	L-EPTA	$\subset$	S-EPTA	$\subseteq$	EPTA
$\cup$				$\cup$		
TA		$\subset$		PTA		

a formal encoding is beyond the scope of the paper, the intuitive idea is that $\mathcal{E}$ can keep track of the current value of frozen clocks using the valuation stack. Then, whenever a clock constraint guarding a transition need to be checked, the automaton recovers the correct clock values by restoring the values of the frozen clocks. More precisely, each transition tr of the SWA $\mathcal{W}$ can be simulated by two successive transitions in $\mathcal{E}$. The first transition tr_1 is a Restore transition which: *(i)* restores all the clocks which are frozen in the source state of tr; *(ii)* checks the clock constraint φ of t after the restore (setting the stack constraint to φ and the current constraint to *True*); and *(iii)* resets all the clocks reset by tr. The second transition tr_2, which must be taken instantly, is a silent Store transition, which simply stores back the clock valuations into the valuation stack. Since the reachability problem for general SWAs is known to be undecidable (see [17,14]), the decidability results of the same problem for both I-EPTA and L-EPTA provided in Section 6.2 entail that neither class can simulate SWAs.

6. The reachability problem: decidability and complexity results

In this section we study the computational properties of TRSMs and EPTAs, by considering the reachability problem for both formalisms. More precisely, we prove that the problem is undecidable for the general classes TRSMs and EPTAs, but it is decidable for L-TRSM and L-EPTA and, in particular for those classes, we prove that the problem is **EXPTIME**-complete. We also establish the complexity of the problem for the classes of I-TRSM and I-EPTA, which are proved to be **PSPACE**-complete as for TAs. The latter result is particularly interesting, as it shows that enriching TAs with a restricted form of the clocks store-restore mechanism does increase the expressiveness, while retaining the same computational complexity.

The *reachability problem* for an untimed global state $\langle b_1 \cdots b_s, u \rangle$ of a TRSM is to determine whether, for some clock valuations $\mathbf{v}_1, \ldots, \mathbf{v}_s$ and $\mathbf{v}$, the global state $\langle b_1 \cdots b_s, u, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v} \rangle$ is reachable from some initial global state. A similar notion of the reachability problem can be given for EPTAs.

6.1. Undecidability of the reachability problem for EPTAs and TRSMs

The undecidability result is obtained for the class S-EPTA. The bisimulation result in Theorem 1 ensures undecidability of the class TRSM as well.

We show that S-EPTA allows to simulate increment and decrement of clocks and, therefore, we can reduce the halting problem of 2-counter Minsky machines, which is known to be undecidable [21], to the reachability problem for S-EPTA. Since we are interested in studying the computational properties of EPTAs and TRSMs, in the rest of the section we shall abstract away the input symbols from the model. Therefore, we consider EPTAs and TRSMs restricted to a one-letter alphabet $\Sigma = \{\sigma\}$ and a two symbols stack alphabet $\Gamma = \{\bot, \natural\}$.

We recall that a 2-counter Machine is a finite set of labeled instructions over two counters, c_1 and c_2. Let $c \in \{c_1, c_2\}$ be a counter, a 2-counter Machine admits only two forms of instructions on c:

- *increment* of the form $p : c := c + 1$; goto p';
- *decrement* of the form p : if $c > 0$ then $c := c - 1$; goto p_1 else goto p_2.

The machine starts at an instruction labeled p_0 with $c_1 = c_2 = 0$ and stops at a special instruction labeled *HALT*. The *halting problem* for a 2-counter Machine consists in deciding whether the machine reaches the instruction *HALT*.

The idea is that the two counters can be simulated by two clocks and the instructions of increment and decrement can be simulated by sequences of transitions of an S-EPTA, exploiting the ability of storing and restoring clock valuations. We use a pair of clocks, x_1 and x_2, to simulate the two counters c_1 and c_2. Two supplementary clocks, y and y', are used to correctly simulate the operations on counters. Fig. 13 shows the gadget automata belonging to S-EPTA, which perform the increment (in the left) and the decrement (on the right) of a clock $z \in \{x_1, x_2\}$, leaving the other clock unchanged. The gadget for the increment operation consists of a Store transition, which saves the value of the clocks, followed by a Restore transition that is taken after one time unit and restores the value of all the clocks except for z. The transitions leading from q_0^- to q_3^- are used to copy the value of z into y', while preserving the value of the other clock. After saving the value of all the clocks and resetting z, the final transition is taken when an amount of time equal to $y' - 1$ has elapsed. At this point, the current value of z is retained, while the original value of the other clock is restored. Therefore, given a 2-counter Machine $\mathcal{M}$, we can build a S-EPTA $\mathcal{E}_{\mathcal{M}}$ by suitably composing the gadget automata just described. The following theorem establishes the result (see the appendix for a formal proof).

Theorem 6. *The reachability problem for* S-EPTA *is undecidable.*

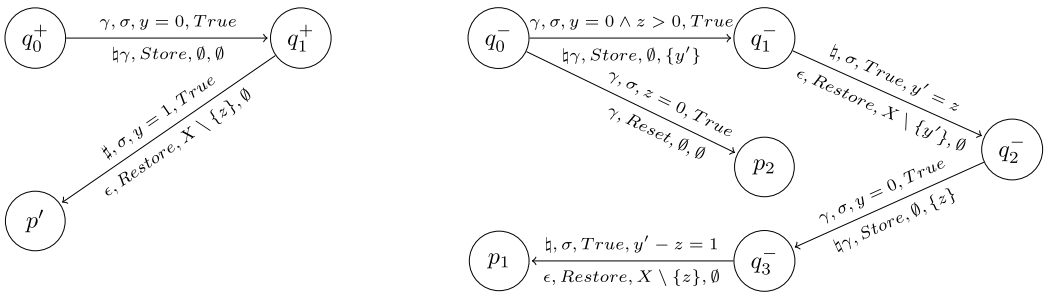


Fig. 13. The two gadgets for the increment (on the left) and the decrement (on the right) of a clock $z \in \{x_1, x_2\}$.

From the above theorem and Theorem 1 it immediately follows that the same undecidability result holds for TRSMs as well.

Corollary 3. *The reachability problem for* TRSM*s is undecidable.*

6.2. The reachability problem for L-TRSM *and* L-EPTA

We consider now the reachability problem for L-TRSM and L-EPTA and prove that it is decidable and, in particular, **EXPTIME**-complete. Given the expressive equivalence between the two formalisms, proving the result for L-EPTA will suffice. The decidability and the complexity upper bound are obtained by exploiting a standard regionalization technique, which has been used to provide finite quotient labeled transition systems preserving the reachability relation in TAs. By employing the same regionalization technique for TAs, for any given L-EPTA $\mathcal{E}$ we define a *Region Pushdown Automaton*, which preserves the reachability relation and accepts the untimed version of the language accepted by $\mathcal{E}$. To show that the reachability problem is **EXPTIME**-hard, instead, we shall provide an encoding of polynomial-space bounded *Alternating Turing Machines* (ATM) into L-EPTA.

6.2.1. The upper bound for L-EPTA

An upper bound for the reachability problem for L-EPTA is obtained by providing a uniform translation of each L-EPTA into a Region Pushdown Automaton, whose accepted language coincides with the untimed version of the language accepted by the L-EPTA.

We first introduce the notion of Clock Region for a Timed Automaton. Since diagonal constraints are allowed in L-EPTA, we exploit the regionalization technique in [13], which extends the standard regionalization construction of [3] to diagonal constraints. As usual, without loss of generality, we assume that all the constants occurring in the clock constraints of the automaton are integers.

In the second part we describe how to build a Pushdown Automaton that uses clock regions as stack symbols and can simulate an L-EPTA.

Clock regions For each clock $x \in X$ (resp. for each pair of clocks $x, y \in X$), let c_x (resp. $d_{x,y}$) be the largest integer constant c (resp. d) such that $x \sim c$ (resp. $x - y \sim d$) is a constraint occurring in the automaton with $\sim \in \{<, \leq, >, \geq, =, \neq\}$. In the case there is no occurrence of a diagonal constraint involving x and y, we take $d_{x,y}$ to be 0. A *region* is defined by a suitable set of *individual* clock constraints, *ordering* clock constraints and *difference* clock constraints.

For a clock $x \in X$ the set $\mathcal{I}_x$ of individual clock constraints is the union of the following sets:

$$\mathcal{I}_x = \{x > c_x\} \cup \{x = c : c \in \{0, 1, \ldots, c_x\}\} \cup \{c < x < c + 1 : c \in \{0, 1, \ldots, c_x - 1\}\}.$$

For a pair of clocks $x, y \in X$, the set $\mathcal{O}_{x,y}$ of ordering clock constraints is defined as follows:

$$\mathcal{O}_{x,y} = \{x - y < d, x - y = d, x - y > d : d \in \{-c_y + 1, \ldots, 0, \ldots, c_x - 1\}\}.$$

Finally, for a pair of clocks $x, y \in X$, the set $\mathcal{J}_{x,y}$ of difference clock constraints is:

$$\begin{aligned}\mathcal{J}_{x,y} = {} & \{x - y > d_{x,y}\} \cup \{x - y < -d_{y,x}\} \cup \\ & \{x - y = d : d \in \{-d_{y,x}, \ldots, 0, \ldots, d_{x,y}\}\} \cup \\ & \{d < x - y < d + 1 : d \in \{-d_{y,x}, \ldots, 0, \ldots, d_{x,y} - 1\}\}. \end{aligned} \tag{1}$$

A *clock region* is a tuple $R = \langle \mathcal{I}_R, \mathcal{O}_R, \mathcal{J}_R \rangle$, such that

1. $\mathcal{I}_R$ contains exactly one element in $\mathcal{I}_x$ for each clock $x \in X$;
2. $\mathcal{O}_R$ is the set containing, for each pair of clocks $x, y \in X$, exactly one element from $\{x - y < (c - c'), x - y = (c - c'), x - y > (c - c')\} \subseteq \mathcal{O}_{x,y}$, where the two constraints of the form $c < x < c + 1$ and $c' < y < c' + 1$ belong to $\mathcal{I}_R$;
3. $\mathcal{J}_R$ is the set containing exactly one element of $\mathcal{J}_{x,y}$ for each pair of different clocks $x, y \in X$ such that a constraint of the form $x > c_x$ or of the form $y > c_y$ belongs to $\mathcal{I}_R$.

For instance, the initial region R_0 is the tuple $\langle \mathcal{I}_{R_0}, \emptyset, \emptyset \rangle$, with $\mathcal{I}_{R_0} = \{x = 0 : x \in X\}$.

A region R describes a set of clock valuations that satisfy all the constraints defining that region. For a region R and a valuation $\mathbf{v}$, we write, by abuse of notation, $\mathbf{v} \in R$ if $\mathbf{v} \in \bigwedge_{\phi \in (\mathcal{I}_R \cup \mathcal{O}_R \cup \mathcal{J}_R)} \phi$. We denote by *Reg* the set of all regions with respect to the set of clocks X and two indexed family $\{c_x\}_{x \in X}$ and $\{d_{x,y}\}_{x,y \in X}$ of integer constants.

Regions can be proved to correspond to equivalent classes of valuations w.r.t. satisfaction of the clock constraints occurring in the considered TA. In other words, each region describes a set of valuations that cannot be distinguished by the constraints occurring in the automaton. Formally, for any region $R \in Reg$, if $\mathbf{v}, \mathbf{v}' \in R$ then $\mathbf{v} \in \varphi$ iff $\mathbf{v}' \in \varphi$, for each clock constraint φ guarding a transition of the automaton. Therefore, the clock constraint satisfaction relation can be lifted in the obvious way from valuations to regions, by defining, for $\varphi \in \mathcal{C}(X)$, $R \in \varphi$ if and only if $\mathbf{v} \in \varphi$, for all $\mathbf{v} \in R$.

The number of clock regions is clearly exponential in the number of clocks of the automaton and the maximal constants occurring in the automaton constraints. More precisely, in the case of diagonal-free constraints, each region is identified by specifying only one clock constraint for each clock and, for every pair of clocks, a corresponding ordering constraint, while the third element of the triple is not required. In this case, if k is the number of the clocks, there are at most $k! \cdot 2^k \cdot \prod_{x \in X}(2c_x + 2)$ regions (see [2]). Allowing diagonal constraints increases the number of regions to, at most, $k! \cdot 2^k \cdot \prod_{x \in X}(2c_x + 2) \cdot \prod_{x,y \in X}(4d_{x,y} + 3)$.

In order to deal with time progress, we need to introduce the notion of time successor of a region. A clock region R' is a *time successor* of a clock region R, in symbols $R' \in Succ(R)$, if and only if there exists a $t \in \mathcal{D}^{\geq 0}$ with $\mathbf{v} + t \in R'$, for all clock valuations $\mathbf{v} \in R$. Intuitively, R' is the set of valuations that can be reached from some valuation in R, by elapse of time only. Similarly, to model resets of clocks, an operation of *region reset* can be defined by setting, for $r \subseteq X$, $R{\downarrow_r}$ as the region $\{\mathbf{v}{\downarrow_r} | \mathbf{v} \in R\}$. The region $R{\downarrow_r}$ can be obtained by $R = \langle \mathcal{I}_R, \mathcal{O}_R, \mathcal{J}_R \rangle$, by replacing each individual constraint in $\mathcal{I}_R$, for $x \in r$, with the individual constraint $x = 0$ and rearranging $\mathcal{O}_R$ and $\mathcal{J}_R$ accordingly.

It is well known that the reachability problem of a Timed Automaton can be reduced to the reachability problem over its *Region Automaton* (see [2]). In this case, a Region Automaton is defined by setting its states to the set of possible pairs formed by a control state of the automaton and a region over the set of its clocks. The transitions of the Region Automaton are obtained by suitably coupling time progress with the transition relation of the automaton. Time progress is accounted for by the time successor relation over regions, clock resets are modeled by a region reset operations and clock constraints are interpreted with respect to clock regions instead of valuations.

Construction for the upper bound Decidability of the reachability problem for L-EPTA and an upper bound to its complexity are obtained by constructing, for an L-EPTA $\mathcal{E}$, a region PDA $\mathcal{P}$, such that if $\mathcal{L}(\mathcal{E})$ is the timed language accepted by $\mathcal{E}$, then its corresponding *untimed language* is accepted by $\mathcal{P}$. Recall that in a L-EPTA the control stack and the valuation stack are always synchronized. Therefore, the information contained in the two stacks can be suitably combined and stored in the single stack of a PDA. In addition, an L-EPTA can operate both on the current valuation, which can be tested against a clock constraint and reset, and on valuation stored on top of the valuation stack, which can be restored and tested against the stack constraint. Moreover, all these operations can be executed by a single transition.

Given a global configuration $gc = \langle q, \mathbf{v}_c, a \cdot \mathbf{w}, \mathbf{v}_s \cdot \mathbf{d} \rangle$ of the L-EPTA $\mathcal{E}$, we shall call the triple $\langle a, \mathbf{v}_s, \mathbf{v}_c \rangle$ the *context* of $\mathcal{E}$ at global configuration gc. In order to simulate L-EPTA operations on the control stack and on the current and stored valuations in gc, we provide the PDA with a stack alphabet containing triples of the form $\langle a, R_s, R_c \rangle$, where a is a control stack symbol, R_c and R_s are regions over the set of clocks of the L-EPTA. Each triple $\langle a, R_s, R_c \rangle$, called a *region context* in the following, is intended to encode the context of $\mathcal{E}$ at some global configuration $\langle q, \mathbf{v}_c, a \cdot \mathbf{w}, \mathbf{v}_s \cdot \mathbf{d} \rangle$, with $\mathbf{v}_s \in R_s$ and $\mathbf{v}_c \in R_c$.

The set of states of the region PDA $\mathcal{P}$ contains the same control states of $\mathcal{E}$ and, for each restore transition tr of $\mathcal{E}$ a corresponding additional state $\hat{tr}$, needed to correctly simulate Restore transitions. The intuitive idea underlying the simulation is the following. Similarly to the case of TAs, time progress is modeled by means of the successor region relation. Consider a global configuration $gc = \langle q, \mathbf{v}_c, a \cdot \mathbf{w}, \mathbf{v}_s \cdot \mathbf{d} \rangle$, resulting from some transition of $\mathcal{E}$ (or an initial global configuration, if no transition has yet occurred) and a global configuration $gc'' = \langle q, \mathbf{v}'', a \cdot \mathbf{w}, \mathbf{v}_s \cdot \mathbf{d} \rangle$, with $\mathbf{v}'' = \mathbf{v}_c + t$ for some $t \geq 0$, resulting from a time progress and from which a transition of $\mathcal{E}$ may occur. The triple $\langle a, R_s, R_c \rangle$, with $\mathbf{v}_s \in R_s$ and $\mathbf{v}_c \in R_c$, encodes the context of gc and R'' is the region containing the current valuation $\mathbf{v}''$ of gc''. Then $R'' \in Succ(R_c)$, since $\mathbf{v}'' \in R''$ and $\mathbf{v}'' = \mathbf{v}_c + t$ for some t, and $\langle a, R_s, R'' \rangle$ is the region context of gc''.

Assume that the transition taken from gc'' is a Reset transition of the form $q \xrightarrow{a,\sigma,\varphi,True} q' \in T_{\mathcal{E}}$, which leads from a state q to a state q', reads symbol a from the stack, swaps it with symbol a' and resets the clocks in r. According to the semantics of EPTAs, the resulting global configuration is $gc' = \langle q', \mathbf{v}', a' \cdot \mathbf{w}, \mathbf{v}_s \cdot \mathbf{d} \rangle$, where $\mathbf{v}' = \mathbf{v}''{\downarrow_r}$. Clearly, $R'' \in \varphi$, since $\mathbf{v}'' \in \varphi$ and, if $R' = R''{\downarrow_r}$, then $\mathbf{v}' \in R'$ and the triple $\langle a', R_s, R' \rangle$ encodes the context of the global configuration gc'. Therefore, the PDA

transition $q \xrightarrow[\langle a', R_s, R' \rangle, Swap]{\langle a, R_s, R_c \rangle, \sigma} q'$, precisely simulates the concatenation of the Progress transition and the Reset transition leading from gc to gc'.

If, instead, the transition taken from gc'' is a Store transition of the form $q \xrightarrow[a_1 \cdot a_2, Store, \emptyset, r]{a, \sigma, \varphi, True} q' \in T_{\mathcal{E}}$, then the resulting global configuration would be $gc' = \langle q', \mathbf{v}', a_1 \cdot a_2 \cdot \mathbf{w}, \mathbf{v}'' \mathbf{v}_s \cdot \mathbf{d} \rangle$, where $\mathbf{v}' = \mathbf{v}'' \downarrow_r$. To simulate this transition we first substitute the region context on top of the PDA stack with the current context of gc'', namely the triple $\langle a_2, R_s, R'' \rangle$, and then push on the stack the region context of gc', namely the triple $\langle a_1, R'', R' \rangle$ with $R' = R'' \downarrow_r$. Therefore, the PDA transition $q \xrightarrow[\langle a_1, R'', R' \rangle \langle a_2, R_s, R'' \rangle, Push]{\langle a, R_s, R_c \rangle, \sigma} q'$, precisely simulates the concatenation of the Progress transition and the Store transition leading from gc to gc'.

Finally, if the transition taken from gc'' is a Restore transition $tr = q \xrightarrow[\epsilon, Restore, X, r]{a, \sigma, \varphi_c, \varphi_s} q' \in T_{\mathcal{E}}$, then $\mathbf{v}'' \in \varphi_c$ and the resulting global configuration would be $gc' = \langle q', \mathbf{v}'_c, a' \cdot w', \mathbf{v}'_s \cdot \mathbf{d}' \rangle$, where $\mathbf{v}_s \in \varphi_s$, $\mathbf{v}'_c = \mathbf{v}_s \downarrow_r$, $\mathbf{w} = a' \cdot \mathbf{w}'$ and $\mathbf{d} = \mathbf{v}'_s \cdot \mathbf{d}'$. In order to simulate this transition, we shall break it into two parts. The first part removes the region context currently stored on the PDA stack, while the second part simulates the reset operation on the previously stored valuation, so as to obtain the new current region context of gc'. This is obtained by concatenating two PDA transitions. In order to perform the correct reset operation, we need to keep track of the reset set r of the transition. This is done by means of the additional state $\hat{tr}$, associated to transition tr, which the first PDA transition leads to and the second transition starts from. Then, the first Pop transition takes from q to state $\hat{tr}$ on reading the region context $\langle a, R_s, R_c \rangle$ of gc (recall that $R'' \in Succ(R_c)$ is such that $R'' \in \varphi_c$, since $\mathbf{v}'' \in R''$ and $\mathbf{v}'' \in \varphi_c$). The second Swap transition from $\hat{tr}$ to q', instead, on reading the region context $\langle a', R'_s, R_s \rangle$ popped up on the stack after the previous transition, completes the simulation by resetting R_s w.r.t. the reset set r of tr, thus obtaining $R'_c = R_s \downarrow_r$. This transition, then, swaps the region context on the top of the PDA stack with the resulting context $\langle a', R'_s, R'_c \rangle$, which corresponds to the region context of gc' (notice also that $R_s \in \varphi_s$ since $\mathbf{v}_s \in R_s$ and $\mathbf{v}_s \in \varphi$).

The intuition above is formalized by the following definition. Let $\mathcal{E} = \langle Q_{\mathcal{E}}, q_{0\mathcal{E}}, X_{\mathcal{E}}, \Gamma_{\mathcal{E}}, \bot_{\mathcal{E}}, T_{\mathcal{E}} \rangle$ be an L-EPTA automaton and $Reg_{\mathcal{E}}$ denote the set of regions induced by the set of clocks $X_{\mathcal{E}}$ and the two indexed family $\{c_x\}_{x \in X_{\mathcal{E}}}$ and $\{d_{x,y}\}_{x,y \in X_{\mathcal{E}}}$. Then the *region PDA* $\mathcal{P} = \langle Q_{\mathcal{P}}, q_{0\mathcal{P}}, \Gamma_{\mathcal{P}}, \bot_{\mathcal{P}}, T_{\mathcal{P}} \rangle$ of $\mathcal{E}$ is defined as follows:

- $Q_{\mathcal{P}} = Q_{\mathcal{E}} \cup \{\hat{tr} : tr \in T_{\mathcal{E}} \text{ is a restore transition}\}$;
- $\Gamma_{\mathcal{P}} = \Gamma_{\mathcal{E}} \times Reg_{\mathcal{E}} \times Reg_{\mathcal{E}}$;
- $\bot_{\mathcal{P}} = \langle \bot_{\mathcal{E}}, R_0, R_0 \rangle$;
- $T_{\mathcal{P}}$ is defined as follows:
  - for each Reset transition $q \xrightarrow[a', Reset, \emptyset, r]{a, \sigma, \varphi, True} q' \in T_{\mathcal{E}}$, $T_{\mathcal{P}}$ contains all the transitions $q \xrightarrow[\langle a', R_s, R' \rangle, Swap]{\langle a, R_s, R_c \rangle, \sigma} q'$ such that $R'' \in \varphi$ with $R'' \in Succ(R_c)$ and $R' = R'' \downarrow_r$;
  - for each Store transition $q \xrightarrow[a_1 \cdot a_2, Store, \emptyset, r]{a, \sigma, \varphi, True} q' \in T_{\mathcal{E}}$, $T_{\mathcal{P}}$ contains all the transitions of the form $q \xrightarrow[\langle a_1, R'', R' \rangle \langle a_2, R_s, R'' \rangle, Push]{\langle a, R_s, R_c \rangle, \sigma} q'$, where $R'' \in \varphi$ with $R'' \in Succ(R_c)$ and $R' = R'' \downarrow_r$;
  - for each Restore transition $tr = q \xrightarrow[\epsilon, Restore, X, r]{a, \sigma, \varphi_c, \varphi_s} q' \in T_{\mathcal{E}}$ and region $R'' \in Succ(R_c)$ with $R'' \in \varphi_c$, $T_{\mathcal{P}}$ contains the pairs of transitions $q \xrightarrow[\epsilon, Pop]{\langle a, R_s, R_c \rangle, \sigma} \hat{tr}$ and $\hat{tr} \xrightarrow[\langle a', R'_s, R'_c \rangle, Swap]{\langle a', R'_s, R_s \rangle, \tau} q'$, where $a' \in \Gamma_{\mathcal{E}}$, $R_s \in \varphi_s$, $R'_c = R_s \downarrow_r$ and R'_s is any clock region.

The following result, whose proof is reported in the appendix, provides the correctness of the construction above, by stating that it preserves the untimed language accepted by L-EPTA.

Theorem 7. *An* L-EPTA $\mathcal{E}$ *accepts a timed sequence* η *if and only if the region PDA* $\mathcal{P}$ *of* $\mathcal{E}$ *accepts the sequence* $Untime(\eta)$.

It is well known that the reachability problem for PDAs can be solved in time polynomial in the size of the PDA. Since the region PDA in the construction above is exponential in the size of the L-EPTA, we have the following upper bound.

Corollary 4. *The reachability problem for* L-EPTA *belongs to* ***EXPTIME****.*

6.2.2. The lower bound for L-EPTA

To prove that the reachability problem for L-EPTA is **EXPTIME**-hard we shall provide a reduction from the acceptance problem for PSPACE-bounded Alternating Turing Machines. Recall that the complexity class **APSPACE** (the class of languages accepted by PSPACE-bounded Alternating Turing Machines) coincides with the class **EXPTIME** (the class of languages accepted by EXPTIME-bounded Turing Machines). We first recall the definition of Alternating Turing Machine (ATM) and, in the second part, provide the actual construction mapping an ATM into a L-EPTA.

Alternating Turing Machines An Alternating Turing Machine is a tuple $\mathcal{M} = \langle Q, Q_\forall, Q_\exists, q_0, \delta, Q_F \rangle$, where:

- Q is the finite set of states;
- $Q_\forall \subseteq Q$, $Q_\exists \subseteq Q$ and $Q_F \subseteq Q$ are a partition of Q into universal, existential and final states, respectively;
- $q_0 \in Q$ is the initial state;
- $\delta \subseteq Q \times \Sigma \times \Sigma \times \{L, R\} \times Q$ is the transition relation.

For convenience, we shall write a transition $\langle q, \sigma, \overline{\sigma}, \beta, \overline{q} \rangle \in \delta$, with $\beta \in \{L, R\}$, as $q \xrightarrow[\overline{\sigma},\beta]{\sigma} \overline{q}$.

Since $\mathcal{M}$ is PSPACE-bounded we can assume, w.l.o.g., that there exists a constant $k_\mathcal{M} \geq 1$ such that, for each input $\xi \in \Sigma^*$, the space needed by $\mathcal{M}$ on input ξ is bounded by $k_\mathcal{M} \cdot |\xi|$.

Each reachable configuration of $\mathcal{M}$ over an input string ξ can be seen as a sequence in the set $\Sigma^* \times (Q \times \Sigma) \times \Sigma^*$, where the element in $(Q \times \Sigma)$ gives the current control state and the symbol in the tape on which the head is positioned. A configuration whose control state belongs to $Q_\exists$ (resp., $Q_\forall$, Q_F) is called an existential (resp., universal, final) configuration. Taking $n = k_\mathcal{M} \cdot |\xi|$ we can assume that each configuration for the input string w has exactly length n (if the length of the configuration is less than n we can complete the configuration string with a suitable sequence of the special symbol $\square \in \Sigma$). If $\xi = a_1 \cdots a_m$, the initial configuration has the form $(q_0, a_1) \cdot a_2 \cdots a_m \cdot \square^{n-m}$. From now on, we shall consider a configuration a as a word $a_1 \cdots a_n$ of length n. We denote with $\mathcal{C}_n$ the set of all possible configurations of length n. An accepting run of an ATM $\mathcal{M}$ on an input string ξ is a (finite) tree whose nodes are labeled by configurations of $\mathcal{M}$ according to the transition relation of $\mathcal{M}$. The leaves of the tree are labeled with final configurations; nodes labeled with an existential configuration have a single descendant; nodes labeled with a universal configuration have one descendant for each transition departing from state q and reading symbol $\sigma \in \Sigma$. In the following, for each state $q \in Q_\forall$ and $\sigma \in \Sigma$, let $Out(q, \sigma) \subseteq \delta$ be the set of transitions having state q as source and reading symbol σ. Moreover, assume that there is a fixed total ordering of each set $Out(q, \sigma)$ and that the function $Out(q, \sigma, i)$ gives the i-th element in the assumed ordering of $Out(q, \sigma)$. Moreover, let $Mout = Max_{q \in Q_\forall, \sigma \in \Sigma} |Out(q, \sigma)|$, be the largest number of non-deterministic transitions outgoing from a state.

More formally, an accepting run of $\mathcal{M}$ on an input ξ is a tree $T_{\mathcal{M},\xi} = \langle N, \nu \rangle$, where

- $N \subseteq \{0, \ldots, Mout - 1\}^*$ is a prefix closed set of strings;
- $\nu : N \to \mathcal{C}_n$ with $n = k_\mathcal{M} \cdot |\xi|$ is the labeling function satisfying the following constraints:
  1. $\nu(\epsilon)$ is the initial configuration;
  2. if $\nu(e) = a_1 \cdots a_{i-1} (q, \sigma) a_{i+1} \cdots a_n$ and $q \in Q_\exists$, then there is only one descendant $e \cdot 0 \in N$ of e and a transition $q \xrightarrow[\overline{\sigma},\beta]{\sigma} \overline{q} \in \delta$ such that: if $\beta = R$, then $i < n$ and $\nu(e \cdot 0) = a_1 \cdots a_{i-1} \overline{\sigma} (\overline{q}, a_{i+1}) \cdots, a_n$; while if $\beta = L$, then $i > 1$ and $\nu(e \cdot 0) = a_1 \cdots (\overline{q}, a_{i-1}) \overline{\sigma} \cdots a_n$;
  3. if $\nu(e) = a_1 \cdots a_{i-1} (q, \sigma) a_{i+1} \cdots a_n$ and $q \in Q_\forall$, then there are exactly $|Out(q, \sigma)|$ descendants of e and, for each $0 \leq j \leq |Out(q, \sigma)| - 1$, if $Out(q, \sigma, j) = q \xrightarrow[\overline{\sigma_j},\beta_j]{\sigma} \overline{q}$, then $\beta_j = R$ implies $i < n$ and $\nu(e \cdot j) = a_1 \cdots a_{i-1} \overline{\sigma_j} (\overline{q_j}, a_{i+1}) \cdots a_n$, while $\beta = L$ implies $i > 1$ and $\nu(e \cdot j) = a_1 \cdots (\overline{q}, a_{i-1}) \overline{\sigma_j} \cdots a_n$;
  4. a node $e \in N$ is a leaf iff $\nu(e) = a_1 \cdots a_{i-1} (q, \sigma) a_{i+1} \cdots a_n$ for some $q \in Q_F$.

Construction for the lower bound Given an ATM $\mathcal{M}$ and an input sequence ξ, we define an L-EPTA $\mathcal{E}_{\mathcal{M},\xi}$ that accepts a timed sequence ρ if and only if ρ corresponds to the encoding of an accepting run of $\mathcal{M}$ on the input w. Recall that an accepting run of $\mathcal{M}$ is a tree whose paths (from the root to a leaf) end in a final (accepting) configuration. The intuitive idea is that, following a pre-order traversal of the computation tree of the ATM, the automaton $\mathcal{E}_{\mathcal{M},\xi}$ checks that every computation path in the tree ends with an accepting configuration. In order to do this, $\mathcal{E}_{\mathcal{M},\xi}$ exploits the current clock valuation to encode the symbols of two adjacent configuration along a path and uses the store/restore mechanism of L-EPTA to simulate the alternating behavior of the ATM. In particular, while traversing the tree, $\mathcal{E}_{\mathcal{M},\xi}$ performs the following operations: *(i)* after reading two adjacent configurations along a path, recorded in its current clock valuation, it checks that they correspond to a legitimate move of the ATM; *(ii)* on reading a universal configuration, it stores as many copy of that configuration (using Store operations) as the number of the expected children (according to the alternation degree of that configuration), except for the first one that is immediately processed; and *(iii)* on reaching a final configuration, it backtracks along the tree by restoring (by means of a Restore operation) the last universal configuration encountered during the downward traversal, so as to proceed and inspect the remaining computation paths exiting from that configuration.

Since the computation tree of an ATM is a tree and the input of an L-EPTA is a sequence, we need to provide an encoding of accepting computation trees of $\mathcal{M}$ into suitable timed words. Let $\hat{\Sigma} = \Sigma \cup \{\sigma_0, \sharp\} \cup (Q \times \Sigma)$ be the input alphabet of $\mathcal{E}_{\mathcal{M},\xi}$, where the symbols σ_0 and $\sharp$ are used as separators, while symbols in $(Q \times \Sigma)$ intuitively encode the current control state and the tape symbol on which the head of the ATM is currently positioned.

Definition 3 *(Configuration encoding).* Let $a = a_1 \cdots a_n$ be a configuration of the ATM and let the k symbols in $\hat{\Sigma}$ be ordered from 1 to k (i.e., $\hat{\Sigma} = \{\sigma_1, \ldots, \sigma_k\}$). A *configuration encoding* of a is a timed sequence of the form

$$\alpha = (a_1, t_1)(\sigma_0, t_2)(a_2, t_3)(\sigma_0, t_4) \cdots (a_n, t_{2n-1})(\sigma_0, t_{2n})$$

where the following conditions hold:

1. $t_{2(i+1)} - t_{2i} = k + 1$, for each $1 \leq i < n$;
2. $t_{2i} - t_{2i-1} = j$ iff $a_i = \sigma_j$, for each $1 \leq i \leq n$. □

Note that symbol σ_0 is used to separate any two configuration symbols $a_i, a_{i+1} \in \Sigma \cup (\Sigma \times Q)$ adjacent within the configuration. Condition 1 requires that two consecutive separators σ_0 have a fixed time-distance (i.e., $t_{2(i+1)} - t_{2i}$) equal to $k + 1$. Condition 2, instead, ensures that the time distance between the i-th alphabet symbol $a_i = \sigma_j$ in the sequence a and the next separator σ_0 (i.e., $t_{2i} - t_{2i-1}$) is equal to the position j of the symbol in alphabet $\hat{\Sigma}$. Note that the total time of any configuration encoding is equal to $n(k + 1)$.

The encoding of the tree, then, corresponds to the sequence of (encodings of) configurations obtained following a prefix-order traversal of the tree. The ordering of the children of a universal configuration, whose current state is q and whose head is positioned on a symbol σ, is induced by the assumed ordering of the set $Out(q, \sigma)$. For technical reasons, we assume that the configurations in the resulting timed word are separated by a suitable number of occurrences of the symbol $\sharp$. These occurrences of symbol $\sharp$ are essentially used to allow $\mathcal{E}$ to perform consecutive transitions without any advance in time, therefore, the time delay to read each symbol $\sharp$ is required to be 0. The number of $\sharp$ symbols separating a pair of consecutive configurations depends upon the type of the current state of the first configuration in the pair. Let $\alpha_j \cdot (\sharp, t_{\sharp,j})^i \cdot \alpha_{j+1}$ be (the encoding of) two consecutive configuration, where $t_{\sharp,j}$ is equal to the timestamp of the last separation symbol σ_0 of the first configuration α_j (hence, all the separators $\sharp$ between two consecutive configurations occur in the encoding at the same time). If the current state q_j in α_j belongs to $Q_\exists$, then $i = 1$ (we have only one occurrence of $\sharp$ used as separator); if $q_j \in Q_\forall$, then $i = |Out(q, \sigma)|$ (we have one occurrence of $\sharp$ used as separator and $|Out(q, \sigma)| - 1$ occurrences of $\sharp$ used to trigger a Store transition in the simulating L-EPTA $\mathcal{E}$, one for each expected descendant of the considered configuration except one); finally, if $q_j \in Q_F$, then $i = 2$ (intuitively, one occurrence of $\sharp$ is used as separator and one $\sharp$ is used to trigger a Restore transition). Since the configurations of the tree are scanned following a pre-order traversal, we have to exploit the control and evaluation stacks of the L-EPTA $\mathcal{E}$ in order to suitably pair source and target configurations.

Definition 4 *(Run encoding).* Let $\mathbf{a} = a_1, a_2, \ldots, a_h$ be a sequence of configurations of an ATM $\mathcal{M}$. The *configuration sequence encoding* of $\mathbf{a}$ is the timed sequence

$$\overline{\alpha} = \alpha_1 \cdot (\sharp, t_{\sharp,1})^{i_1} \cdot \alpha_2 \cdot (\sharp, t_{\sharp,2})^{i_2} \cdots (\sharp, t_{\sharp,h-1})^{i_{h-1}} \cdot \alpha_h \cdot (\sharp, t_{\sharp,h})^{i_h}$$

where, for $1 \leq r \leq h$, the following conditions hold:

a) α_r is a configuration encoding of a_r;
b) $t_{\sharp,r} = n(k + 1)r$;
c) $i_r = 1$ if a_r is an existential configuration, $i_r = 2$ if a_r is an accepting configuration, and $i_r = l$ if a_r is a universal configuration with l children.

If, in addition, $a_1, a_2, \ldots, a_h$ is the ordering induced by a pre-order traversal of the configurations in a run $\mathcal{T}_{\mathcal{M},\xi}$ of $\mathcal{M}$ on an input sequence $\xi \in \Sigma^*$, then $\overline{\alpha}$ is called the *run encoding* of $\mathcal{T}_{\mathcal{M},\xi}$. □

Note that, by condition *b)*, all symbols $\sharp$ that separate two consecutive configuration encodings occur at the same time. Moreover, the time distance between the last occurrence of the separator symbol σ_0 within a configuration encoding and its first occurrence within the subsequent one is $(k + 1)$, thus lifting validity of Condition 1 of Definition 3 to the entire run encoding.

In order for the L-EPTA $\mathcal{E}$ to simulate the ATM, it has to be able to read two adjacent configurations in the tree and to check that they correspond to a legitimate move of the ATM. To check that a pair of configurations corresponds to a move, say the right move $q \xrightarrow[\overline{\sigma}, R]{\sigma} \overline{q}$, of the ATM and assuming that in the source configuration a symbol $\hat{\sigma} \in (\Sigma \times Q)$ is at position i (i.e., the head of $\mathcal{M}$ is at position i) and a symbol σ' at position $i + 1$, then we need to check that: *(i)* $\hat{\sigma} = (q, \sigma)$; *(ii)* the symbol $\overline{\sigma}$ occurs at position i in the target configuration; and *(iii)* some symbol σ' occurs at position $i + 1$ in source configuration; *(iv)* symbol $(\hat{q}, \sigma')$ is at position $i + 1$ in target configuration; and *(v)* the symbols occurring at any position z different from i and $i + 1$ must be equal in the two configurations. In other words, the automaton has to be able to test for equality of symbols at the same position in the two configurations and to check that a given position in a configuration contains a specific symbol. The following lemma provides a fundamental property of the encoding provided in Definition 4, and stipulates that each of the five conditions above can be simulated by suitable checks on the timestamps of the run encoding.

Lemma 1. *Let* $\overline{\alpha} = \alpha_1 \cdot (\sharp, t_{\sharp,1})^{i_1} \cdot \alpha_2 \cdot (\sharp, t_{\sharp,2})^{i_2} \cdots (\sharp, t_{\sharp,h-1})^{i_{h-1}} \cdot \alpha_h \cdot (\sharp, t_{\sharp,h})^{i_h}$ *be the configuration sequence encoding of the sequence of configurations* $\mathbf{a} = a_1, a_2, \ldots, a_h$ *and*

$$\alpha_r = (a_1, t_1)(\sigma_0, t_2)(a_2, t_3)(\sigma_0, t_4) \cdots (a_n, t_{2n-1})(\sigma_0, t_{2n})$$

$$\alpha_{r+1} = (a_{n+1}, t_{2n+1})(\sigma_0, t_{2(n+1)})(a_{n+2}, t_{2n+3})(\sigma_0, t_{2(n+2)}) \cdots (a_{2n}, t_{4n-1})(\sigma_0, t_{4n})$$

be two consecutive configuration encodings in $\overline{\alpha}$ *and* $1 \leq r < h$. *Then the pair* (α_r, α_{r+1}) *is a right move induced by an ATM transition* $q \xrightarrow[\overline{\sigma}, R]{\sigma} \overline{q}$ *if and only if, for some* $1 \leq i < n$, *the following conditions hold:*

1. $t_{2(n+z)-1} - t_{2z-1} = n(k+1)$, *for all* $1 \leq z \leq n$ *and* $i \neq z \neq i+1$;
2. $t_{4n} - t_{2i-1} = (2n-i)(k+1) + j$, *with* $\sigma_j = (q, \sigma)$;
3. $t_{4n} - t_{2(n+i)-1} = (n-i)(k+1) + w$, *with* $\sigma_w = \overline{\sigma}$;
4. $t_{4n} - t_{2(i+1)-1} = (2n-(i+1))(k+1) + s$, *for some* $1 \leq s \leq k$;
5. $t_{4n} - t_{2(n+(i+1))-1} = (n-(i+1))(k+1) + l$, *with* $\sigma_l = (\overline{q}, \sigma_s)$.

Similarly, (α_r, α_{r+1}) *is a left move induced by* $q \xrightarrow[\overline{\sigma}, L]{\sigma} \overline{q}$ *if and only if, for some* $1 < i \leq n$,

1. $t_{2(n+z)-1} - t_{2z-1} = n(k+1)$, *for all* $1 \leq z \leq n$ *and* $i-1 \neq z \neq i$;
2. $t_{4n} - t_{2i-1} = (2n-i)(k+1) + j$, *with* $\sigma_j = (q, \sigma)$;
3. $t_{4n} - t_{2(n+i)-1} = (n-i)(k+1) + w$, *with* $\sigma_w = \overline{\sigma}$;
4. $t_{4n} - t_{2(i-1)-1} = (2n-(i-1))(k+1) + s$, *for some* $1 \leq s \leq k$;
5. $t_{4n} - t_{2(n+(i-1))-1} = (n-(i-1))(k+1) + l$, *with* $\sigma_l = (\overline{q}, \sigma_s)$.

Proof. Let us consider the case of a right move induced by transition $t = q \xrightarrow[\overline{\sigma}, R]{\sigma} \overline{q}$, the other case being similar. For (a_r, a_{r+1}) to be a move induced by t, it must hold that for some $1 \leq i \leq n$: *(i)* $a_z = a_{n+z}$, for all positions z different from i and $i+1$; *(ii)* $a_i = (q, \sigma)$; *(iii)* $\overline{\sigma} = a_{n+i}$; *(iv)* $a_{(i+1)} = \sigma'$, for some σ'; and *(v)* $a_{n+(i+1)} = (\overline{q}, \sigma')$. Assume that j (resp., w and s) is the index of symbol (q, σ) (resp., $\overline{\sigma}$ and σ') in the ordering of $\hat{\Sigma}$.

Each condition of the lemma can be proved equivalent to one of the above conditions. Let us consider condition (i) first. From Definition 3 and Definition 4 (Condition *b)*), we have that $t_{4n} = n(k+1)(r+1)$ and that the time distance between any two consecutive separator symbols σ_0 in $\overline{\alpha}$ is precisely $(k+1)$. This gives us $t_{2(n+z)} - t_{2z} = n(k+1)$. Moreover, by Definition 3, the index (in $\hat{\Sigma}$) of a symbol occurring at any position z in a configuration encoding corresponds to the time difference $t_{2z} - t_{2z-1}$. Therefore, for any $1 \leq z \leq n$ and $i \neq z \neq i+1$, $a_z = a_{n+z}$ if and only if $t_{2z} - t_{2z-1} = t_{2(n+z)} - t_{2(n+z)-1}$ or, equivalently, if and only if $t_{2(n+z)-1} - t_{2z-1} = t_{2(n+z)} - t_{2z} = n(k+1)$. Consider now condition *(ii)*. By Definition 3 (Condition 2), $a_i = (q, \sigma) = \sigma_j$ if and only if $t_{2i} - t_{2i-1} = j$. In addition, we have that $t_{4n} - t_{2i} = (2n-i)(k+1)$. Hence, $t_{4n} - t_{2i-1} = (2n-i)(k+1) + j$, as required. An analogous argument proves the equivalence of the remaining conditions of the lemma to conditions *(iii)*, *(iv)* and *(v)*. □

We can easily define an L-EPTA automaton that reads a run encoding and checks pairs of adjacent configurations encodings (the source and the target configuration for a move), by keeping track of their content by means of $2n+1$ clocks $x_0, x_1^p, \ldots x_n^p$, with $p \in \{0, 1\}$. Clock x_0 is used to count the elapsing of time between two separator symbols σ_0 and is reset when it reaches value $(k+1)$ and an occurrence of σ_0 is read. For each position $1 \leq i \leq n$ in a configuration encoding, we use two clocks x_i^0 and x_i^1. Each such pair of clocks stores the symbols read in the i-th position in the two adjacent configurations. In order to reuse clocks, x_i^0 and x_i^1 are used in an alternating way to encode the source and the target configurations, respectively. In other words, after reading each configuration, the parity switches value.

For instance, let us consider the two consecutive configuration of Lemma 1, and assume that α_r is scanned with parity $p \in \{0, 1\}$ and α_{r+1} with parity $\overline{p} = 1 - p$. When reading the i-th element (with $1 \leq i \leq n$) of the first configuration, clock x_i^p is reset. When reading the i-th symbol of α_{r+1}, clock $x_i^{\overline{p}}$ is reset. Therefore, at time t_{4n}, when the two configurations have been scanned, we have that $x_i^p = t_{4n} - t_{2i-1}$ and $x_i^{\overline{p}} = t_{4n} - t_{2(n+i)-1}$. As a consequence, $x_i^p - x_i^{\overline{p}} = t_{2(n+i)-1} - t_{2i-1}$. By Lemma 1, we conclude that the clock constraint $x_i^p - x_i^{\overline{p}} = n(k+1)$ holds *iff* the two symbols at position i in the two configurations are equal. Moreover, since $x_i^p = t_{4n} - t_{2i-1}$, by Lemma 1 the symbol at position i in a_r is equal to the j-th symbol in the ordering of the alphabet $\Sigma \cup (\Sigma \times Q)$ *iff* the clock constraint $x_i^p = (2n-i)(k+1) + j$ holds. By a similar argument, the symbol at position i in a_{r+1} is equal to the j-th symbol in $\Sigma \cup (\Sigma \times Q)$ *iff* the clock constraint $x_i^{\overline{p}} = (n-i)(k+1) + j$ holds. By the above observations, all the checks needed to verify that two adjacent configurations correspond to a legal move can be encoded by a suitable conjunction of clock constraints of the forms just described (see proof of Theorem 8 in the appendix for a complete account of these clock constraints).

Algorithm: CHECK-COMPUTATION-TREE().

```
control_stack := ⟨⊥⟩;
valuation_stack := ⟨v_0⟩;
parity := 0;
expected := q_0;
(current, symbol) := SCAN_CONFIGURATION(parity,expected);
READ(♯);
while control_stack ≠ ϵ do
    source := current;
    if current ∈ Q_F then
        READ(♯);
        (parity, source, target, clocks) := RESTORE();
        expected := target;
    else if current ∈ Q_∀ then
        for i = |Out(current, symbol)| downto 2 do
            READ(♯);
            STORE(parity, current, TARGET(OUT(current,symbol,i)), clocks);
        expected := TARGET(OUT(current,symbol,1));
    else        /* current ∈ Q_∃                                              */
        expected := '_';
    if control_stack ≠ ϵ then
        parity := 1 - parity;
        (current, symbol) := SCAN_CONFIGURATION(parity,expected);
        READ(♯);
        CHECK_TRANSITION(parity, source, current, clocks);
```

Fig. 14. The algorithm checking a computation tree of $\mathcal{M}$.

Fig. 14 reports the pseudocode of an algorithm summarizing the behavior of the automaton $\mathcal{E}_{\mathcal{M},\xi}$ simulating the ATM $\mathcal{M}$ on ξ. The algorithm maintains two stacks, *control_stack* and *valuation_stack*, to store control information and clock values, respectively.

The command SCAN_CONFIGURATION*(parity,expected)* scans a configuration with the given *parity*, stopping at the first ♯ symbol, and returns its control state and current symbol. During scanning, it also resets the clocks, according to the description given above, and checks that the read sequence is indeed a correct configuration encoding: exactly n symbols in $\hat{\Sigma}$ occur, each followed by the separator σ_0 at the expected time, and precisely one of them is a pair in $Q \times \Sigma$ (the *current* state and *symbol* on the tape). In addition, it checks that, if *expected* $\neq$ '_', then the control state in *expected* corresponds to the *current* state that appears in the configuration.

When the current configuration is universal, the algorithm performs $k-1$ push operations on the stacks, one for each of the k transitions over the symbol *symbol* outgoing from state *current* in decreasing order, except for the first one. In particular, the operation STORE*(parity, current, target, clocks)* stores in the *control_stack* the triple $\langle parity, current, target\rangle$, containing the parity of the current configuration, its (universal) control state, and the target control state of a transition from *current* to *target*. In addition, the values of the *clocks*, encoding the current configuration, are stored in the *valuation_stack*. Notice that each Store operation is triggered by the special symbol ♯. The control state of the first outgoing transition from *current* is, then, assigned as the *expected* state for the next configuration scan.

When the current configuration is final, the algorithm restores the last universal configuration, so as to proceed on checking the next computation branch. The call to RESTORE*()* pops from the control stack the triple $\langle parity, source, target\rangle$ that identifies the next move of the run exiting from state *source* that needs to be checked, and restores from the valuation stack the values of the *clocks*, encoding the source configuration with *source* as control state. The *target* control state of the move is then assigned as the *expected* state for the next configuration scan.

If the current configuration is existential, instead, no specific action is necessary and no specific control state is *expected* at the next scan.

If at least two configurations have been scanned, CHECK_TRANSITION*(parity, source, current, clocks)* checks that the *source* and *target* configurations are compatible with a move of $\mathcal{M}$. This is done by checking satisfaction of clock constraints based on Lemma 1 and explained above. We assume that the procedure stops as soon as a check fails. Notice that at the beginning of every new configuration scan, the parity is complemented by the assignment *parity := 1 - parity*. The translation of this algorithm into an L-EPTA is straightforward and is reported in the appendix. As a consequence we obtain the following result.

Theorem 8. *The emptiness problem and the reachability problem for* L-EPTA *are* ***EXPTIME****-hard.*

6.3. The reachability problem for I-TRSM

We conclude the section by considering the complexity of the reachability problem for the smallest subclass I-TRSM, showing that the problem is **PSPACE**-complete, as in the case of TAs. Since every TA is a special case of I-TRSM with no boxes, **PSPACE**-hardness follows directly from the **PSPACE**-hardness of reachability in TAs. To show that the problem is in **PSPACE**, we provide an encoding, loosely inspired by the construction in [5], which reduces the reachability problem in an I-TRSM $\mathcal{T}$ to a polynomially bounded number of reachability problems in a suitable set of TAs, each having size polynomial in the size of some component of $\mathcal{T}$.

Intuitions The idea underlying the following construction is that, given a run of a I-TRSM $\mathcal{T}$, it is possible to abstract away all the sub-runs bounded by matching pairs of Call and Return transitions by means of suitable Reset transitions. This abstraction can be performed by augmenting $\mathcal{T}$ with *summary nodes* of the form $(en, ex)_b$, which can be used during the reachability analysis to skip the invocation of a component b, provided that the exit node ex of the component is actually reachable from the corresponding entry node en of the same component.

We start by computing reachability from entry nodes to exit nodes, considering only internal transitions (thus, avoiding Call transitions). For two nodes u, v of the same machine A_i of $\mathcal{T}$, we say that u is *locally reachable* from v if it is reachable following only Reset transitions of A_i. Notice that any Call transition in an I-TRSM resets all the clocks when entering the invoked component. Hence, the only relevant local reachability problem from an entry to an exit node in the invoked component is the one which assumes the clock valuation set to zero at the entry node. Therefore, this form of reachability corresponds to a standard reachability problem for Timed Automata, starting from an initial valuation of clocks. Observe that this is the main difference with respect to L-TRSM, where different clock valuations for the entry node have to be considered, thus forcing to explicitly take into account regions of entry nodes for the called machines. If an exit node ex can be reached from an entry node en within a machine, then we add a suitable *summary transition*, which is itself a Reset transition and connects the source node of the Call transition to the corresponding *summary node* for the called machine, abstracting away the internal activity of that machine. Essentially, by means of summary transitions we can avoid component invocations and reduce reachability within an I-TRSM to the simpler local reachability problem.

Given an I-TRSM $\mathcal{T}$ with n components, we start by building a set of n Timed Automata, one for each machine of $\mathcal{T}$. The Timed Automaton $\mathcal{A}_i^{(0)}$ for machine A_i of $\mathcal{T}$ is obtained from A_i by: *(i)* replacing each box $b \in B_i$ with the summary nodes of the form $(en, ex)_b$ (one for each pair of entry node (b, en) and exit node (b, ex) of that box); and *(ii)* deleting all the Call transitions. A summary transition from a state of a Timed Automaton to a summary node $(en, ex)_b$ is then added whenever there is a Call transition entering (b, en) and node ex is locally reachable from en in the Timed Automaton, corresponding to the machine associated with box b.

Summary transitions are computed iteratively. At each iteration, the set of Timed Automata is enriched with new summary transitions. This is done by solving, for each summary node, a local reachability problem within the corresponding Timed Automaton. The process iterates until no new summary transitions can be added. Since the number of summary nodes and transitions in each δ_i is finite, the procedure always terminates.

This gives us a fixpoint procedure, which builds a sequence $\{\mathcal{T}^{(k)}\}_{k\geq 0}$ of tuples of Timed Automata, according to the above intuition. Since the number of summary nodes and transitions is clearly polynomial in the number of transitions of $\mathcal{T}$ (indeed, bounded by the product of the total number Call transitions and the total number of exit nodes), and reachability in a Timed Automaton is known to be solved in polynomial space [2], we can compute the fixpoint using only polynomial space.

The reachability procedure The procedure starts from an I-TRSM $\mathcal{T} = \langle A_1, \cdots, A_n, X\rangle$. For each machine A_i, with $1 \leq i \leq n$, we first compute the set of summary nodes $SumSt_i = \{(en, ex)_b : b \in B_i, (b, en) \in Calls_i, (b, ex) \in Retns_i\}$. Let $target(t)$, for $t \in \delta_i$, denote the target node of the transition t and $Exits_i = \{(ex, t) : ex \in Ex_i, t \in \delta_i, ex = target(t)\}$ the set of *exit pairs* for component A_i, each composed by an exit node $ex \in Ex_i$ and the identifier t of a transition entering node ex.

The initial tuple $\mathcal{T}^{(0)} = \langle \mathcal{A}_1^{(0)}, \cdots, \mathcal{A}_n^{(0)}\rangle$ of Timed Automata is built as follows. The automaton $\mathcal{A}_i^{(0)}$ for the component A_i, with $1 \leq i \leq n$, contains: a state for each node in $N_i \setminus Ex_i$; a state for each exit pair in $Exits_i$, replacing the exit nodes of A_i; and the summary nodes for component A_i, replacing the boxes of A_i. The transition relation $\delta_i^{(0)}$ contains all the Reset transitions in δ_i connecting nodes in $N_i \setminus Ex_i$, where the return guard φ_s and the restore set are removed. In addition, for each Return transition t, leading from a node $q \in N_i \setminus Ex_i$ to a node $ex \in Ex_i$ and with current guard φ_c, we add a transition from q to (ex, t), with guard φ_c. Finally, for every Reset transition $t \in \delta_i$ from a return node $(b, ex) \in Retns_i$ to a node in $q \in N_i$, we add a transition from the summary node $(en, ex)_b$ (for any en with $(b, en) \in Calls_i$) to q, if $q \notin Ex_i$, and to (ex, t), if $q = ex \in Ex_i$.

In this way, each TA $\mathcal{A}_j^{(0)}$ contains only the Reset transitions of A_j and one reset transition for each Return transition. The return constraint φ_s of each Return transition t entering the exit node ex is recorded in the exit pair (ex, t). Moreover, Call transitions are not included and summary nodes have no incoming transitions, therefore no summary node is reachable in $\mathcal{A}_j^{(0)}$ from any initial state (no summary node is an initial state of the Timed Automaton either). If a state (ex, t) is reachable from en in $\mathcal{A}_j^{(0)}$, then node ex is reachable from en in A_j along a run ending in transition t and performing no

Call transition. Assume, now, that there is a Call transition $t' = q \xrightarrow{\sigma,\varphi,True}_{\emptyset,X} (b, en)$ in A_i (with $Y_i(b) = j$), a run leading to q in A_i and that state (ex, t) is reachable from en in $\mathcal{A}_j^{(0)}$. The semantics of an I-TRSM ensures that if there is a clock valuation $\mathbf{v}$ at q satisfying φ (hence, triggering t') and, in addition, $\mathbf{v}$ satisfies the return constraint φ_s of the Return transition t, thus ensuring that t is executable at ex in A_j, then there is also a run from q to (b, ex). Therefore, we can abstract away the computation performed within A_j into a single Reset transition $q \xrightarrow[r]{\sigma,\varphi\wedge\varphi_s} (en, ex)_b$, from q to the summary node $(en, ex)_b$. This Reset transition is guarded by the conjunction of the current constraint φ of t' and the return constraint φ_s of t, and has the same reset set r as t.

To compute the remaining elements of the sequence $\{\mathcal{T}^{(k)}\}_{k\geq 0}$ we proceed as follows. Let $Treach_A(q, q')$ be the standard reachability procedure that, given two control states $q, q' \in Q$ of a Timed Automaton $A = \langle Q, Q_0, \Sigma, X, Q_F, \delta\rangle$, decides whether, for some clock valuation $\mathbf{v}$, state $(q', \mathbf{v})$ can be reached from $(q, \mathbf{v}_0)$ in the LTS of A, where $\mathbf{v}_0$ is the initial clock valuation. Recall that this function can be computed using polynomial space (see [2]), by exploiting the standard regionalization technique for Timed Automata described in Section 6.2.

At each iteration $k \geq 1$ the procedure computes the tuple $\mathcal{T}^{(k)} = \langle \mathcal{A}_1^{(k)}, \ldots, \mathcal{A}_n^{(k)}\rangle$ as follows. First, for each $b \in B_i$ with $j = Y_i(b)$, and $(b, en) \in Calls_i$, the set

$$ReachEx_j^{(k-1)}(en) = \left\{(ex, t) \in Q_j : Treach_{\mathcal{A}_j^{(k-1)}}(en, (ex, t)) \text{ holds}\right\}$$

of states of the form (ex, t) that are reachable from en in the LTS of the Timed Automaton $\mathcal{A}_j^{(k-1)}$ is computed.

Then, the set $SumTr(i, k)$ of summary transitions for machine A_i is collected as follows. If the pair $(ex, t) \in ReachEx_j^{(k-1)}(en)$, then we abstract the run leading from (b, en) to (b, ex) by adding to $\delta_i^{(k)}$ the transition $q \xrightarrow[r]{\sigma,\varphi\wedge\varphi_s} (en, ex)_b$ where: (i) φ_s is the return constraint of the Return transition t; and (ii) either $q \in N_i$ and $q \xrightarrow[\emptyset,X]{\sigma',\varphi,True} (b, en) \in \delta_i$, or $q = (en', ex')_{b'}$ and $(b', ex') \xrightarrow[\emptyset,X]{\sigma',\varphi,True} (b, en) \in \delta_i$.

The procedure described above is formalized by the following definition:

Definition 5 *(*I-TRSM *encoding).* Let $\mathcal{T} = \langle A_1, \cdots, A_n, X\rangle$ be an I-TRSM. The sequence $\{\mathcal{T}^{(k)}\}_{k\geq 0}$ of tuples of Timed Automata is inductively defined as follows:

- $\mathcal{T}^{(0)} = \langle A_1^{(0)}, \cdots, A_n^{(0)}\rangle$, where, for each $1 \leq i \leq n$, the automaton $\mathcal{A}_i^{(0)} = \langle Q_i, Q_{i0}, X, \Sigma, Q_{iF}, \delta_i^{(0)}\rangle$, $Q_i = (N_i \setminus Ex_i) \cup SumSt_i \cup Exits_i$, $Q_{i0} = En_i$ and $Q_{iF} = Exits_i$ and

$$\begin{aligned}\delta_i^{(0)} = &\left\{q \xrightarrow[r]{\sigma,\varphi} q' : q \xrightarrow[\emptyset,r]{\sigma,\varphi,True} q' \in \delta_i, q, q' \in (N_i \setminus Ex_i)\right\} \cup\\ &\left\{q \xrightarrow[r]{\sigma,\varphi_c} (ex, t) : t = q \xrightarrow[X,r]{\sigma,\varphi_c,\varphi_s} ex \in \delta_i, q \in (N_i \setminus Ex_i)\right\} \cup\\ &\left\{(en, ex)_b \xrightarrow[r]{\sigma,\varphi_c} (ex', t) : t = (b, ex) \xrightarrow[X,r]{\sigma,\varphi_c,\varphi_s} ex' \in \delta_i\right\} \cup\\ &\left\{(en, ex)_b \xrightarrow[r]{\sigma,\varphi} q : (b, ex) \xrightarrow[\emptyset,r]{\sigma,\varphi,True} q \in \delta_i, q \in (N_i \setminus Ex_i)\right\}.\end{aligned}$$

- For $k \geq 1$, $\mathcal{T}^{(k)} = \langle A_1^{(k)}, \cdots, A_n^{(k)}\rangle$, where, for each $1 \leq i \leq n$, $\mathcal{A}_i^{(k)} = \langle Q_i, Q_{i0}, X, \Sigma, Q_{iF}, \delta_i^{(k)}\rangle$ is obtained from $\mathcal{A}^{(k-1)}$ by setting $\delta_i^{(k)} = \delta_i^{(k-1)} \cup SumTr(i, k)$ and retaining all of its other components. The set of transition $SumTr(i, k)$ is defined as follows:

$$\begin{aligned}SumTr(i, k) = &\left\{q \xrightarrow[r]{\sigma,\varphi\wedge\varphi_s} (en, ex)_b : q \xrightarrow[\emptyset,X]{\sigma,\varphi,True} (b, en) \in \delta_i \text{ and } (ex, t) \in ReachEx_j^{(k-1)}(en),\right.\\ &\left.\text{for some } t = q' \xrightarrow[X,r]{\sigma,\varphi_c,\varphi_s} ex \in \delta_j\right\} \bigcup\\ &\left\{(en', ex')_{b'} \xrightarrow[r]{\sigma,\varphi\wedge\varphi_s} (en, ex)_b : (b', ex') \xrightarrow[\emptyset,X]{\sigma,\varphi,True} (b, en) \in \delta_i \text{ and}\right.\\ &\left.(ex, t) \in ReachEx_j^{(k-1)}(en), \text{for some } t = q' \xrightarrow[X,r]{\sigma,\varphi_c,\varphi_s} ex \in \delta_j\right\}.\end{aligned}$$

□

The above definition induces a procedure that builds a sequence of tuples of Timed Automata. Once the procedure reaches a fixpoint, say at iteration $\hat{k}$, we obtain the tuple $\mathcal{T}^{\hat{k}} = \langle \mathcal{A}_1^{(\hat{k})}, \cdots, \mathcal{A}_n^{(\hat{k})} \rangle$ of Timed Automata, from which we can build a new I-TRSM $\mathcal{T}' = \langle A'_{init}, A'_1, \cdots, A'_n \rangle$, called *summarized* I-TRSM. This is done by merging the original I-TRSM $\mathcal{T}$ with the reachability information encoded by the summary nodes and the transitions in the fixpoint automata, and adding a new machine A'_{init}. The additional machine A'_{init} will play the role of the external calling environment, which is modeled by box b_{init} in the invocation history of global states in the semantics of TRSMs. Formally,

Definition 6 *(Summarized* I-TRSM*).* Given an I-TRSM $\mathcal{T} = \langle A_1, \cdots, A_n \rangle$, the *summarized* I-TRSM $\mathcal{T}' = \langle A'_{init}, A'_1, \cdots, A'_n \rangle$ contains, for all $1 \leq i \leq n$, the machine $A'_i = \langle N'_i \cup B_i, Y_i, En_i, Ex_i, \delta'_i \rangle$, where $N'_i = N_i \cup SumSt_i$ and

$$\delta'_i = \delta_i \cup SumTr(i, \hat{k}) \cup \left\{ (en, ex)_b \xrightarrow[\emptyset, X]{\varphi, True} (b', en') : (b, ex) \xrightarrow[\emptyset, X]{\varphi, True} (b', en') \text{ and } (b', en') \in Calls_i \right\}$$

$A'_{init} = \langle N'_{init} \cup B'_{init}, Y'_{init}, En'_{init}, Ex'_{init}, \delta'_{init} \rangle$, where

- $N'_{init} = \{(en, ex)_b : en \in En_1 \text{ and } ex \in Ex_1\}$,
- $En'_{init} = \{en_0\}$,
- $B'_{init} = Ex'_{init} = Y'_{init} = \emptyset$,
- the transition function δ'_{init} contains the transitions in the set

$$\left\{ en_0 \xrightarrow[\emptyset, \emptyset]{\sigma, True, True} (en, ex)_b : (ex, t) \in ReachEx_1^{(k-1)}(en), \text{ for } t = q \xrightarrow[X, r]{\sigma, \varphi_c, \varphi_s} ex \in \delta_1 \text{ and } \mathbf{v}_0 \in \varphi_s \right\}. \qquad \square$$

The new machine A'_{init} has a single entry node en_0, a summary node of the form $(en, ex)_b$ for each entry node en and exit node ex of the initial component A_1. It has neither boxes nor exit nodes. The transitions in A'_{init} connect the entry node en_0 to each summary node $(en, ex)_b$, such that an exit pair (ex, t) is reachable from en in the fixpoint Timed Automaton $\mathcal{A}_1^{(\hat{k})}$ for component A_1. Each such transition summarizes an entire terminating computation of $\mathcal{T}$, i.e. a run from an entry node to an exit node of A_1, with a single Reset transitions in A'_{init}.

Correctness of the procedure By observing that the summarized $\mathcal{T}'$ is simply $\mathcal{T}$ augmented with additional transitions that abstract actual runs of $\mathcal{T}$, it is immediate to see that $\mathcal{T}$ and $\mathcal{T}'$ are indeed equivalent w.r.t. global states reachability.

Lemma 2. *A global state gs is reachable from an initial global state gs_0 in $\mathcal{T}$ if and only if gs is reachable from gs_0 in the summarized* I-TRSM $\mathcal{T}'$.

A more interesting property of $\mathcal{T}'$ is that reachability of two global states of $\mathcal{T}'$, both having control states of the same machine j and the same control and valuation prefixes of length 1 (hence an empty invocation history), collapses to local reachability in a Timed Automaton. Concretely, given two global states of the form $gs = \langle b_1, u, \mathbf{v}_1, \mathbf{v} \rangle$ and $gs' = \langle b_1, u', \mathbf{v}_1, \mathbf{v}' \rangle$, reachability of gs' from gs reduces to a corresponding reachability in the fixpoint Timed Automaton $\mathcal{A}_j^{(\hat{k})}$. The property is made precise by the following lemma (see the appendix for the proof).

Lemma 3. *Let $gs = \langle b_1, u, \mathbf{v}_1, \mathbf{v} \rangle$ be a global state of the summarized* I-TRSM $\mathcal{T}'$*, with $\mathbf{v}, \mathbf{v}_1$ arbitrary valuations, $u \in N'_j$, for some j, and $b \in B'_i$, for i such that $Y'_i(b) = j$. Then,*

1. *the global state $gs' = \langle b_1, u', \mathbf{v}_1, \mathbf{v}' \rangle$, with $u' \in N'_j \setminus Ex'_j$, is reachable from gs iff $\langle u', \mathbf{v}' \rangle$ is reachable from $\langle u, \mathbf{v} \rangle$ in $\mathcal{A}_j^{(\hat{k})}$;*
2. *the global state $gs' = \langle \epsilon, (b_1, ex), \epsilon, \mathbf{v}'_1 \rangle$ is reachable from gs if and only if there exists a Return transition $t = q \xrightarrow[X, r]{\varphi_c, \varphi_s} ex \in \delta'_j$ such that $\mathbf{v}_1 \in \varphi_s$, $\mathbf{v}'_1 = \mathbf{v}_1 \downarrow_r$ and $\langle (ex, t), \mathbf{v}' \rangle$ is reachable from $\langle u, \mathbf{v} \rangle$ in $\mathcal{A}_j^{(\hat{k})}$.*

The previous lemma is crucial, as it implicitly allows us to break the untimed reachability problem in $\mathcal{T}$ down to a sequence of purely local reachability problems in $\mathcal{T}'$. The following lemma, whose proof is given in the appendix, provides the result.

Lemma 4. *Let the* I-TRSM $\mathcal{T}$ *and its summarized* I-TRSM $\mathcal{T}'$ *be given. Let, in addition, $g_0 = \langle b_{init}, en \rangle$ and $g = \langle b_1 \cdots b_s, u \rangle$ be untimed global states of $\mathcal{T}$, with $b_1 = b_{init}$ if $s \geq 1$, and $A_{i_1}, \ldots, A_{i_s}$ be the sequence of the invoked components in the history of g, with $i_1 = 1$ and $Y_{i_{j-1}}(b_j) = i_j$, for $2 \leq j \leq s$. Then, g is reachable from g_0 in $\mathcal{T}$ iff there exists a sequence of entry nodes $en_1, \ldots, en_s$, with $en_j \in En_{i_j}$ such that: for all $1 \leq j \leq s-1$, (b_{j+1}, en_{j+1}) is reachable from en_j in A'_{i_j}, following only Reset transitions except for the last Call transition leading to (b_{j+1}, en_{j+1}), and*

- *either* $u \in N_{i_s}$ *and* u *is locally reachable from* en_s *in* A'_{i_s};
- *or* $u = (b, ex')$ *and there exists a summary node* $(en', ex')_b \in N'_{i_s}$ *which is locally reachable from* en_s *in* A'_{i_s},

where we take $en_s = en_0$ *and* $A'_{i_s} = A'_{init}$ *if* $s = 0$.

Lemma 4 allows us to solve the untimed reachability problem of an arbitrary untimed global state $g = \langle b_1 \cdots b_s, u \rangle$ in $\mathcal{T}$ by executing a sequence of local reachability checks in $\mathcal{T}'$, according to the following non-deterministic procedure. Let $A_{i_1}, \ldots, A_{i_s}$ be the sequence of the invoked components in g, with $i_1 = 1$ and $Y_{i_{j-1}}(b_j) = i_j$, for $2 \leq j \leq s$. We, then, proceed as follows:

(i) guess a sequence of entry nodes $en_1, \ldots, en_s$, with $en_j \in En_{i_j}$;
(ii) check whether, for each $1 \leq j \leq s-1$, (b_{j+1}, en_{j+1}) is locally reachable from en_j in component A'_{i_j};
(iii) check whether u is locally reachable from en_s in component A'_{i_s}.

Lemma 4 ensures that the answer to the reachability problem is positive if and only if all these checks are fulfilled. Finally, by observing that **NPSPACE** = **PSPACE** and that each local reachability check corresponds to a reachability analysis in a Timed Automaton, whose solution only requires polynomial space, we obtain the desired complexity result.

Theorem 9. *The reachability problem for* I-TRSM *is* ***PSPACE****-complete.*

7. Conclusions

In this paper we have introduced and studied Timed Recursive State Machines, a real-time extension of Recursive State Machines, which allows for modeling real-time recursive systems. The distinctive feature of TRSMs is the ability to store a clock valuation when a component is invoked and restore the clock values upon return from the component. Context-free properties involving both the untimed and the timed dimensions can be specified by TRSMs, which cannot be expressed either by Timed Automata or by Pushdown Timed Automata. Though the reachability problem in the general framework is undecidable, we have shown that the problem is still decidable for meaningful subclasses, when suitable restrictions to the store/restore mechanism apply. In particular, the problem is proved to be **EXPTIME**-complete for the class of L-TRSM and **PSPACE**-complete for the class I-TRSM, the same complexity of reachability in standard Timed Automata. Clearly, the decidability results for the reachability problem for I-TRSM and L-TRSM can be generalized to decidability results for the model checking problem w.r.t. untimed properties (e.g., LTL and CTL properties). However, the negative results concerning non-closure under intersection with timed regular languages prevents such a generalization for the model checking of timed properties, such as Timed CTL [4] properties. Therefore, an interesting open problem is whether suitable restrictions of these subclasses can be identified which are closed under intersection with non-trivial classes of timed (regular) languages.

A number of other interesting issues regarding expressiveness are still to be investigated. The paper focuses on the relationship with the most common real-time formalisms, namely Timed Automata and Pushdown Timed Automata. On the other hand, the relationship with other related formalisms needs to be settled. In particular, the undecidability proof for the reachability problem in TRSMs and EPTAs shows that they are able to simulate clock updates (increment and decrement) similar to those provided by Updatable Timed Automata [13]. Also the formalisms of Stopwatch Automata [14], briefly considered in the paper, still requires a deeper comparison. It seems, indeed, that Stopwatch Automata can be simulated by a suitable extension of TRSMs allowing for silent transitions, while the opposite does not seems to hold.

Appendix A. Additional proofs

In the following we report the detailed proofs of the results provided in Sections 5 and 6.

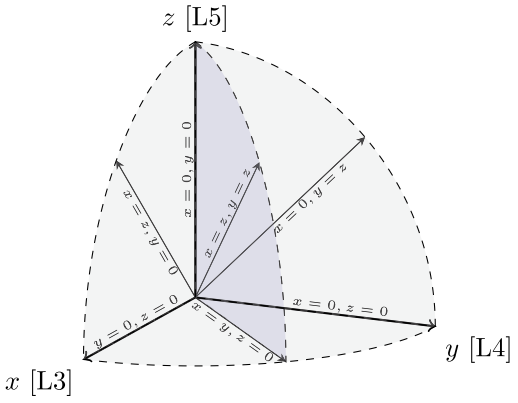
A.1. Results in Section 5

Since the proofs of the following claims are all based on geometric arguments, we shall first recall some basic geometric notions. The *relative interior* of a convex set P, in symbols $rint(P)$, is the interior of P taken in the space corresponding to its affine hull $ahull(P)$. More formally, $x \in rint(P)$ if and only if there is a ball $N_\epsilon(x)$ of radius $\epsilon > 0$ centered in x, such that $N_\epsilon(x) \cap ahull(P) \subseteq P$. For instance, any segment has empty interior in the space $\mathbb{R}^3$, but a non-empty relative interior w.r.t. its affine hull, which is the straight line containing it. Similarly, any square in $\mathbb{R}^3$ has a relative interior w.r.t. its affine hull, which corresponds to the plane containing it. The *affine dimension* of P is the dimension of its affine hull. Hence, the affine dimension of a segment in $\mathbb{R}^3$ is 1, while the affine dimension of a square is 2.

Claim 1. *For every path* π, $\mathcal{L}_\pi \subseteq \mathcal{L}_0$ *if and only if the constraint* φ^c, *guarding the transition reading* c *in* π, *satisfies one of the following condition:*

L1	$x - z = k_1,\ y - z = k_2\ (x - y = k_1 - k_2)$
L2	$x - z = k_1,\ y = k_2$
L3	$y = k_2,\ z = k_3\ (y - z = k_2 - k_3)$
L4	$x = k_1,\ z = k_3,\ (x - z = k_3 - k_1)$
L5	$x = k_1,\ y = k_2,\ (x - y = k_2 - k_1)$
L6	$x = k_1,\ y - z = k_2$
L7	$x - y = k_1,\ z = k_3$

(a) The general schema of the seven straight lines



(b) An instance of the straight lines in Table 15a. Here all the constants are equal to 0.

Fig. 15. The seven straight lines in $\mathbb{R}^3$ identified by equality clock Constraints, for the proof of Claim 1.

1. *there exists a constant* $k \geq 1$ *such that* φ^c *implies both* $x = k$ *and* $y - z = k - 1$*;*
2. *there exists a constant* $0 \leq \hat{k} \leq 1$ *such that* φ^c *implies both* $x - y = \hat{k}$ *and* $z = 1 - \hat{k}$*.*

Proof. The *if* direction is obvious, since whatever condition holds, 1. or 2., necessarily $\Delta_1 + \Delta_3 = 1$.

The proof of the other direction follows a geometric argument. First, notice that the clock constraint φ^c is a conjunction of (linear) difference constraints, hence its solutions form a convex region of points in $\mathbb{R}^3$. Let us denote with $[\![\varphi^c]\!] \subseteq \mathbb{R}^3$ the (convex) set of solutions to φ^c. By considering its possible forms, we prove that either at least one between conditions 1. or 2. above holds, or at least one timed sequence in $\mathcal{L}_\pi$ does not belong to $\mathcal{L}_0$. In the latter case, it will suffice to show that $[\![\varphi^c]\!]$ contains two solutions assigning different values to the clock expression $(x - y) + z$, which corresponds to the quantity $\Delta_1 + \Delta_3$. Let us consider the possible cases:

i) if $[\![\varphi^c]\!] = \emptyset$, then $\mathcal{L}_\pi = \emptyset$ and we have an immediate contradiction.
ii) if $[\![\varphi^c]\!]$ has relative interior of affine dimension 0, then it contains a single point (i.e., φ^c has a single solution). Therefore, π admits a single accepting run. Therefore, either both Conditions 1. and 2. hold, or the only solution falsifies both conditions. Indeed, assuming $x = k_x$ and $y = k_y$ and $z = k_z$, for non-negative reals k_x, k_y and k_z, it is immediate to see that the single timed sequence accepted by π belongs to $\mathcal{L}_0$ if and only if the following three conditions hold: $k_x \geq 1$, $k_z = 1 - (k_x - k_y)$, and $0 \leq k_z \leq 1$. Hence, the accepted timed sequence is in $\mathcal{L}_0$ if and only if the claim is fulfilled.
iii) if $[\![\varphi^c]\!]$ has relative interior of affine dimension ≥ 1, then, by convexity φ^c, there must be a convex subset X of $[\![\varphi^c]\!]$, which has relative interior of affine dimension 1. Therefore, X is contained within a straight line in $\mathbb{R}^3$ and has relative interior of dimension 1. Since φ^c is a conjunction of difference constraints, X must be completely contained in one of the straight lines reported in Fig. 15a, for suitable non-negative reals k_1, k_2 and k_3. Fig. 15b shows an instance of those seven straight lines when all the constants k_1, k_2 and k_3 are equal to 0. As we shall see, in all the cases either one of the conditions of the claim holds or we obtain a contradiction with the hypothesis that $\mathcal{L}_\pi \subseteq \mathcal{L}_0$. In Cases 1–5 the contradiction is obtained by providing two distinct points (clock valuations), which are contained in the corresponding straight line, belong to $[\![\varphi^c]\!]$, but assign different values to the expression $(x - y) + z$. Therefore, one of the two valuations must assign to that expression a value different from 1, thus witnessing a run induced by π accepting a timed sequence with $\Delta_1 + \Delta_3 \neq 1$.
Assume (x_1, y_1, z_1) is a solution in the relative interior of $[\![\varphi^c]\!]$. By assumption it must correspond to the clock valuation at the end of an accepting run of π over a timed sequence in $\mathcal{L}_0$. Let us consider in turn each straight line in Fig. 15a:
1. since (x_1, y_1, z_1) lies in the relative interior, there exists some $\Delta > 0$ such that the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 + \Delta$ and $z_2 = z_1 + \Delta$ still lies on the straight line L1 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
2. similarly, for some $\Delta > 0$, the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 = k_2$ and $z_2 = z_1 + \Delta$ still lies on the straight line identified L2 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
3. once again, for some $\Delta > 0$, the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 = k_2$ and $z_2 = z_1 = k_3$ still lies on the straight line L3 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
4. in case 4 the possibility to shift (x_1, y_1, z_1) by some $\Delta > 0$, without moving from the corresponding straight line, depends upon whether $(x_1 - y_1)$ is positive or zero and, in this last case, whether $(y_1 - z_1)$ is positive or zero. We have three possible cases:
   - if $(x_1 - y_1) > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 + \Delta$ and $z_2 = z_1 = k_3$ still lies on the straight line L4 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   - if $(x_1 - y_1) = 0$ and $(y_1 - z_1) > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 - \Delta$ and $z_2 = z_1 = k_3$ still lies on the straight line L4 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;

- if $(x_1 - y_1) = 0$ and $(y_1 - z_1) = 0$, then the accepted timed sequence provides a, b at the same time instant 0. Therefore, if $z_1 = 1$, then this implies Condition 2 of the claim with $\hat{k} = 0$. Otherwise $\Delta_1 + \Delta_3 = k_3 \neq 1$ and the timed sequence does not belong to $\mathcal{L}_0$, contradicting the hypothesis;

5. similarly to the above case, we have three subcases:
   - if $(y_1 - z_1) > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 = k_2$ and $z_2 = z_1 + \Delta$ still lies on the straight line L5 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   - if $(y_1 - z_1) = 0$ and $z_1 > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 = k_2$ and $z_2 = z_1 - \Delta$ still lies on the straight line L5 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   - in the case where $(y_1 - z_1) = 0$ and $z_1 = 0$ we also have $y_1 = 0$. Then, either $x_1 \neq 1$ and a contradiction follows, since π would accept a timed sequence with $\Delta_1 + \Delta_3 \neq 1$ not in $\mathcal{L}_0$, or $x_1 = 1$ and Condition 1 of the claim with $k = 1$ is ensured;
6. if $k_2 = k_1 - 1$ and $k_1 \geq 1$, then Condition 1 of the claim holds. Otherwise, either $(y_1 - z_1) = k_2 \neq k_1 - 1 = x_1 - 1$ or $0 \leq x_1 = k_1 < 1$. In either case we obtain a contradiction, since $(x_1 - y_1) + z_1 = k_1 - k_2 \neq 1$. Therefore, either we obtain an accepted timed sequence with $\Delta_1 + \Delta_3 \neq 1$, or it must hold $(y_1 - z_1) < 0$, which is impossible since, by the definition of the timed automaton, $y \geq z$ along any run;
7. similarly to the above case, if $k_3 = 1 - k_1$ and $0 \leq k_1 \leq 1$, then Condition 2 of the claim holds. Otherwise, $z_1 = k_3 \neq 1 - k_1 = 1 - (x_1 - y_1)$ or $z_1 = k_3 > 1$ or $k_3 < 0$. In each case we obtain a contradiction, since either $(x_1 - y_1) + z_1 = k_1 + k_3 \neq 1$ and the accepted timed sequence would not be in $\mathcal{L}_0$, or at least one of $(x_1 - y_1)$ and z_1 is negative, which is clearly impossible given the definition of the timed automaton. □

Claim 2. *Let π be a* timed path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_1$ if and only if the current constraint φ_c^c of the transition reading c satisfies the following condition:*

1. *there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^c implies $x - z = 1$, $x - y = \hat{k}$ and $z = 1 - \hat{k}$.*

Proof. The *if* direction is obvious since, whenever the condition of the claim holds, both $\Delta_1 + \Delta_3 = 1$ and $\Delta_1 + \Delta_2 = 1$.

For the other direction, we shall once again prove that either the conditions of the claim holds, or some timed sequence in $\mathcal{L}_\pi$ does not belong to $\mathcal{L}_1$. The proof proceeds similarly to the proof of Claim 1. However, in this case the additional constraint, requiring that $\Delta_1 + \Delta_2 = 1$, rules out the possibility that $[\![\varphi_c^c]\!]$ contains solutions assigning a value different from 1 to the clock expression $(x - z)$. Therefore, $[\![\varphi_c^c]\!]$ must be contained within the plane identified by the constraint $(x - z) = 1$. Let us consider the possible cases:

i) if $[\![\varphi_c^c]\!] = \emptyset$, then $\mathcal{L}_\pi = \emptyset$ and we have an immediate contradiction.

ii) if $[\![\varphi_c^c]\!]$ contains a single point (i.e., φ_c^c has a single solution), then φ_c^c would imply that all the clock expressions have a constant value. Therefore, either this solution satisfies Conditions 1., or it corresponds to an accepted timed sequence not in $\mathcal{L}_1$ and we have a contradiction. In particular, assuming $x = k_x$ and $y = k_y$ and $z = k_z$, for non-negative reals k_x, k_y and k_z, it is immediate to see that the single timed sequence accepted by π belongs to $\mathcal{L}_1$ if and only if all the following conditions hold: $k_x = k_z + 1$, and $0 \leq k_z \leq 1$ and $(k_x - k_y) = 1 - k_z$.

iii) if $[\![\varphi_c^c]\!]$ has relative interior of affine dimension ≥ 1, then, by convexity of φ_c^c, there must be a convex set $X \subseteq [\![\varphi_c^c]\!]$, whose relative interior has affine dimension 1. Hence, X is contained within a straight line in $\mathbb{R}^3$. Since the plane $(x - z) = 1$ intersects the straight lines L3, L5, L6 and L7 (see Fig. 15a) in a single point, all these cases collapse to Case ii) above. The only cases for which $X \subseteq [\![\varphi_c^c]\!]$ has relative interior of affine dimension 1 are those corresponding to the lines L1, L2 and L4. Let (x_1, y_1, z_1) be a solution in the relative interior of the intersection of X and the straight line $\mathrm{L} \in \{\mathrm{L1}, \mathrm{L2}, \mathrm{L4}\}$. By assumption, it must correspond to the clock valuation at the end of an accepting run induced by π over a timed sequence in $\mathcal{L}_1$. Let us consider in turn the three cases above:
   1) if $\mathrm{L} = \mathrm{L1}$, then there exists some $\Delta > 0$ such that the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 + \Delta$ and $z_2 = z_1 + \Delta$ still lies on the straight line L1 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   2) similarly, if $\mathrm{L} = \mathrm{L2}$, then for some $\Delta > 0$, the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 = k_2$ and $z_2 = z_1 + \Delta$ still lies on the straight line L2 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   3) finally, if $\mathrm{L} = \mathrm{L4}$, then the possibility to shift (x_1, y_1, z_1) by some $\Delta > 0$, while still remaining in L4, depends upon whether $(x_1 - y_1)$ is positive or zero and, in this last case, whether $(y_1 - z_1)$ is positive or zero. We have three possible cases:
      - if $(x_1 - y_1) > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 + \Delta$ and $z_2 = z_1 = k_3$ still lies on the straight line L4 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
      - if $(x_1 - y_1) = 0$ and $(y_1 - z_1) > 0$, then for some $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 - \Delta$ and $z_2 = z_1 = k_3$ still lies on the straight line L4 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
      - the case where $(x_1 - y_1) = 0$ and $(y_1 - z_1) = 0$ is impossible, since it implies that π accepts a timed sequence with $\Delta_1 + \Delta_2 = 0$, which does not belong to $\mathcal{L}_1$, contradicting the hypothesis. □

Claim 3. *Let π be a* store path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_1$ if and only if the constraints φ_c^b and φ_c^c satisfy the following condition:*

1. *there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^b implies $y = \hat{k}$ and φ_c^c implies both $x = 1$ and $x - z = 1 - \hat{k}$.*

Proof. Let us consider the *if* direction first. As reported in Table 2, the value of the clock expression y when φ_c^b is evaluated is precisely Δ_2; the value of the clock expression x when φ_c^c is evaluated is precisely $\Delta_1 + \Delta_3$; and the value of the clock expression z when φ_c^c is evaluated is precisely Δ_3. It is immediate, then, to see that any run, induced by π, accepting a timed sequence in $\mathcal{L}_1$ must satisfy the condition of the claim for some $0 \leq \hat{k} \leq 1$, with $\hat{k} = \Delta_2$.

For the other direction, no solution in $[\![\varphi_c^c]\!]$ can assign to x a value different from 1. Indeed, when the (restore) transition reading c is executed, the value of clock x contain precisely the duration $\Delta_1 + \Delta_3$, which must be equal to 1 for any timed sequence in $\mathcal{L}_1$. Let us consider a solution in $[\![\varphi_c^b]\!]$ and let k_y be the value of clock y for that solution. Similarly, let k_z be the value of clock z in some solution in $[\![\varphi_c^c]\!]$. Recall that, when the (Restore) transition reading b is executed, the value k_y of clock y corresponds precisely to the duration Δ_2. Moreover, when the (Restore) transition reading c is executed, the value k_z of clock z corresponds precisely to the duration Δ_3 and the value $(1 - k_z)$ of the expression $(x - z)$ to the duration Δ_1. Since $\Delta_1 \geq 0$ and any timed sequence in $\mathcal{L}_1$ must satisfy $\Delta_1 + \Delta_2 = 1$, both $k_y \leq 1$ and $k_z \leq 1$ must hold. Assume now that $k_y \neq k_z$ (and, therefore, the value of the expression $x - z \neq 1 - k_y$). Then the accepted timed sequence would satisfy $\Delta_1 + \Delta_2 \neq 1$, contradicting the hypothesis that it belongs to $\mathcal{L}_1$.

Therefore, there can only be a single solution in $[\![\varphi_c^b]\!]$, assigning the value $0 \leq k_y \leq 1$ to y, and a single solution in $[\![\varphi_c^c]\!]$, assigning 1 to x (and y), and $k_z (= k_y)$ to z (so that $x - z = 1 - k_y$), thus satisfying the condition of the claim. □

Claim 4. *Let π be a* timed path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_2$ if and only if the constraint φ_c^c on the transition reading c satisfies the following condition:*

1. *if π is of* timed type*, then there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^c implies $y = 1$, $x - y = \hat{k}$ and $z = 1 - \hat{k}$.*

Proof. The *if* direction is obvious, since if the condition of the claim holds, then necessarily both $\Delta_1 + \Delta_3 = 1$ and $\Delta_1 + \Delta_2 = 1$.

For the other direction, the proof proceeds similarly to the proof of Claim 1. We shall that either the conditions of the claim holds, or some timed sequence in $\mathcal{L}_\pi$ does not belong to $\mathcal{L}_2$. Notice that the additional constraint, requiring that $\Delta_2 + \Delta_3 = 1$, rules out the possibility that $[\![\varphi_c^c]\!]$ contains solutions assigning a value different from 1 to the clock expression y, as any induced accepting run would accept a timed sequence not in $\mathcal{L}_2$. Therefore, $[\![\varphi_c^c]\!]$ must be contained within the plane identified by the constraint $y = 1$. Let us consider the possible cases:

i) if $[\![\varphi_c^c]\!] = \emptyset$, then $\mathcal{L}_\pi = \emptyset$ and we have an immediate contradiction.
ii) if $[\![\varphi^c]\!]$ contains a single point (i.e., φ_c^c has a single solution), then φ_c^c would imply that all the clock expressions have a constant values. Therefore, either Conditions 1. holds, or the only solution corresponds to a timed sequence not in $\mathcal{L}_2$ and we have a contradiction. Indeed, assuming $x = k_x$ and $y = 1$ (since the point cannot lie outside the plane $y = 1$) and $z = k_z$, for non-negative reals k_x and k_z, it is immediate to see that the single timed sequence accepted by any run along π belongs to $\mathcal{L}_2$ if and only if the following conditions hold: $0 \leq k_z \leq 1$ and $(k_x - k_y) = (k_x - 1) = 1 - k_z$;
iii) assume $[\![\varphi_c^c]\!]$ has relative interior of affine dimension ≥ 1. Then, by convexity of φ_c^c, there must be a convex set $X \subseteq [\![\varphi_c^c]\!]$, whose relative interior has affine dimension 1. Hence, X is contained within a straight line in $\mathbb{R}^3$. Since the plane $y = 1$ intersects the straight lines L1, L4, L6 and L7 (see Fig. 15a) in a single point, all these cases collapse to Case ii) above. The only cases where $X \subseteq [\![\varphi_c^c]\!]$ has relative interior of dimension 1 are the straight lines L2, L3 and L5. Let us consider a solution (x_1, y_1, z_1) in the relative interior of the intersection of $[\![\varphi_c^c]\!]$ and the straight line $\mathrm{L} \in \{\mathrm{L2}, \mathrm{L3}, \mathrm{L5}\}$. By assumption, it must correspond to the clock valuation at the end of an accepting run of π over a timed sequence in $\mathcal{L}_2$. Let us consider in turn the three cases above:
   1) if L = L2, then for some $\Delta > 0$, the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 = k_2$ and $z_2 = z_1 + \Delta$ still lies on the straight line L2 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   2) similarly, if L = L3, then for some $\Delta > 0$, the solution (x_2, y_2, z_2) with $x_2 = x_1 + \Delta$, $y_2 = y_1 = k_2$ and $z_2 = z_1 = k_3$ still lies on the straight line L3 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
   3) consider the case where L = L5. Notice that, by construction of the automaton, $(y - z) \geq 0$ always holds on reading c. It follows, then, that any solution must satisfy $0 \leq z \leq 1$. Hence, we have two cases:
      - if $z_1 < 1$, then there exists $\Delta > 0$ such that $x_2 = x_1 = k_1$, $y_2 = y_1 = k_2$ and $z_2 = z_1 + \Delta$ still lies on the straight line L5 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$;
      - if, on the other hand, $z_1 > 0$, then there exists $\Delta > 0$, $x_2 = x_1 = k_1$, $y_2 = y_1 = k_2$ and $z_2 = z_1 - \Delta$ still lies on the straight line L5 and $(x_2 - y_2) + z_2 \neq (x_1 - y_1) + z_1$. □

Claim 5. *Let π be a* store path *of the automaton. Then $\mathcal{L}_\pi \subseteq \mathcal{L}_2$ if and only if the constraints φ_c^b and φ_c^c satisfy the following condition:*

1. *there exists a constant $0 \leq \hat{k} \leq 1$ such that φ_c^b implies $y = \hat{k}$ and φ_c^c implies both $x = 1$ and $z = 1 - \hat{k}$.*

Proof. As reported in Table 3, the value of the clock expression y is precisely Δ_2, when constraint φ_c^b is evaluated; the value of the clock expression x is precisely $\Delta_1 + \Delta_3$, when φ_c^c is evaluated; and the value of the clock expression z is Δ_3, when φ_c^c is evaluated. It is, then, immediate to see that any run induced by π and accepting a timed sequence in $\mathcal{L}_2$ must satisfy the condition of the claim for some $0 \leq \hat{k} \leq 1$, where $\hat{k} = \Delta_2$.

For the other direction, no solution in $[\![\varphi_c^c]\!]$ can assign to x (recording $\Delta_1 + \Delta_3$) a value different from 1, otherwise a timed sequence in the complement of $\mathcal{L}_2$ would be accepted, contradicting the hypothesis.

Let us consider a solution in $[\![\varphi_c^b]\!]$ and let k_y be the value of clock y in that solution. Similarly, let k_z be the value of clock z in some solution in $[\![\varphi_c^c]\!]$. Recall that, when the (restore) transition reading b is executed, the value k_y of clock y corresponds precisely to the duration Δ_2. Moreover, when the (restore) transition reading c is executed, the value k_z of clock z corresponds precisely to the duration Δ_3. Since $\Delta_1 \geq 0$ and any timed sequence in $\mathcal{L}_2$ must satisfy $\Delta_2 + \Delta_3 = 1$, both $k_y \leq 1$ and $k_z \leq 1$ must hold. Assume now that $k_z \neq 1 - k_y$. Then the accepted timed sequence would satisfy $\Delta_2 + \Delta_3 \neq 1$, contradicting the hypothesis that it belongs to $\mathcal{L}_2$.

Therefore, there can only be a single solution in $[\![\varphi_c^b]\!]$, assigning the value $0 \leq k_y \leq 1$ to y, and a single solution in $[\![\varphi_c^c]\!]$, assigning 1 to x (and y), and $k_z (= 1 - k_y)$ to z, thus satisfying the condition of the claim. □

A.2. Results in Section 6.1

Theorem 6. *The reachability problem for* S-EPTA *is undecidable.*

Proof. Let $\mathcal{M}$ be any 2-counter Machine. Let, in addition, C^+ (resp.: C^-) denote the set of the labels of increment (resp.: decrement) instructions. We build a S-EPTA $\mathcal{P}_\mathcal{M}$, whose set of states of consists of pairs of the form (p, q), where either $p \in C^+$ and $q \in \{q_0^+, q_1^+\}$ or $p \in C^-$ and $q \in \{q_i^- \mid 0 \leq i \leq 3\}$. An additional state *Halt* represents the halting state of $\mathcal{M}$. Each instruction of $\mathcal{M}$ is simulated by a suitable instantiation of the corresponding gadget reported in Fig. 13.

Formally, the S-EPTA automaton for $\mathcal{M}$ is defined as follows. $\mathcal{P}_\mathcal{M} = \langle Q, (p_0, s), X, \Gamma, \bot, T \rangle$, where:

- $Q = (C^+ \times \{q_0^+, q_1^+\}) \cup (C^- \times \{q_i^- : 0 \leq i \leq 3\}) \cup \{Halt\}$;
- $s = q_0^+$ if $p_0 \in C^+$ and $s = q_0^-$ if $p_0 \in C^-$;
- $X = \{x, y, x', x''\}$ and $\Gamma = \{\bot, \natural\}$;
- for every $p \in C^+$ associated with the increment of the counter $z \in \{x, y\}$ with target label p', T has the transitions:
  1. $(p, q_0^+) \xrightarrow[\natural\gamma, Store, \emptyset, \emptyset]{\gamma, \sigma, y=0, True} (p, q_1^+)$ with $\gamma \in \{\bot, \natural\}$;
  2. $(p, q_1^+) \xrightarrow[\epsilon, Restore, X\setminus\{z\}, \emptyset]{\natural, \sigma, y=1, True} q$,

  where:
  1. $q = (p', q_0^+)$ if $p' \in C^+$;
  2. $q = (p', q_0^-)$ if $p' \in C^-$;
  3. $q = Halt$, if $p' = HALT$;
- for every $p \in C^-$ associated with the decrement of the counter $z \in \{x, y\}$, with target labels p_1 and p_2 respectively, T has the transitions:
  1. $(p, q_0^-) \xrightarrow[\natural\gamma, Store, \emptyset, \{y'\}]{\gamma, \sigma, y=0 \wedge z>1, True} (p, q_1^-)$, with $\gamma \in \{\bot, \natural\}$;
  2. $(p, q_0^-) \xrightarrow[\gamma, Reset, \emptyset, \emptyset]{\gamma, \sigma, z=0, True} q_2$, with $\gamma \in \{\bot, \natural\}$;
  3. $(p, q_1^-) \xrightarrow[\epsilon, Restore, X\setminus\{y'\}, \emptyset]{\natural, \sigma, True, y'=z} (p, q_2^-)$;
  4. $(p, q_2^-) \xrightarrow[\natural\gamma, Store, \emptyset, \{z\}]{\gamma, \sigma, y=0, True} (p, q_3^-)$, with $\gamma \in \{\bot, \natural\}$;
  5. $(p, q_3^-) \xrightarrow[\epsilon, Restore, X\setminus\{z\}, \emptyset]{\natural, \sigma, True, y'-z=1} q_1$;

  where, for $i \in \{1, 2\}$:
  1. $q_i = (p_i, q_0^+)$ if $p_i \in C^+$;
  2. $q_i = (p_i, q_0^-)$ if $p_i \in C^-$;
  3. $q_i = Halt$, if $p_i = HALT$;
- a transition $HALT \xrightarrow[\epsilon, Restore, \emptyset, \emptyset]{\gamma, \sigma, True, True} HALT$, with $\gamma \in \{\bot, \natural\}$.

It is immediate to verify that $\mathcal{P}_\mathcal{M}$ actually simulates $\mathcal{M}$. Hence, we conclude that solving the reachability problem for S-EPTA is at least as hard as solving the same problem for 2-counter Machines. □

A.3. Results in Section 6.2

Theorem 7. *An* L-EPTA $\mathcal{E}$ *accepts a timed sequence* η *if and only if the region PDA* $\mathcal{P}$ *of* $\mathcal{E}$ *accepts the sequence* $Untime(\eta)$.

Proof. Let us consider the two directions in turn:

(⇒) Let

$$\lambda = gc_0 \xrightarrow{\sigma_0} gc_1 \xrightarrow{\sigma_1} gc_2 \cdots gc_{n-1} \xrightarrow{\sigma_{n-1}} gc_n$$

be a run of $\mathcal{E}$ from some reachable configuration gc_0, with $gc_j = \langle q_j, \mathbf{v}_j, w_j, d_j \rangle$ for all $0 \le j \le n$. We start setting a correspondence between configurations of the L-EPTA $\mathcal{E}$ and of the region PDA $\mathcal{P}$ of $\mathcal{E}$. For a valuation $\mathbf{v}$, let $Reg(\mathbf{v})$ denote the region of $\mathbf{v}$, namely the region R such that $\mathbf{v} \in R$. We combine a string of symbols $\mathbf{w} = w_{k-1} \dots w_0$, a valuation $\mathbf{v}_k$ and a sequence of valuations $\mathbf{d} = \mathbf{v}_{k-1}\, \mathbf{v}_{k-1} \dots \mathbf{v}_0$ into a sequence of region contexts with the following operation:

$$\mathbf{w} \times_{Reg} \mathbf{v} \cdot \mathbf{d} = (w_{k-1}, Reg(\mathbf{v}_{k-1}), Reg(\mathbf{v}_k)) \dots (w_0, Reg(\mathbf{v}_0), Reg(\mathbf{v}_1)).$$

The operation $\times_{Reg}$ builds a sequence of triples of the form (w, R, R'), where w is a symbol at some position j in the first sequence $\mathbf{w}$, while R and R' are the regions containing two adjacent valuations in the second sequence $\mathbf{v} \cdot \mathbf{d}$, respectively at positions j and $j+1$, with $0 \le j \le k$. The sequence $\mathbf{w}$ corresponds to a configuration of length k of the control stack, while $\mathbf{v}_k \cdot \mathbf{d}$ contains $k+1$ valuations, where $\mathbf{v}_k$ is the current valuation and $\mathbf{d}$ contains the k valuations in the valuation stack. Then, each triple (w, R, R') in the sequence $\mathbf{w} \times_{Reg} \mathbf{d}$ contains precisely a region context, i.e., the semantic information which is contained in a global configuration of the LTS of $\mathcal{E}$ and is needed for the simulation.

For an EPTA configuration $gc = \langle q, \mathbf{v}, \mathbf{w}, \mathbf{d} \rangle$, let $rpda(gs) = \langle q, \mathbf{w} \times_{Reg} \mathbf{v} \cdot \mathbf{w} \rangle$ be the induced PDA configuration if $|\mathbf{w}| > 0$ and $rpda(gc) = \langle q, \epsilon \rangle$, otherwise. Now, let $\eta = tseq(\lambda) = (a_0, t_0)(a_1, t_1) \dots (a_{k-1}, t_{k-1})$ be the timed sequence inducing the considered run and $j_0 < j_1 < \cdots < j_{k-1} \le n$ be the sequence of symbols such that $a_i = \sigma_{j_i}$, for all $0 \le i < k$. We can show, by induction on the length $k \ge 0$ of $tseq(\lambda)$, that there exists a run of $\mathcal{P}$ over η of the form

$$\lambda' = pc_0 \xrightarrow{\sigma'_0} pc_1 \xrightarrow{\sigma'_1} pc_2 \cdots pc_{r-1} \xrightarrow{\sigma'_{r-1}} pc_r = st$$

with $k \le r \le n$, such that, for some sequence $0 \le j'_0 < j'_1 < \cdots < j'_{k-1} \le r$, the following conditions hold: *(i)* $pc_0 = rpda(gc_0)$; *(ii)* $\sigma'_{j'_i} = a_i$ and $pc_{j'_i} = rpda(gc_{j_i})$, for $0 \le i < k$; and *(iii)* $st = rpda(gc_{j_{k-1}+1})$ if $gc_{j_{k-1}+1}$ is such that $d_{j_{k-1}+1} \ne \epsilon$, and $st = \langle \hat{tr}, \epsilon \rangle$ if $d_{j_{k-1}+1} = \epsilon$ and tr is the Restore transition of $\mathcal{E}$ taking from $gc_{j_{k-1}}$ to $gc_{j_{k-1}+1}$. Moreover λ' accepts $Untime(\eta)$ if and only if λ accepts η.

The case where $k = 0$ is trivial. Let us consider $k \ge 1$ and let the initial configuration be $gc_0 = \langle q_0, \mathbf{v}_0, a \cdot \mathbf{w}_0, \mathbf{v} \cdot \mathbf{d}_0 \rangle$, with $|\mathbf{w}_0| = |\mathbf{d}_0| \ge 0$. Then, $rpda(gc_0) = \langle q_0, (a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)) \cdot \mathbf{z}_0 \rangle$, where $\mathbf{z}_0 = \mathbf{w}_0 \times_{Reg} \mathbf{v} \cdot \mathbf{d}_0$. The transitions triggered by σ_i, with $0 \le i < j_0$, are clearly all Progress transitions and $\mathbf{v}_{j_0} = \mathbf{v}_0 + t$, with $t = \sum_{j=0}^{j_0-1} \sigma_j$ and $q_{j_0} = q_0$. Therefore, there is a transition $tr \in T_E$ (either a Reset, a Store or a Restore transition), which induces the corresponding transition labeled by σ_{j_0} in the LTS of $\mathcal{E}$. Let us consider the three possible cases, according to the type of the transition tr:

Reset: if tr is of the form $q_0 \xrightarrow[\gamma, Reset, \emptyset, r]{a, \sigma_{j_0}, \varphi, True} q_{j_0+1}$, then we have a Swap transition in $\mathcal{P}$ of the form $q_0 \xrightarrow[(\gamma, Reg(\mathbf{v}), R'), Swap]{(a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)), \sigma_{j_0}} q_{j_0+1}$, where $R' = R''{\downarrow_r}$ and $R'' = Reg(\mathbf{v}_0 + t)$. Notice that $R'' \in Succ(R)$ and $R'' \in \varphi$ since $\mathbf{v}_0 + t \in \varphi$. Then, by construction, the LTS of $\mathcal{P}$ has a transition $\langle q_0, (a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)) \cdot \mathbf{z}_0 \rangle \xrightarrow{\sigma_{j_0}} \langle q_{j_0+1}, (\gamma, Reg(\mathbf{v}), R') \cdot \mathbf{z}_0 \rangle$, where $\langle q_{j_0+1}, (\gamma, Reg(\mathbf{v}), R') \cdot \mathbf{z}_0 \rangle = rpda(gc_{j_0+1})$. By applying the inductive hypothesis to the sub-run of $\mathcal{E}$ $gc_{j_0+1} \xrightarrow{\sigma_{j_0+1}} \cdots gc_{n-1} \xrightarrow{\sigma_{n-1}} gc_n$, we obtain a run λ'' of $\mathcal{P}$ from $rpda(gc_{j_0+1})$ over $Untime(\eta')$, with $\eta = (a_0.t_0) \cdot \eta'$, of length $k-1$. As a consequence, the run $rpdagc_0 \xrightarrow{\sigma_0} rpdagc_{j_0+1} \cdot \lambda''$ is a run of $\mathcal{P}$ accepting $Untime(\eta)$ if λ accepts η, and the thesis follows.

Store: if tr is of the form $q_0 \xrightarrow[a_1.a_2, Store, r]{a, \sigma_{j_0}, \varphi, True} q_{j_0+1}$, then we have a Push transition in the region PDA of the form $q_0 \xrightarrow[(a_1, R'', R'), (a_2, Reg(\mathbf{v}), R''), Store]{(a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)), \sigma_{j_0}} q_{j_0+1}$, where $R' = R''{\downarrow_r}$ and $R'' = Reg(\mathbf{v}_0 + t)$. Notice that $R'' \in Succ(Reg(\mathbf{v}_0))$ and $R'' \in \varphi$ since $\mathbf{v}_0 + t \in \varphi$. Again, by construction, the LTS of $\mathcal{P}$ has a transition $\langle q_0, (a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)) \cdot \mathbf{z}_0 \rangle \xrightarrow{\sigma_{j_0}} \langle q_{j_0+1}, (a_1, R'', R')\, (a_2, Reg(\mathbf{v}), R'') \cdot \mathbf{z}_0 \rangle$, with $\langle q_{j_0+1}, (a_1, R'', R')\, (a_2, Reg(\mathbf{v}), R'') \cdot \mathbf{z}_0 \rangle = rpda(gc_{j_0+1})$. As in the previous case, the inductive hypothesis applied to $gc_{j_0+1} \xrightarrow{\sigma_{j_0+1}} \cdots gc_{k-1} \xrightarrow{\sigma_{k-1}} gc_k$ proves the thesis.

Restore: if tr is of the form $q_0 \xrightarrow[\epsilon, Restore, X]{a, \sigma_{j_0}, \varphi_c, \varphi_s} q_{j_0+1} \in T_{\mathcal{E}}$, then we have a Pop transition in the region PDA of the form $q_0 \xrightarrow[\epsilon, Restore]{(a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)), \sigma_{j_0}} \hat{tr}$. Notice that $Reg(\mathbf{v}_0 + t) \in Succ(Reg(\mathbf{v}_0))$, $Reg(\mathbf{v}_0 + t) \in \varphi_c$, since $\mathbf{v}_0 + t \in \varphi_c$, and $Reg(\mathbf{v}) \in \varphi_s$, since $\mathbf{v} \in \varphi_s$.

By construction, the LTS of $\mathcal{P}$ has a transitions $\langle q_0, (a, Reg(\mathbf{v}), Reg(\mathbf{v}_0)) \cdot \mathbf{z}_0\rangle \xrightarrow{\sigma_{p_{j_0}}} \langle \hat{tr}, \mathbf{z}_0\rangle$.

Let us consider the case where $|\mathbf{z}_0| > 0$ and, in particular, $\mathbf{z}_0$ has the form $(a', R_s, Reg(\mathbf{v})) \cdot \mathbf{z}'$, for some $a' \in \Gamma_{\mathcal{E}}$ and some clock region R_s. By construction, there is a Swap transition $\hat{tr} \xrightarrow[\langle a', R_s, R'\rangle, Swap]{\langle a', R_s, Reg(\mathbf{v})\rangle, \tau} q_{j_0+1}$, and the LTS for $\mathcal{P}$ has a transition $\langle \hat{tr}, \langle a', R_s, Reg(\mathbf{v}) \cdot \mathbf{z}'\rangle\rangle \xrightarrow{\tau} \langle q_{j_0+1}, \mathbf{z}'\rangle$. Now, $\langle q_{j_0+1}, \langle a', R_s, R' \cdot \mathbf{z}'\rangle\rangle = rpda(gc_{j_0+1})$ and, by applying the inductive hypothesis to $gc_{j_0+1} \xrightarrow{\sigma_{j_0+1}} \cdots gc_{k-1} \xrightarrow{\sigma_{k-1}} gc_k$, we obtain the thesis.

Finally, let us consider the case where $|\mathbf{z}_0| = 0$. In this case gc_{j_0+1} is such that $d_{j_0+1} = \epsilon$ and $\mathbf{z}_0 = \mathbf{z}_{j_0+1} = \epsilon$ and there is no transition in the LTS of $\mathcal{E}$ from gc_{j_0+1}. Therefore, λ has the form $gc_0 \xrightarrow{\sigma_0} gc_1 \xrightarrow{\sigma_1} gc_2 \cdots gc_{k-1} \xrightarrow{\sigma_{j_0}} gc_{j_0+1}$ and it is an accepting run. Now, notice that the corresponding run of $\mathcal{P}$ is also accepting (it leads to the accepting configuration $\langle \hat{tr}, \epsilon\rangle$, from which no transition can be taken) and the thesis holds.

($\Leftarrow$) For the other direction, let $\lambda = pc_0 \xrightarrow{\sigma_0} pc_1 \cdots pc_{n-1} \xrightarrow{\sigma_{n-1}} pc_n$ be a run of $\mathcal{P}$, with $pc_i = \langle q_i, \mathbf{z}_i\rangle$, for $0 \le i \le n$. Let, in addition, $\rho = \sigma_{j_0}, \sigma_{j_1}, \ldots \sigma_{j_k}$ be the sequence of symbols in Σ accepted by λ, for a suitable sequence of indexes $j_0 < j_1 < \cdots < j_k \le n$ and a suitable $k \ge 0$. (Notice that $\sigma_i \in \Sigma \cup \{\tau\}$, for $0 \le i \le n$ whereas $\sigma_{j_h} \in \Sigma$ for $0 \le h \le k$ since τ occurrences are omitted). We prove that, whenever the initial configuration pc_0 is the image of $rpda(gc_0)$, for some global configuration gc_0 of the LTS of $\mathcal{E}$, then there exists a run of $\mathcal{E}$ of the form

$$gc_0 \xrightarrow{\tau(t_0)} gc_1 \xrightarrow{\sigma_{j_0}} gc_2 \xrightarrow{\tau(t_1)} gc_3 \xrightarrow{\sigma_{j_1}} gc_4 \cdots gc_{2k-2} \xrightarrow{\tau(t_{k-1})} gc_{2k-1} \xrightarrow{\sigma_{j_{k-1}}} gc_{2k}$$

with $rpda(gc_{2h}) = pc_{j_h}$, for all $0 \le h \le k$, and which accepts η if and only if λ accepts $\rho = Untime(\eta)$. Observe that this is sufficient to prove the theorem, since every initial configuration of $\mathcal{P}$ satisfies the assumption of being the image of some initial configuration of $\mathcal{E}$ under $rpda()$. Notice also that in the required run of $\mathcal{E}$, there is a strict alternation of Progress transitions and discrete transitions.

We proceed by induction on the length $k \ge 0$ of the accepted sequence $\sigma_{j_0}, \sigma_{j_1}, \ldots \sigma_{j_{k-1}}$ (i.e. on the number of non-silent symbols occurring in $\sigma_0, \sigma_1, \ldots \sigma_n$). The base case where $k = 0$ is trivial. Let us consider the case where $k > 0$. Assume $pc_0 = \langle q_0, (a, R, R_0) \cdot \mathbf{z}\rangle$ and that $gc_0 = \langle q_0, \mathbf{v}_0, a \cdot \mathbf{w}, \mathbf{v} \cdot \mathbf{d}\rangle$, with $pc_0 = rpda(gc_0)$. Then, by definition, $Reg(\mathbf{v}) = R$, $Reg(\mathbf{v}_0) = R_0$ and $\mathbf{w} \times_{Reg} \mathbf{v} \cdot \mathbf{d} = \mathbf{z}$.

Since $pc_0 = rpda(gc_0)$, the control state q_0 of pc_0 must belong to $Q_{\mathcal{E}}$. Therefore, by construction of $\mathcal{P}$, it holds that $\sigma_0 \ne \tau$ (recall that τ-transitions in $\mathcal{P}$ cannot start from states in $Q_{\mathcal{E}}$) and, therefore, $j_0 = 0$. We consider different cases, depending on the type of first transition in λ, triggered by the $\sigma_0 \ne \tau$.

Swap: Assume the first transition in λ is of the form $\langle q_0, (a, R, R_0) \cdot \mathbf{z}\rangle \xrightarrow{\sigma_0} \langle q_1, (a', R, R') \cdot \mathbf{z}\rangle$ and is induced by the application of a Swap transition of $\mathcal{P}$ having the form $q_0 \xrightarrow[(a', R, R'), Swap]{(a, R, R_0), \sigma_0} q_1$. Then, by construction, there must be a corresponding transition $tr \in T_{\mathcal{E}}$ of the form $q_0 \xrightarrow[a', Reset, \emptyset, r]{a, \sigma_0, \varphi, True} q_1$, such that $R' = R''\downarrow_r$, $R'' \in Succ(R_0)$ and $R'' \in \varphi$. Therefore, there exists a time $t_0 \ge 0$ such that $R'' = Reg(\mathbf{v}_0 + t_0)$ and $(\mathbf{v}_0 + t_0) \in \varphi$, since $R'' \in \varphi$.

As a consequence, there exists a Progress transition in the LTS of $\mathcal{E}$ of the form $gc_0 \xrightarrow{\tau(t_0)} gc_1$, with $gc_1 = \langle q_0, \mathbf{v}_0 + t_0, a \cdot \mathbf{w}, \mathbf{v} \cdot \mathbf{d}\rangle$. The transition tr can then be taken from state gc_1, inducing the transition $gc_1 \xrightarrow{\sigma_{j_0}} gc_2$, with $gc_2 = \langle q_1, (\mathbf{v}_0 + t_0)\downarrow_r, a' \cdot \mathbf{w}, \mathbf{v} \cdot \mathbf{d}\rangle$, in the LTS of $\mathcal{E}$. Since $pc_1 = \langle q_1, (a', R, R') \cdot \mathbf{z}\rangle$, and $R' = Reg(\mathbf{v}_0 + t_0)\downarrow_r$, we have that $rpda(gc_2) = pc_1$.

Now, since pc_1 is the image of gc_2 under $rpda()$ and the sub-run $pc_1 \xrightarrow{\sigma_1} pc_2 \cdots pc_{n-1} \xrightarrow{\sigma_{n-1}} pc_n$ is induced by the sequence $\sigma_{j_1}, \sigma_{j_2}, \ldots \sigma_{j_{k-1}}$ of length $k-1$, we can apply the inductive hypothesis and obtain a run of $\mathcal{E}$ of the form

$$\lambda' = gc_2 \xrightarrow{\tau(t_1)} gc_3 \xrightarrow{\sigma_{j_1}} gc_4 \cdots gc_{2k-2} \xrightarrow{\tau(t_{k-1})} gc_{2k-1} \xrightarrow{\sigma_{j_{k-1}}} gc_{2k}$$

starting from configuration gc_2 and such that $rpda(gc_{2h}) = pc_{j_h}$, for all $1 \le h \le k-1$. This proves the thesis, since $gc_0 \xrightarrow{\tau(t_0)} gc_1 \xrightarrow{\sigma_{j_0}} \lambda'$ is a run of $\mathcal{E}$ which is accepting if and only if λ is.

Push: The case where the first transition in λ is induced by a Push transition of $\mathcal{P}$ can be managed analogously and is, then, omitted.

Pop: Assume the first transition in λ is of the form $\langle q_0, (a, R, R_0) \cdot \mathbf{z}\rangle \xrightarrow{\sigma_0} \langle q_1, \mathbf{z}\rangle$ and is induced by the application of a Pop transition of $\mathcal{P}$ having the form $q_0 \xrightarrow[\epsilon, Pop]{(a, R, R_0), \sigma_0} \hat{tr}$, where $\hat{tr}$ is the location of $\mathcal{P}$ associated with the transition $tr \in T_{\mathcal{E}}$. By construction, then, $q_1 = \hat{tr}$ and tr is a Restore transition of the form $q_0 \xrightarrow[\epsilon, Restore, X, r]{a, \sigma_{j_0}, \varphi_c, \varphi_s} q_2$. Moreover, $R \in \varphi_s$ and $R'' \in \varphi_c$, for some $R'' \in Succ(R_0)$. Assume first that $|\mathbf{z}| > 0$. Since $pc_0 = rpda(gc_0)$, then $gc_0 = \langle q_0, \mathbf{v}_0, a \cdot a' \cdot \mathbf{w}', \mathbf{v} \cdot \mathbf{v}' \cdot \mathbf{d}'\rangle$ and $z = (a' \cdot \mathbf{w}') \times_{Reg} (\mathbf{v}' \cdot \mathbf{d}') = (a', R', R) \cdot \mathbf{z}'$, with $a' \cdot \mathbf{w}' = \mathbf{w}$, $Reg(\mathbf{v}') = R'$, $Reg(\mathbf{v}) = R$.

Moreover, the second transition in λ from $pc_1 = \langle \hat{tr}, \mathbf{z} \rangle$ is of the form $\langle \hat{tr}, (a', R', R) \cdot \mathbf{z}' \rangle \xrightarrow{\tau} \langle q_2, (a', R', R_2) \cdot \mathbf{z}' \rangle$, induced by the Swap τ-transition of $\mathcal{P}$ of the form $\hat{tr} \xrightarrow[(a',R',R_2),Swap]{(a',R',R),\tau} q_2$, where $R_2 = R{\downarrow_r}$. Hence, $(a', R', R_2) \cdot \mathbf{z}' = a'\mathbf{w}' \times_{Reg} (\mathbf{v}{\downarrow_r}) \cdot \mathbf{d}'$.

Since $R'' \in Succ(R_0)$ and $Reg(\mathbf{v}_0) = R_0$, there exists a time t_0 such that $R'' = Reg(\mathbf{v}_0 + t_0)$ and the fact that $R'' \in \varphi_c$ implies $\mathbf{v}_0 + t_0 \in \varphi_c$. Similarly, since $R = Reg(\mathbf{v})$ and $R \in \varphi_s$, then $\mathbf{v} \in \varphi_s$. Therefore, the LTS of $\mathcal{E}$ has a Progress transition of the form $gc_0 \xrightarrow{\tau(t_0)} gc_1$, with $gc_1 = \langle q_0, \mathbf{v}_0 + t_0, a \cdot a' \cdot \mathbf{w}', \mathbf{v} \cdot \mathbf{v}' \cdot \mathbf{d}' \rangle$. Now, the transition tr can be taken from state gc_1 and we have a transition in the LTS of $\mathcal{E}$ of the form $gc_1 \xrightarrow{\sigma_0} gc_2$, with $gc_2 = \langle q_2, \mathbf{v}{\downarrow_r}, a' \cdot \mathbf{w}', \mathbf{v}' \cdot \mathbf{d}' \rangle$. Notice that since $pc_2 = \langle q_2, (a', R', R_2) \cdot \mathbf{z}' \rangle$, with $R_2 = Reg(\mathbf{v}){\downarrow_r}$ and $(a', R', R_2) \cdot \mathbf{z}' = (a' \cdot \mathbf{w}') \times_{Reg} (\mathbf{v}{\downarrow_r} \cdot \mathbf{v}' \cdot \mathbf{d}')$, we obtain that $rpda(gc_2) = pc_2$.

Let us consider the sub-run of λ starting from $pc_2 = rpda(gc_2)$. This sub-run is induced by the sequence of symbols $\sigma_{j_1}, \sigma_{j_2}, \ldots \sigma_{j_{k-1}}$ having length $k-1$. Once again, we can apply the inductive hypothesis and obtain a run of $\mathcal{E}$ of the form

$$gc_2 \xrightarrow{\tau(t_1)} gc_3 \xrightarrow{\sigma_{j_1}} gc_4 \cdots gc_{2k-2} \xrightarrow{\tau(t_{k-1})} gc_{2k-1} \xrightarrow{\sigma_{j_{k-1}}} gc_{2k}$$

starting from gc_2 and such that $rpda(gc_{2h}) = pc_{j_h}$, for all $1 \leq h \leq k$. Hence, the thesis follows.

Assume, now, that $|\mathbf{z}| = 0$. Then $pc_0 = \langle q_0, \langle a, R, R_0 \rangle \rangle$ and $pc_1 = \langle \hat{tr}, \epsilon \rangle$, with $tr = q_0 \xrightarrow[\epsilon,Restore,X,r]{a,\sigma_0,\varphi_c,\varphi_s} q_2 \in T_{\mathcal{E}}$. (Notice that in this case $n = 0$ and that the run is accepting since the resulting control stack is empty.) Now, $gc_0 = \langle q_0, \mathbf{v}_0, a, \mathbf{v} \rangle$ and, since $rpda(gc_0) = pc_0$, we have that $Reg(\mathbf{v}_0) = R_0$ and $Reg(\mathbf{v}) = R$. As in the previous case, there exists $R'' \in Succ(R_0)$, with $R'' \in \varphi_c$ and $R \in \varphi_s$. This implies that $\mathbf{v}_0 + t_0 \in \varphi_c$, for some $t_0 \geq 0$, and $\mathbf{v} \in \varphi_s$. Therefore, the LTS of $\mathcal{E}$ has a Progress transition of the form $gc_0 \xrightarrow{\tau(t_0)} gc_1$, with $gc_1 = \langle q_0, \mathbf{v}_0 + t_0, \mathbf{a}, \mathbf{v} \rangle$. Now, the transition tr can be applied to the configuration gc_1, resulting in the transition $gc_1 \xrightarrow{\sigma_0} gc_2$, with $gc_2 = \langle q_2, (\mathbf{v}_0 + t_0){\downarrow_r}, \epsilon, \epsilon \rangle$, in the LTS of $\mathcal{E}$. Since the control stack is empty, the run is accepting, thus proving the thesis. □

Theorem 8. *The emptiness problem and the reachability problem for* L-EPTA *are* ***EXPTIME****-hard.*

Proof. To prove the theorem, given an ATM $\mathcal{M}$ and an input word $\xi \in \Sigma^*$, we shall define an EPTA $\mathcal{E}_{\mathcal{M},\xi} = \mathcal{E}$ over an alphabet $\hat{\Sigma}$ belonging to L-EPTA, which implements the algorithm in Fig. 14.

The alphabet of $\mathcal{E}$ coincides with the alphabet of the configurations of $\mathcal{M}$, namely it is $\hat{\Sigma} = \Sigma \cup (Q \times \Sigma) \cup \{\sigma_0, \sharp\}$, where σ_0 and $\sharp$ are special symbols not in Σ. Assume a given ordering $\{\sigma_0, \sigma_1, \ldots, \sigma_k\}$ of the alphabet $\hat{\Sigma}$ and, for $p \in Parity$, let us denote with $\overline{p}$ the complement of p.

Let the EPTA $\mathcal{E}$ be defined as the tuple $\langle Q_{\mathcal{E}}, q_{0,\mathcal{E}}, X_{\mathcal{E}}, \Gamma_{\mathcal{E}}, \bot_{\mathcal{E}}, T_{\mathcal{E}} \rangle$, where

- $Q_{\mathcal{E}}$ is $Op \times Parity \times Seq \times Src \times Exp \times Curr \times Sy$, where
  - $Op = \{SC, CT, ST, RS\}$ is the set of operations (*SC* for scan, *CT* for check transition, *ST* for store, *RS* for restore);
  - $Parity = \{0, 1\}$ is the alphabet of parity (0 is even a 1 is odd);
  - $Seq = \{1 \ldots Max(Mout, 2n)\}$;
  - $Exp = Curr = Src = Q \cup \{\_\}$, are the alphabets for expected, current, source states respectively (_ stands for the undefined state);
  - $Sy = \Sigma \cup \{\_\}$, is the alphabet for read symbols (_ stands for the undefined symbol);
- $q_{0,\mathcal{E}} = \langle SC, 0, 1, \_, q_0, \_, \_ \rangle$;
- $X_{\mathcal{E}} = \{x_0, x_1^0, \ldots, x_n^0, x_1^1 \ldots, x_n^1\}$;
- $\Gamma_{\mathcal{E}} = Parity \times Q_\forall \times Q$;
- $T_{\mathcal{E}}$ contains the sets of transitions for Scanning a configuration, Checking a transition, Storing and Restoring a configuration defined below.

Scanning a configuration: for all $1 \leq i < n$, $p \in Parity$, $e, s \in Q \cup \{\_\}$, and $a \in Sy$:

$$\langle SC, p, 2i, s, e, c, a \rangle \xrightarrow[\gamma,Reset,\emptyset,\{x_0\}]{\sigma_0,\gamma,x_0=k+1,True} \langle SC, p, 2i+1, s, e, c, a \rangle \text{ with } c \in Q \cup \{\_\};$$

$$\langle SC, p, 2n, s, e, q, a \rangle \xrightarrow[\gamma,Reset,\emptyset,\{x_0\}]{\sigma_0,\gamma,x_0=k+1,True} \langle CT, p, 1, s, e, q, a \rangle \text{ with } q \in Q;$$

$$\langle SC, p, 2i-1, s, e, c, a \rangle \xrightarrow[\gamma,Reset,\emptyset,\{x_i^p\}]{\sigma_j,\gamma,x_0=k+1-j,True} \langle SC, p, 2i, s, e, c, a \rangle \text{ with } \sigma_j \in \Sigma, c \in Q \cup \{\_\};$$

$$\langle SC, p, 2i-1, s, e, \_, \_ \rangle \xrightarrow[\gamma,Reset,\emptyset,\{x_i^p\}]{(e',a),\gamma,x_0=k+1-j,True} \langle SC, p, 2i, s, e, e', a \rangle \text{ with } (e', a) = \sigma_j, e = \_ \text{ or } e = e'.$$

Table 5
The constraints to check transitions of $\mathcal{M}$.

Inv_z^p	$x_z^{\overline{p}} - x_z^p = n(k+1)$
$Curr_z^p(\sigma_j)$	$x_z^p = (n-z)(k+1)+j$
$Pre_z^p(\sigma_j)$	$x_z^{\overline{p}} = (2n-z)(k+1)+j$

Check a transition:

$$\langle CT, p, 1, \_, q_0, q_0, a\rangle \xrightarrow[\gamma, Reset, \emptyset, \emptyset]{\sharp, \gamma, x_0=0, True} \langle Op(q_0), Par^p(q), Id(q_0, a), q_0, \_, \_, \_\rangle$$

and, for all $1 \le h \le n$, $\sigma \in \Sigma$, and all $s, q \in Q$ such that the transition $t = \langle s, a, \sigma_j, m, q\rangle$ (with $m \in \{L, R\}$) belongs to δ,

$$\langle CT, p, 1, s, q, q, a\rangle \xrightarrow[\gamma, Reset, \emptyset, \emptyset]{\sharp, \gamma, x_0=0 \wedge tr^p(t,h,\sigma), True} \langle Op(q), Par^p(q), Id(q, a), q, \_, \_, Sym(q, a)\rangle$$

where $Op(q)$, $Id(q, a)$, $Par^p(q)$ and $Sym(q, a)$ for all $q \in Q$, $a \in \Sigma$, are defined as

$$Op(q) = \begin{cases} ST & \text{if } q \in Q_\forall \\ SC & \text{if } q \in Q_\exists \\ RT & \text{if } q \in Q_F \end{cases} \qquad Id(q, a) = \begin{cases} |Out(q, a)| & \text{if } q \in Q_\forall \\ 1 & \text{otherwise} \end{cases}$$

$$Par^p(q) = \begin{cases} \overline{p} & \text{if } q \in Q_\exists \\ p & \text{otherwise;} \end{cases} \qquad Sym(q, a) = \begin{cases} a & \text{if } q \in Q_\forall \\ \_ & \text{otherwise} \end{cases}$$

and $tr^p(t, h, \sigma)$, the clock constraint associated to transition t and parity $p \in Parity$, is defined as follows: If $t = \langle s, a, \sigma_j, R, q\rangle$ performs a right move, then

$$tr^p(t, h, \sigma) = Pre^p_{h-1}((s, a)) \wedge Pre^p_h(\sigma) \wedge Curr^p_{h-1}(\sigma_j) \wedge Curr^p_h((q, \sigma)) \wedge \bigwedge_{\substack{1 \le z \le n \\ h-1 \ne z \ne h}} Inv^p_z$$

where, for $1 \le z \le n$, the definitions of the constraints Inv^p_z, $Curr^p_z(\sigma_j)$ and $Pre^p_z(\sigma_j)$ are reported in Table 5, while $Curr^p_0(\sigma) = Pre^p_0(\sigma) = True$, for $p \in \{0, 1\}$ and each $\sigma \in \hat{\Sigma}$. The constraint Inv^p_z checks if the symbols at the z-th position in the two configurations are identical, by comparing the difference between clock x^p_z, encoding the z-th symbol in the target configuration, and clock $x^{\overline{p}}_z$, encoding the corresponding symbol in the source configuration. Constraint $Curr^p_z(\sigma)$ holds if the z-th position in the target configuration contains symbol σ, while $Pre^p_z(\sigma)$ holds if position k-th in the source configuration contains symbol σ. The constraint $tr^p(t, h, \sigma)$, then, checks that, if the machine head moves from position $h-1$ to position h, then: (i) the source and target configuration are identical in all positions except for positions $h-1$ and h; (ii) position $h-1$ in the source configuration contains (s, σ_i) (i.e., $Pre^p_{h-1}((s, a))$ holds) and the target configuration contains σ_j, the symbol written on the tape by t, in the same position (i.e., $Curr^p_{h-1}(\sigma_j)$ holds); and (iii) if position h in the source configuration contains σ (i.e., $Pre^p_h(\sigma)$ holds) then the target configuration contains the symbol (q, σ) in that position (i.e., $Curr^p_{h-1}((q, \sigma))$ holds). Notice that the non-determinism among the outgoing transitions takes care of choosing the correct position $h-1$ of the head in the source configuration and the symbol σ currently in position h in the target configuration.
Similarly, if $t = \langle s, a, \sigma_j, L, q\rangle$ performs a left move, instead, then

$$tr^p(t, h, \sigma) = Pre^p_h((s, a)) \wedge Pre^p_{h-1}(\sigma) \wedge Curr^p_h(\sigma_j) \wedge Curr^p_{h-1}((q, \sigma)) \wedge \bigwedge_{\substack{1 \le z \le n \\ h-1 \ne z \ne h}} Inv^p_z.$$

Storing clock valuations: for all $s \in Q_\forall$, $3 \le i \le |Out(s, a)|$ and $p \in Parity$

$$\langle ST, p, i, s, \_, \_, a\rangle \xrightarrow[\langle p,s,q\rangle\cdot\gamma, Store, \emptyset, \emptyset]{\sharp, \gamma, x_0=0, True} \langle ST, p, i-1, s, \_, \_, a\rangle \text{ with } q = Target(Out(s, a, i));$$

and, if $s \in Q_\forall$, $q = Target(Out(s, a, 2))$ and $q' = Target(Out(s, a, 1))$, then

$$\langle ST, p, 2, s, \_, \_, a\rangle \xrightarrow[\langle p,s,q\rangle\cdot\gamma, Store, \emptyset, \emptyset]{\sharp, \gamma, x_0=0, True} \langle SC, \overline{p}, 1, s, q', \_, \_\rangle;$$

Restoring clock valuations: for all $q', q'' \in Q$, $p, p' \in Parity$,

$$\langle RS, p, 1, s, e, c, \_\rangle \xrightarrow[\epsilon, Restore, \emptyset, \emptyset]{\sharp, \langle p', q', q''\rangle, x_0=0, True} \langle SC, \overline{p'}, 1, q', q'', \_, \_\rangle.$$

Note that the resulting EPTA is polynomial in n and in the size of the ATM $\mathcal{M}$. □

A.4. Results in Section 6.3

Lemma 2. *Let $gs = \langle b, u, \mathbf{v}_1, \mathbf{v}\rangle$ be a global state of $\mathcal{T}'$, with $\mathbf{v}, \mathbf{v}_1$ arbitrary valuations, $u \in N'_j$, for some j, and $b \in B'_i$, for i such that $Y'_i(b) = j$. Then,*

1. *the global state $gs' = \langle b_1, u', \mathbf{v}_1, \mathbf{v}'\rangle$, with $u' \in N'_j \setminus Ex'_j$, is reachable from gs iff $\langle u', \mathbf{v}'\rangle$ is reachable from $\langle u, \mathbf{v}\rangle$ in $\mathcal{A}_j^{(\hat{k})}$.*
2. *the global state $gs' = \langle \epsilon, (b_1, ex), \epsilon, \mathbf{v}'_1\rangle$ is reachable from gs if and only if there exists a Return transition $t = q \xrightarrow[X,r]{\varphi_c, \varphi_s} ex \in \delta'_j$ such that $\mathbf{v}_1 \in \varphi_s$, $\mathbf{v}'_1 = \mathbf{v}_1{\downarrow_r}$ and $\langle (ex, t), \mathbf{v}'\rangle$ is reachable from $\langle u, \mathbf{v}\rangle$ in $\mathcal{A}_j^{(\hat{k})}$.*

Proof. **(⇒)** The proof proceeds by induction on the number l of matching Call and Return transitions in the run λ from gs to gs'. If $l = 0$ then condition 1 is trivial, as λ can only contain Reset transitions, which are all included in the Timed Automaton $\mathcal{A}_j^0$ and, *a fortiori*, in $\mathcal{A}_j^{\hat{k}}$. For the second condition, the last transition in λ is due to a Return transition of the form $t = q \xrightarrow[X,r]{\sigma, \varphi_c, \varphi_s} ex$ taken from the global state $gs'' = \langle b_1, q, \mathbf{v}_1, \mathbf{v}''\rangle$, for some $\mathbf{v}''$ with $\mathbf{v}'' \in \varphi_c$. Moreover, global state gs'' is reachable from gs by following only Reset transitions, and $\mathbf{v}_1 \in \varphi_s$. Hence, $\langle q, \mathbf{v}''\rangle$ is reachable from $\langle u, \mathbf{v}\rangle$ in $\mathcal{A}_j^0$. By construction, the transition $t' = q \xrightarrow[r]{\sigma, \varphi_c} (ex, t)$ is a transition of $\mathcal{A}_j^0$ and, since $\mathbf{v}'' \in \varphi_c$, $\langle (ex, t), \mathbf{v}'\rangle$ is reachable from $\langle u, \mathbf{v}\rangle$ in $\mathcal{A}_j^0$. Hence the thesis for condition 2 follows.

For the inductive case, the run λ contains at least a Call transition and its matching Return. Let s be the maximal length of the invocation history of global states in λ. In addition, let h be the greatest index of a global state $gs_h = \langle b_1 \cdots b_{s-1}, q'', \mathbf{v}_1 \cdots \mathbf{v}_{s-1}, \mathbf{v}''\rangle$ in the run with invocation history of length $s-1$ and such that the transition leading to $gs_{h+1} = \langle b_1 \cdots b_s, en_s, \mathbf{v}_1 \cdots \mathbf{v}_s, \mathbf{v}_0\rangle$ is the result of the Call transition $t = q' \xrightarrow[\emptyset, X]{\sigma', \varphi, True} (b_s, en_s) \in \delta'_i$, with $1 \le i \le n$ and $Y_i(b_s) = j$. Let $z > h$ be the index of its matching Return transition, with $gs_z = \langle b_1 \cdots b_{s-1}, (b_s, ex_s), \mathbf{v}_1 \cdots \mathbf{v}_{s-1}, \mathbf{v}_z\rangle$. By maximality of h, the sub-run from index $h+1$ to index $z-1$ can only contain Reset transitions, and assume $t = q \xrightarrow[X,r]{\sigma, \varphi_c, \varphi_s} ex_s$ is the Return transition which leads from gs_{z-1} to gs_z within the run. Therefore, (ex_s, t) is reachable in the Timed Automaton $\mathcal{A}_j^0$ from en_s and, by construction, the transition $t'' = q' \xrightarrow[\emptyset, r]{\sigma, \varphi \wedge \varphi_s} (en_s, ex_s)_{b_s}$ belongs to $Summary(1, i) \subseteq Summary(\hat{k}, i)$. By replacing the sub-run from index h to index z in λ with the single transition induced by the summary transition t'', we obtain another run λ' from gs to gs', having $l-1$ matching Call and Return transitions. The induction hypothesis provides the thesis.

(⇐) The other direction for the first condition follows directly from the construction of the Timed Automata $\mathcal{A}_i^{(\hat{k})}$. Indeed, all the Reset transitions in the TAs are contained in $\mathcal{T}'$ and the conclusion immediately follows.

For the second condition, consider the run in $\mathcal{A}_j^{(\hat{k})}$ from $\langle u, \mathbf{v}\rangle$ to $\langle (ex, t), \mathbf{v}'\rangle$ and let $\langle q, \mathbf{v}''\rangle \to \langle (ex, t), \mathbf{v}'\rangle$ be the last transition of the run. The sub-run up to $\langle q, \mathbf{v}''\rangle$ contains only Reset transitions and, by the first condition, there is a run in $\mathcal{T}$ from gs to $\langle b_1, q, \mathbf{v}_1, \mathbf{v}''\rangle$. By construction, there must be a transition $t = q \xrightarrow[X,r]{\varphi_c, \varphi_s} ex \in \delta'_j$ such that $\mathbf{v}'' \in \varphi_c$ and, by the hypothesis, $\mathbf{v}_1 \in \varphi_s$, $\mathbf{v}'_1 = \mathbf{v}_1{\downarrow_r}$. Therefore, transition t can be applied to $\langle b_1, q, \mathbf{v}_1, \mathbf{v}''\rangle$, resulting in a run leading from gs to $gs' = \langle \epsilon, (b_1, ex), \epsilon, \mathbf{v}'_1\rangle$. □

Lemma 3. *Let $g_0 = \langle b_{init}, en\rangle$ and $g = \langle b_1 \cdots b_s, u\rangle$ be untimed global states of $\mathcal{T}$, with $b_1 = b_{init}$ if $s \ge 1$. Let $A_{i_1}, \ldots, A_{i_s}$, with $i_1 = 1$ and $Y_{i_{j-1}}(b_j) = i_j$ for $2 \le j \le s$, be the sequence of the invoked components in the history of g. Then, g is reachable from g_0 in $\mathcal{T}$ iff there exists a sequence of entry nodes $en_1, \ldots, en_s$, with $en_j \in En_{i_j}$ such that: for all $1 \le j \le s-1$, (b_{j+1}, en_{j+1}) is reachable from en_j in A'_{i_j}, following only Reset transitions except for the last Call transition leading to (b_{j+1}, en_{j+1}), and*

- *either $u \in N_{i_s}$ and u is locally reachable from en_s in A'_{i_s};*
- *or $u = (b, ex')$ and there exists a summary node $(en', ex')_b \in N'_{i_s}$ which is locally reachable from en_s in A'_{i_s},*

where we take $en_s = en_0$ and $A'_s = A'_{init}$ if $s = 0$.

Proof. To prove the lemma we first need to introduce the following function. Let $H : GS_{\mathcal{T}'} \times \mathbb{N} \to GS_{\mathcal{T}'}$ be a function that, given a global state $gs = \langle b_1 \cdots b_h, u, \mathbf{v}_1 \cdots \mathbf{v}_h, \mathbf{v} \rangle$, with invocation history $b_1 \cdots b_h$ of length h, and an integer $n \leq h$, returns the global state $\langle b_{n+1} \cdots b_h, u, \mathbf{v}_{n+1} \cdots \mathbf{v}_h, \mathbf{v} \rangle$, obtained by dropping the invocation history prefix $b_1 \cdots b_n$ and the valuation history prefix $\mathbf{v}_1 \cdots \mathbf{v}_n$ from gs. The function H can be extended to sequences of global states, hence runs, in the obvious way. Therefore, for a sequence $\lambda = gs_0 \to gs_1 \to \cdots gs_k$ of global states, such that each gs_i has a invocation history of length $> n$, we shall write $H(\lambda, n)$ to denote the sequence $H(gs_0, n) \to H(gs_1, n) \to \cdots H(gs_k, n)$. It is immediate to see that if λ is a run and all global states gs_i, with $0 \leq i \leq k$, have invocation history of length $> n$, then they all share the same invocation history prefix $b_1 \cdots b_n$ and, in addition, $H(\lambda, n)$ is also a run. We can now prove the two directions of the lemma.

($\Rightarrow$). We proceed by induction on the length of the invocation history in g. In the base case either $g = \langle \epsilon, (b_{init}, ex) \rangle$, for some $ex \in Ex_1$, or $g = \langle b_{init}, u \rangle$ and $u \in N_1 \setminus Ex_1$. In the first case, untimed reachability implies that there exists a valuation $\mathbf{v}$ such that $gs = \langle \epsilon, (b_{init}, ex), \epsilon, \mathbf{v} \rangle$ is reachable from $\langle b_{init}, en, \mathbf{v}_0, \mathbf{v}_0 \rangle$ in $\mathcal{T}'$. Lemma 3, together with definition of $\mathcal{T}'$, ensure that there exists a summary node $(en', ex)_b$ locally reachable from en' in the additional component A'_{init} of $\mathcal{T}'$, and the thesis follows. In the second case, there exists a valuation $\mathbf{v}$ such that $gs = \langle b_{init}, u, \mathbf{v}_0, \mathbf{v} \rangle$ is reachable from $\langle b_{init}, en, \mathbf{v}_0, \mathbf{v}_0 \rangle$ in $\mathcal{T}'$, and the thesis follows immediately from Lemma 3.

In the inductive case, let $gs_0 = \langle b_{init}, en, \mathbf{v}_0, \mathbf{v}_0 \rangle$ be the initial global state corresponding to g_0. For valuations $\mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s$ (with $\mathbf{v}_0$ is the initial valuation) and $\mathbf{v}$, the global state $gs_n = \langle b_{init} b_2 \cdots b_s, u, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s, \mathbf{v} \rangle$ is reachable in $\mathcal{T}'$ from gs_0. Let $\lambda = gs_0 \to gs_1 \cdots \to gs_n$ be the witnessing run, and h be the maximal index such that the h-th global state in λ is of the form $gs_h = \langle b_{init} b_2 \cdots b_s, en_s, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s, \mathbf{v}_0 \rangle$. Hence, gs_h corresponds to the last call to the box b_s, with invocation history $b_{init} b_2 \cdots b_s$, from the machine A'_{j_s} in λ. Maximality of h ensures that all global states in the sub-run λ_h from gs_h to gs_n share the same invocation history prefix $b_{init} b_2 \cdots b_s$ and the same valuation history prefix $\mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s$. Moreover, $H(gs_h, s-1) = \langle b_s, en_s, \mathbf{v}_s, \mathbf{v}_0 \rangle$ and $H(gs_n, s-1) = \langle b_s, u, \mathbf{v}_s, \mathbf{v} \rangle$, and the sequence $H(\lambda_h, s-1)$ is also a run in $\mathcal{T}'$ from $H(gs_h, s-1)$ to $H(gs_n, s-1)$. By Lemma 3 we obtain that $\langle u, \mathbf{v} \rangle$ is reachable from $\langle en_s, \mathbf{v}_0 \rangle$ in the TA $\mathcal{A}^{(\hat{k})}_{j_s}$, hence u is locally reachable from en_s in A'_{j_s}.

The transition from gs_{h-1} to gs_h is a Call transition of the form $t = u' \xrightarrow{\varphi, True}_{\emptyset, X} (b_s, en_s)$, and let $gs_{h-1} = \langle b_{init} b_2 \cdots b_{s-1}, u', \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_{s-1}, \mathbf{v}'' \rangle$, for some node u' and valuation $\mathbf{v}''$. The induction hypothesis ensures that there exists a sequence of entry nodes $en_1, \ldots, en_{s-1}$, with $en_j \in En_{i_j}$ such that: for all $1 \leq j < s-1$, (b_{j+1}, en_{j+1}) is reachable from en_j, following only Reset transitions except for the last Call transition leading to (b_{j+1}, en_{j+1}) in A'_j and u' is locally reachable from en_{s-1}, if $u' \in N_{j_{s-1}}$, or the summary node $(en', ex')_{b'}$ (with $b' \in B_{j_{s-1}}$) is locally reachable from en_{s-1}, if $u' = (b', ex')$.

In order to conclude the thesis, we need to show that (b_s, en_s) is reachable from en_{s-1}, following only Reset transitions except for the last Call transition. If $u' \in N_{j_{s-1}}$, then the desired Call transition is t itself. If, on the other hand, $u' = (en', ex')_{b'}$ then the Call transition $t' = (en', ex')_{b'} \xrightarrow{\varphi, True}_{\emptyset, X} (b_s, en_s)$ belongs to δ'_{j_s}, by construction of $\mathcal{T}'$, and we obtain the thesis.

($\Leftarrow$). For the other direction, notice that from the assumption and Lemma 2, all the Call nodes (b_{j+1}, en_{j+1}), with $1 \leq j < s-1$, are reachable in $\mathcal{T}$ from the entry nodes en_j along a run λ_j from $\langle b, en_j, \mathbf{v}, \mathbf{v}_0 \rangle$ to $\langle b\, b_{j+1}, en_{j+1}, \mathbf{v}\mathbf{v}', \mathbf{v}_0 \rangle$, where $\mathbf{v}_0$ is the initial valuation, b an arbitrary box name and $\mathbf{v}$ an arbitrary valuation. Clearly, by replacing b with the invocation history $b_{init} b_2 \cdots b_j$ and $\mathbf{v}$ with the valuation history $\mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_j$ in all the states in λ_j (with $1 \leq j \leq s-1$), we obtain a run λ'_j reaching $\langle b_{init} b_2 \cdots b_j b_{j+1}, en_{j+1}, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_j\mathbf{v}', \mathbf{v}_0 \rangle$ from $\langle b_{init} b_2 \cdots b_j, en_j, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_j, \mathbf{v}_0 \rangle$ in $\mathcal{T}$. Let us consider the two cases.

If $u' \in N_{j_s}$, again Lemma 2 ensures that $\langle b, u', \mathbf{v}, \mathbf{v}'' \rangle$ is reachable from $\langle b, en_s, \mathbf{v}, \mathbf{v}_0 \rangle$ in $\mathcal{T}$ along some run λ_s and, therefore, $\langle b_{init} b_2 \cdots b_s, en_s, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s, \mathbf{v} \rangle$ from $\langle b_{init} b_2 \cdots b_s, u, \mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s, \mathbf{v}_0 \rangle$ along a run λ'_s, obtained by replacing b and $\mathbf{v}$ with the same invocation history $b_{init} b_2 \cdots b_s$ and valuation history $\mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s$, respectively, in all the states occurring in λ_s. It is immediate to see that the concatenation of all those runs is indeed a run of $\mathcal{T}$ and the thesis follows.

If, however, $u' = (b_{s+1}, ex')$ and the summary node $(en', ex')_{b_{s+1}}$ is reachable in $\mathcal{T}'$ from en_s, then the witnessing run λ contains at least a summary transition, namely the last one.

We proceed by induction on the length h of the chain of consecutive summary transition ending in $(en', ex')_{b_{s+1}}$. If $h = 1$, then the last transition of λ exits from a node $q \in N_{j_s}$. Lemma 2 ensures that q is reachable from en_s in $\mathcal{T}$ and, by construction of $\mathcal{T}'$, there must be a Call transition from q to (b_{j+1}, en') in $\mathcal{T}$ which is enabled whenever the summary transition is. Moreover, for that summary transition to be added, (b_{s+1}, ex') must be reachable from en' in $\mathcal{T}$, hence the thesis.

If $h > 1$, then the last transition of λ exits from some summary node $q = (en'', ex'')_{b''}$. The induction hypothesis ensures that (b'', ex'') is reachable from en_s in $\mathcal{T}$ along some run $\hat{\lambda}$. Since there is a summary transition from $(en'', ex'')_{b''}$ to $(en', ex')_{b_{s+1}}$, the construction of $\mathcal{T}'$ (see second item in the definition of $SumTr(i,k)$ in Section 6.3) ensures that: *(i)* $\mathcal{T}$ contains a Call transition t from (b'', ex'') to (b_{s+1}, en'); and *(ii)* (b_{s+1}, ex') is reachable from en' in $\mathcal{T}$ along a run λ'. Therefore, (b_{s+1}, ex') is reachable from (b'', ex'') as well, by the run $\langle b, (b_{s+1}, ex'), \mathbf{v}, \mathbf{v}' \rangle \xrightarrow{t} \lambda'$. Finally, by composing the reachability witnesses we obtain the desired run $\lambda_s = \hat{\lambda} \xrightarrow{t} \lambda'$ in $\mathcal{T}$. Again, the thesis follows by simply replacing b and $\mathbf{v}$ with the same invocation history $b_{init} b_2 \cdots b_s$ and valuation history $\mathbf{v}_0\mathbf{v}_2 \cdots \mathbf{v}_s$, respectively, in all the states in run λ_s and concatenating the resulting $s+1$ witnessing runs. □

References

[1] P.A. Abdulla, M.F. Atig, J. Stenman, Dense-timed pushdown automata, in: Proceedings of the 27th ACM/IEEE Symposium on Logic in Computer Science, 2012, pp. 35–44.
[2] R. Alur, D.L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235, http://dx.doi.org/10.1016/0304-3975(94)90010-8.
[3] R. Alur, P. Madhusudan, Decision problems for timed automata: a survey, in: Proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication and Software System: Real Time, SFM-RT 2004, 2004, pp. 1–24.
[4] R. Alur, C. Courcoubetis, D. Dill, Model-checking in dense real-time, Inform. and Comput. 104 (1) (1993) 2–34, http://dx.doi.org/10.1006/inco.1993.1024.
[5] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T.W. Reps, M. Yannakakis, Analysis of recursive state machines, ACM Trans. Program. Lang. Syst. 27 (4) (2005) 786–818, http://dx.doi.org/10.1145/1075382.1075387.
[6] M. Benerecetti, S. Minopoli, A. Peron, Analysis of timed recursive state machines, in: Proceedings of the 17th International Symposium on Temporal Representation and Reasoning, TIME'10, 2010, pp. 61–68.
[7] B. Bérard, P. Gastin, A. Petit, On the power of non-observable actions in timed automata, in: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96, 1996, pp. 257–268.
[8] B. Bérard, A. Petit, V. Diekert, P. Gastin, Characterization of the expressive power of silent transitions in timed automata, Fund. Inform. 36 (2–3) (1998) 145–182, http://dx.doi.org/10.3233/FI-1998-36233.
[9] B. Bérard, S. Haddad, M. Sassolas, Interrupt timed automata: verification and expressiveness, Form. Methods Syst. Des. 40 (1) (2012) 41–87.
[10] A. Bouajjani, R. Echahed, R. Robbana, Verification of context-free timed systems using linear hybrid observers, in: Proceedings of the 6th International Conference on Computer Aided Verification, CAV'94, Springer, 1994, pp. 118–131.
[11] A. Bouajjani, R. Echahed, P. Habermehl, On the verification problem of nonregular properties for nonregular processes, in: Proceedings of the 10th ACM/IEEE Symposium on Logic in Computer Science, 1995, pp. 123–133.
[12] A. Bouajjani, R. Echahed, R. Robbana, On the automatic verification of systems with continuous variables and unbounded discrete data structures, in: Hybrid Systems II, Springer, 1995, pp. 64–85.
[13] P. Bouyer, C. Dufourd, E. Fleury, A. Petit, Updatable timed automata, Theoret. Comput. Sci. 321 (2–3) (2004) 291–345, http://dx.doi.org/10.1016/j.tcs.2004.04.003.
[14] F. Cassez, K.G. Larsen, The impressive power of stopwatches, in: Proceedings of the 11th International Conference on Concurrency Theory, CONCUR'00, 2000, pp. 138–152.
[15] Z. Dang, Pushdown timed automata: a binary reachability characterization and safety verification, Theoret. Comput. Sci. 302 (1–3) (2003) 93–121, http://dx.doi.org/10.1016/S0304-3975(02)00743-0.
[16] Z. Dang, T. Bultan, O.H. Ibarra, R.A. Kemmerer, Past pushdown timed automata and safety verification, Theoret. Comput. Sci. 313 (1) (2004) 57–71, http://dx.doi.org/10.1016/j.tcs.2003.10.004.
[17] T.A. Henzinger, P.W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, J. Comput. System Sci. 57 (1) (1998) 94–124, http://dx.doi.org/10.1006/jcss.1998.1581.
[18] K.G. Larsen, P. Pettersson, W. Yi, Uppaal in a nutshell, Int. J. Softw. Tools Technol. Transf. 1 (1–2) (1997) 134–152, http://dx.doi.org/10.1007/s100090050010.
[19] G. Li, X. Cai, M. Ogawa, S. Yuen, Nested timed automata, in: Proceedings of the 11th International Conference on Formal Modelling and Analysis of Timed Systems, FORMATS'13, 2013, pp. 168–182.
[20] J. McManis, P. Varaiya, Suspension automata: a decidable class of hybrid automata, in: Proceedings of the 6th International Conference on Computer Aided Verification, CAV'94, Springer, 1994, pp. 105–117.
[21] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall Series in Automatic Computation, Prentice-Hall, 1967.
[22] A. Trivedi, D. Wojtczak, Recursive timed automata, in: Proceedings of the 8th International Symposium Automated Technology for Verification and Analysis, ATVA'10, 2010, pp. 306–324.
[23] S. Yovine Kronos, A verification tool for real-time systems, Int. J. Softw. Tools Technol. Transf. 1 (1–2) (1997) 123–133, http://dx.doi.org/10.1007/s100090050009.