Analysis of Timed Recursive State Machines

Massimo Benerecetti  
Università degli studi di Napoli  
Federico II  
Napoli, Italy  
bene@na.infn.it

Stefano Minopoli  
Università degli studi di Napoli  
Federico II  
Napoli, Italy  
minopoli@na.infn.it

Adriano Peron  
Università degli studi di Napoli  
Federico II  
Napoli, Italy  
peron@na.infn.it

Abstract—The paper proposes a temporal extension of Recursive State Machines (RSMs), called Timed RSMs (TRSMs). A TRSM is an indexed collection of Timed Automata allowed to invoke other Timed Automata (procedural calls). The classes of TRSMs are 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. A number of subclasses of TRSMs and EPTAs are considered and compared through bisimulation of their timed LTSs. It is shown that EPTAs and TRSMs can be used to recognize classes of 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, but decidable for meaningful subclasses. The complexity is stated for a TRSMs subclass.

Keywords—Recursive State Machines; Timed Automata; Real Time Systems; Context Free Languages.

I. INTRODUCTION

The formalism of Recursive State Machines (RSMs) has been introduced in [1] to model control flow in typical sequential imperative programming languages with recursive procedure calls. A RSM is an indexed collection of finite state machines (components). Components enhance the power of ordinary state machines by allowing vertices which correspond to invocations of other components in a potentially recursive manner. As shown in [1], RSMs are closely related to 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 (TA) [2] are the reference framework to model real time systems. In this paper we consider a real time extension of RSMs (Timed RSMs or TRSMs), which allows to model real time recursive systems. Roughly speaking, a TRSM is an indexed collection of Timed Automata, with the additional ability of allowing states corresponding to invocations of other timed components. All the timed components refer to a common set of clocks used to constrain the behavior. Within a timed component, the only explicit update of clocks is the standard operation of clock reset. In addition, clock updates occur when transitions, corresponding to invocations of other components or returns from components, are performed. In particular, at invocation time one can choose the subset of clocks to reset, and at return time one can choose to restore a subset of clocks 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., [7], [4]). For instance, [4] proposes the formalism of Pushdown Timed Automata (PTA), which are Timed Automata augmented with a pushdown control stack. However, besides that fact that TRSMs, similarly to RSMs, are more appropriate than PTAs for visual modeling, it seems that PTAs are not expressive enough to account for storing and restoring clock values. Indeed, the control stack of PTAs allows to trace the history of component invocation but cannot be exploited to record the history of clock values stored at invocation time and which are needed at the matching returns for restoration. For instance, the store/restore of clock values in correspondence of invocations and returns allows to model in a very natural way a notion of time local to a component. In other words, TRSMs can manage an evolution of time within a component, abstracting away the elapse of time within the invoked components. Since the number of recursive invocation can be unbounded, it seems that this notion of local time cannot be modeled by PTAs without an unbounded number of clocks. In the paper we also lift to the timed setting the correspondence between RSMs and PDSs, as stated in [1]. Since PTAs are not expressive enough to account for TRSMs, we propose Extended PTAs, which augment PTAs with an additional stack. The additional stack is used to save/restore 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, ETPAs, besides the standard clock reset operations, also allows for store and restore operations on clock valuations. We shall prove that TRSMs are equivalent (via weak timed bisimulation) to a syntactic restriction of EPTAs (i.e.: EPTA_2). It is...

1Partially supported by Italian MIUR Project “Integrating automated reasoning in model checking: towards push-button formal verification of large-scale and infinite-state systems.”
interesting to notice that EPTAs seem to be also 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 shall provide an example of a context-free timed language with mirror distribution of symbols and of temporal delays between consecutive symbols.

The main technical contributions of the paper are decidability and complexity results for the reachability problem in TRSMs and EPTAs. In particular, we show that the problem is undecidable for the general classes of TRSMs and EPTAs. However, decidability can be recovered by forcing to restore all the clock values at invocation time. The class of TRSMs satisfying this restriction is called TRSM_1. The equivalent class of EPTAs, which can be characterized with suitable syntactical restrictions, is denoted by EPTA_1. The complexity of reachability is given for the subclass TRSM_0 of TRSM_1, which further forces to reset all the clock values at invocation time. For this class, the reachability problem is shown to be PSPACE-complete, as in classical TA. The paper also provides a comprehensive picture of the expressive hierarchy and equivalence among TRSMs and EPTAs.

The structure of the paper is as follows. Section 1 defines syntax and the semantics of TRSMs. Section 2 introduces syntax and semantics of EPTAs. Section 3 establishes the correspondence between subclasses of TRSMs and EPTAs. Section 4 is devoted to the decidability and complexity analysis of the reachability problem for TRSMs and EPTAs.

II. TIMED RECURSIVE STATE MACHINES

In this section we define syntax and semantics of TRSMs. 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 preliminary recall some standard notions of Timed Automata.

A dense clock is a variable over a dense domain \(D \geq 0\) (as usual non negative reals \(R \geq 0\) or rationals \(Q \geq 0\)). A clock valuation is a function \(v: X \rightarrow D \geq 0\). For a set of clocks \(X\), the set of clock constraints \(C(X)\) is defined by the following grammar:

\[
\varphi ::= x \sim c \mid x - y \sim c \mid \varphi \lor \varphi \mid \varphi \land \varphi
\]

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

Following [2], we write \(v \in \varphi\) when the clock valuation \(v\) satisfies the clock constraint \(\varphi \in C(X)\). The notion of constraint satisfaction by a clock valuation is defined in the standard way (e.g., see [2]). For \(t \in D \geq 0\), \(v + t\) denotes the valuation \(v'\) such that \(v'(x) = v(x) + t\), for all \(x \in X\) (clock progress). Given \(r \subseteq X\), \(v \downarrow r\) denotes the valuation resulting from the reset of the clocks in \(r\), namely:

\[
(v \downarrow r)(i) = \begin{cases} 
0 & \text{if } i \in r \\
v(i) & \text{otherwise}.
\end{cases}
\]

Moreover, given the valuations \(v, \nu\) and a set \(r \subseteq X\), \(v \uparrow (r, \nu)\) denotes the valuation resulting from the update of the clocks in \(r\), according to \(\nu\), namely:

\[
(v \uparrow (r, \nu))(i) = \begin{cases} 
\nu(i) & \text{if } i \in r \\
v(i) & \text{otherwise}.
\end{cases}
\]

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 sets 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 an index of a component. There are four kinds of transitions: (i) internal transitions connecting nodes of the same component; (ii) call transitions which lead to a box and an entry node of the called component; (iii) return transitions which lead from a box and an exit node of the callee to a node of the caller; (iv) return-and-call transitions which combines a return and a call transitions. Transitions are decorated with clock constraints as in TA. Internal transitions can only reset clocks as in TA. 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.

**Definition 1:** For an alphabet \(\Sigma\), a TRSM 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 \(\langle N_i \cup B_i, Y_i, En_i, Ex_i, \delta_i \rangle\) where:

- \(N_i\) and \(B_i\) are the disjoint sets of nodes and boxes, respectively;
- \(Y_i : B_i \rightarrow \{1, \ldots, n\}\) assigns to every box the index of a component;
- \(En_i \subseteq N_i, Ex_i \subseteq N_i\) are the sets of entry and exit nodes, respectively;
- \(\delta_i \subseteq (N_i \cup \text{Retns}_{i}) \times \Sigma \cup \{\tau\} \times C(X) \times 2^X \times 2^X \times (N_i \cup \text{Calls}_{i})\) is the transition relation, where \(\text{Calls}_{i} = \{(b, en) : b \in B_i, en \in En_{i(b)}\}\), \(\text{Retns}_{i} = \{(b, ex) : b \in B_i, ex \in Ex_{i(b)}\}\).

For each transition \(\langle u_{1, i}, \sigma, \varphi, r_1, r_2, u_2 \rangle \in \delta_i\):

- \(u_1\) (resp. \(u_2\)) are the source (resp. target) of the transition;
- \(\sigma \in \Sigma \cup \{\tau\}\) is an input symbol or the silent action;
- \(\varphi \in C(X)\) is a constraint over the clocks in \(X\);
- \(r_1\) (resp. \(r_2\)) is the subset of clocks to reset (resp. restore).

We use the following abbreviations: \(N = \bigcup_i N_i, \text{Calls} = \bigcup_i \text{Calls}_{i}, \text{Retns} = \bigcup_i \text{Retns}_{i}, \text{En} = \bigcup_i \text{En}_i\) and \(\text{Ex} = \bigcup_i \text{Ex}_i\). In order to avoid confusion with the empty stack symbol, we use the symbol \(\tau\) to represent the internal silent
A global state of a TRSM \( T=(A_1, \ldots, A_n, X) \) is a tuple 
\( gs=(b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \), where \( b_1, \ldots, b_s \) are boxes, \( u \) is a node and \( v_1, \ldots, v_s, v \) are clock valuations. 
Intuitively, \( b_1, \ldots, b_s \) represent the history of the invoked components, \( v_1, \ldots, v_s \) are the corresponding valuations stored at invocation time, \( u \) is the current node, and \( v \) is the current clock valuation. A global state \( gs=(b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) such that \( b_i \in B_j, \) for \( 1 \leq i \leq r \) and \( u \in N_j \) is well-formed if \( Y_j(b_i)=y_{i+1} \) for \( 1 \leq i < r \) and \( Y_j(b_r)=r \). The set of well-defined global states is denoted by \( GS \).

The semantics of a TRSM over the alphabet \( \Sigma \) is given by the timed Labelled Transition System (timed LTS) \( \langle GS, GS_0, \Sigma_2^D, \Delta \rangle \), where:

- \( GS_0 \subseteq GS \) is the set of states of the form \( \langle \epsilon, u, v_0 \rangle \), with \( u \in E \) and \( v_0(x)=0 \), for all \( x \in X \);
- \( \Sigma_2^D \) is the set \( \Sigma \cup \{ \tau \} \cup \{ t \} t \in D^\geq \), where \( \Sigma \) and \( D^\geq \) are disjoint;
- \( \Delta \subseteq GS \times \Sigma_2^D \times GS \) is the transition relation. For \( gs=(b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \), with \( u \in N_j \) and \( b_i \in B_m \), \( (gs, \sigma, gs') \in \Delta \) whenever one of the following holds:

1. **progress transition**: \( \sigma = \tau(t) \) and \( gs' = (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v') \), with \( v' = v + t \) and \( t \in D^\geq \);
2. **reset transition**: if \( \langle u, \sigma, \varphi, r_1, 0, u' \rangle \in \delta_j \), with \( u' \in N_j \) and \( \varphi \) then \( gs' = (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v') \), with \( v' = v \uparrow r_1 \);
3. **call transition**: if \( \langle u, \sigma, \varphi, r_1, 0, b', en \rangle \in \delta_j \), with \( b' \in B_j \) and \( \varphi \) then \( gs' = (b_1, \ldots, b_s, b', en, v_1, \ldots, v_s, v, v') \), with \( v' = v \downarrow r_1 \);
4. **return transition**: if \( \langle b_1, u, \sigma, \varphi, r_1, r_2, u' \rangle \in \delta_m \), with \( u' \in N_m, u \in E \), and \( \varphi \) then \( gs' = (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v') \), with \( v' = v \uparrow (r_2, v_1) \);
5. **return-and-call transition**: if \( \langle b_s, u, \sigma, \varphi, r_1, r_2, b', en \rangle \in \delta_m \), with \( b' \in B_m, u \in E \), and \( \varphi \) then \( gs' = (b_1, \ldots, b_s-1, b', en, v_1, \ldots, v_s-1, v', v') \), with \( v' = v \uparrow (r_2, v_1) \) and \( v' = v' \uparrow r_1 \).

A progress transition occurs when the control remains in the same vertex and there is only a clock progress. A reset transition occurs when there is an internal transition inside a component, which can possibly reset a subset of clocks. A call transition occurs when a box is entered and a component is invoked. In this case the current clock valuation is stored and a subset of clocks are possibly reset. A return transition occurs when the control returns to the invoking component. In this case, the valuation at invocation time may be restored for that subset of clocks.
power and decidability properties.

TRSM_1: the class of TRSMs, where return and return-and-call transitions are of the form \((b, ex), \sigma, \varphi, r_1, X, u_2\), with \(u_2 \in N \cup \text{Calls}\) (the whole set of clocks is restored at every return);

TRSM_0: the class of TRSM_1, where call and return-and-call transitions are of the form \((u_1, \sigma, \varphi, X, \emptyset, b, en)\) and \((b', ex), \sigma, \varphi, X, (b, en)\), respectively (the whole set of clocks is reset at every call).

Unlike the general TRSMs class, where subsets of clock values can be restored, in the TRSM_1 and TRSM_0 subclasses the entire set of clocks are restored at return time from the invoked component. Therefore, the clock valuations at return time are the same clock valuations at invocation time. This allows to model in a very natural way a notion of time within an invoked component. In the TRSM_0 class this abstraction is even stronger, because all clocks are reset at invocation time. Notice that, despite the fact that TRSM_1 and TRSM_0 abstract away the time spent into the invoked components, it does not seems possible to model them with PTAs. Indeed, since the number of recursive invocation can be unbounded, an unbounded number of clocks would be necessary.

Notice that the TRSM of Example 1 belongs to the class TRSM_0.

III. Extended Pushdown Timed Automata

In this section we introduce syntax and semantics of Extended Pushdown Timed Automata (EPTAs). An EPTA is a Pushdown Automaton enriched with a set of clocks and with an additional stack used to store/restore clock valuations. 1

Definition 2: An EPTA \(\mathcal{P}\) over \(\Sigma \cup \{\tau\}\) is a tuple \((Q, q_0, X, \Gamma, T)\), where:

- \(Q\) is a finite set of states and \(q_0 \in Q\) is the initial state;
- \(X\) is a finite set of clocks and \(\Gamma\) is a finite stack alphabet;
- \(T \subseteq Q \times \Sigma \cup \{\tau\} \times \Gamma \cup \{\epsilon\} \times \{\epsilon\} \cap \Gamma \cup \Gamma^2\) \(\times C(X) \times 2^X \times Op \times Q\) is the transition relation, with \(Op=\{\text{Reset}, \text{Store}, \text{Restore}\}\).

Intuitively, for each transition of the form \((q_1, \sigma, \gamma_1, \gamma_2, \varphi, r, op, q_2) \in T\), \(q_1\) (resp.: \(q_2\)) is the source (resp.: the target) state; \(\sigma\) is the input symbol; \(\gamma_1\) is the symbol on top of the control stack (\(\epsilon\) denotes the empty stack); \(\gamma_2\) is the string of length at most 2 which replaces the symbol on top of the control stack; \(\varphi\in C(X)\) is a clock constraint; \(op\) is the operation requested on the set of clocks. If \(op\in\{\text{Reset, Store}\}\), then \(r\) indicates the set of clocks to reset. In addition, if \(op=\text{Store}\), the current clock valuation is stored on top of the valuation stack. If \(op=\text{Restore}, r\) indicates the set of clocks to restore, using the clock valuation on top of the valuation stack.

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

The semantics of EPTAs is given by a timed LTS \(\langle GC, GC_0, \Sigma_{\mathcal{T}}^{D=\geq 0}, \Delta, \rangle\), where:

- \(GC_0 = \{\langle q_0, v_0, e, r\rangle\}\), with \(v_0(x) = 0\), for every \(x \in X\);
- \(\Delta \subseteq GC \times \Sigma_{\mathcal{T}}^{D=\geq 0} \times GC\) is the transition relation. For \(gc = \langle q, v, w, v_s, d\rangle, \langle gc, r, gc'\rangle \in \Delta\) whenever one of the following holds:
  1) progress transition: \(\sigma = \tau(t)\) and \(gc' = \langle q, v', w, v_s, d\rangle\), with \(v' = v + t\) and \(t \in \mathbb{D}^{\geq 0}\);
  2) reset transition: if \(\langle q, \sigma, a, \gamma, \varphi, r, \text{Reset}, q'\rangle \in T\), with \(\varphi \in \varphi\) and \(w = a \cdot w'\), then \(gc' = \langle q', \gamma, \varphi^*, w, v_s, d\rangle\) with \(v' = \varphi \downarrow t\);
  3) store transition: if \(\langle q, \sigma, a, \gamma, \varphi, r, \text{Store}, q'\rangle \in T\), with \(\varphi \in \varphi\) and \(w = a \cdot w'\), then \(gc' = \langle q', \gamma^*, \varphi, w', v, v_s, d\rangle\) with \(v' = \varphi \uparrow t\);
  4) restore transition: if \(\langle q, \sigma, a, \gamma, \varphi, r, \text{Restore}, q'\rangle \in T\), with \(\varphi \in \varphi\) and \(w = a \cdot w'\), then \(gc' = \langle q', \gamma^*, w, v, v_s, \cdot, d\rangle\), with \(v' = v \uparrow r\).

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, pushes a string of symbols on the control stack, and resets a set of clocks, 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 on the valuation stack. A restore transition pops a control symbol from the stack, pushes a string on the control stack, and pops a valuation from the valuation stack, restoring a subset of the clock values according to the popped clock valuation. The notions of (initial) run and reachability among pairs of states can be defined exactly as in the case of TRSMs. \(A_{\mathcal{P}}\) denotes the set of runs and \(ge \rightarrow \gamma\) denotes a pair of reachable states.

Example 2: In Fig. 2 we show an EPTA whose behavior is triggered by a timed sequence of symbols having the form: \((a_1t_1), \ldots, (a_n, t_n)(b, t_{n+1}), \ldots, (b, t_{2n})\) such that \(\Delta_{2n} - \Delta_1 = t_{n+1} - t_i\). Notice that the untime part of the sequence above describes a context-free language (i.e. \(L = \{a^n b^n : n \geq 1\}\)). As for the timed part, the sequence is required to satisfy a mirror distribution of the delays between consecutive symbols. 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 ([4]) where there is no mean to check context free properties on times.
As in the case of TRSMs, we introduce three subclasses of EPTAs by suitably constraining the operations associated with transitions. These subclasses will be naturally related with the subclasses of the TRSMs defined in the previous section. The restrictions tightly couple the type of operations performed on the control stack and on the valuation stack.

**EPTA\_2** is the subclass of EPTAs where:
- reset transitions have the form $\langle q_1, \sigma, a_1, a_2, \varphi, r, \text{Reset}, q_2 \rangle$, with $a_1 \in \Gamma \cup \{\epsilon\}$, $a_2 \in \Gamma$ (only a swap on the top of the control stack is allowed);
- store transitions have the form $\langle q_1, \sigma, a, \gamma, \varphi, r, \text{Store}, q_2 \rangle$ with $a \in \Gamma \cup \{\epsilon\}$ and $\gamma \in \Gamma^2$ (only a swap followed by a push is allowed on the control stack);
- restore transitions have the form $\langle q_1, \sigma, a, e, \varphi, r, \text{Restore}, q_2 \rangle$, with $a \in \Gamma$ (only a pop operation is allowed on the control stack);

**EPTA\_1** is the subclass of EPTA\_2 where:
- restore transitions have the form $\langle q_1, \sigma, a, e, \varphi, X, \text{Restore}, q_2 \rangle$, with $a \in \Gamma$ (the whole set of clocks is restored);

**EPTA\_0** is the subclass of EPTA\_1 where:
- store transitions have the form $\langle q_1, \sigma, a, \gamma, \varphi, X, \text{Store}, q_2 \rangle$, with $a \in \Gamma \cup \{\epsilon\}$, $\gamma \in \Gamma^2$ (the whole set of clocks is reset);

Notice that the EPTA of the Example 2 belongs to the class EPTA\_2.

**IV. RELATIONSHIP BETWEEN TRSMs AND EPTAs**

In this section we investigate the relationship between TRSMs and EPTAs and their subclasses. In order to show such a correspondence, we shall use the standard notion of (timed) weak bisimulation between timed LTS. The general picture is reported in Table I.

For the definition of weak bisimilarity for timed LTSs we follow the approach in [3]. Notice that bisimulation for Timed LTS is a nontrivial generalization of the untimed case, and requires some additional technicalities. Formally an LTS $L = (S, S_0, \Sigma, \Delta)$ *simulates* a transition system $L' = (S', S'_0, \Sigma, \Delta')$ if there exists a simulation relation $\prec \subseteq S \times S'$ defined as follows:

![Diagram](image)

**Figure 2. An example of EPTA**

Notice that the relation $\prec^{-1}$, defined as $x \prec^{-1} y \iff y \prec x$, is also a simulation relation, then $\prec$ is called a *bisimulation relation*. Two LTSs $L$ and $L'$ are strongly bisimilar, written $L \equiv_s L'$, if there is a bisimulation relation between $L$ and $L'$.

In order to introduce a notion of weak bisimulation we suitably transform a timed transition system by abstracting sequences of $\tau$-transitions. For a timed LTS $L$, a *delay execution* is a run of the form $s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \ldots \xrightarrow{\sigma_n} s_n$, such that, for every $1 \leq i \leq n$, either $\sigma_i = \tau$ or $\sigma_i = \tau(t_i)$, for some $t_i \in D^{\geq 0}$. The abstract transition system associated to $L$ is $L_{abs} = (S, S_0, \Sigma_{\tau}^{\geq 0} \setminus \{\tau\}, \Rightarrow)$, where:

- $s \Rightarrow s'$ if $a \in \Sigma$ and there exists $s'' \in S$, $s \xrightarrow{\tau^*} s'' \xrightarrow{a} s'$;
- $s \Rightarrow (t) s'$ if there exists a delay execution $s = s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \ldots \xrightarrow{\sigma_n} s_n = s'$ such that $t = \sum t_i | \sigma_i = \tau(t_i)$.

The relation $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. The transition system $L_{abs}$ abstracts all the silent actions of $L$. The relation $\Rightarrow^*$ thus corresponds to $\Rightarrow^0$. Notice also that the relation $\Rightarrow$ only abstracts silent actions that can be done before $a$. Of course, if $L$ is the timed LTS of a TRSM or of a EPTA without silent actions, then $L$ and $L_{abs}$ are identical.

Two LTS $L$ and $L'$ are weakly bisimilar, written $L \equiv_w L'$, if there exists a bisimulation relation between $L_{abs}$ and $L_{abs}'$.

The following theorem shows the tight corresponding between TRSMs and EPTAs and their subclasses.

**Theorem 1:** For any TRSM (resp.: TRSM\_1, TRSM\_0) $\overline{T}$, there exists a weakly similar EPTA\_2 (resp.: EPTA\_1, EPTA\_0) $\overline{P}$ and vice versa.

**Proof:**

We first show the bisimulation relation between TRSM and EPTA\_2.

Given a TRSM $T = (A_1, \ldots, A_n, X)$ over an alphabet $\Sigma$, we first give the construction of an EPTA\_2 $\overline{P}$ over the alphabet $\Sigma \cup \{\tau\}$ that is weakly bisimilar to a $\overline{T}$. (The idea of the construction is similar to the one presented in [1] to encode RSMs into pushdown systems). In the following we assume that $A_i$ has the form $(N_i \cup B_i, Y_i, \overline{E_n}_i, \overline{E_x}_i, \overline{\delta}_i)$. $\overline{P} = (Q, \overline{q}_0, X \cup \{x^i\}, \Gamma, T)$, is defined as follows:

- $Q = \{\overline{q}_0\} \cup \{\overline{q}_i^{(k)} | 1 \leq i \leq n_{ex}, 1 \leq k \leq k_{max} \text{ and } u \in N \cup \text{Calls}\}$, where $n_{ex}$ is the greatest number of

<table>
<thead>
<tr>
<th>Expressiveness relations among TRSMs and EPTAs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPTA_0 $\subseteq$ EPTA_1 $\subseteq$ EPTA_2 $\subseteq$ EPTA</td>
</tr>
<tr>
<td>TRSM_0 $\subseteq$ TRSM_1 $\subseteq$ TRSM</td>
</tr>
</tbody>
</table>
exit nodes of all component machines and $k_{max}$ is the greatest number of return transitions exiting from every $ex \in Ex$;

- $\Gamma = \{(j,u) | 1 \leq j \leq n, u \in N_j \cup B_j\}$. A pair $(j,u)$ gives the index of a component and the current box or node in the component;

- $T$ is defined as follows:
  
  - for every pair $(j,ex_i) \in \Gamma$ with $ex_i \in Ex_j$, $(\sigma_0, (j,ex_i), (j,ex_i), True, \{x\}, Reset, q_0) \in T$;
  
  - for every reset transition of the form $(\sigma, \varphi, r_1, \emptyset, (b, en)) \in \delta_j$, where $u \notin Retns$ and $u' \notin Calls$, there is a transition $(\sigma_0, \varphi, r_1, (j,u'), \varphi, r_1, Reset, q_0) \in T$;
  
  - for every call transition $(\sigma, \varphi, r_1, \emptyset, (b, en)) \in \delta_j$ where $u \notin Retns$, $b \in B_j$, $Y_j(b)=j'$ and $en \in En_j$, there is a transition $(\sigma_0, \varphi, \delta_1, \{x\}, Reset, q_0) \in T$;

- if $(\langle b,ex_i \rangle, \sigma_k, \varphi_k, r_{1k}, r_{2k}, (b',en)) \in \delta_j$ is the k-th return transition from the exit node $ex_i$, with $u \in N_j$, $Y_j(b)=j'$, then there are the following transitions:
  
  1) $(\sigma_0, \sigma_k, (j,ex_i), (j,ex_i), (j,ex_i), True, \{x\}, Reset, q_0) \in T$;
  
  2) $(\sigma_0, \sigma_k, (j,ex_i), (j,ex_i), (j,ex_i), True, \{x\}, Reset, q_0) \in T$;

- if $(\langle b,ex_i \rangle, \sigma_k, \varphi_k, r_{1k}, r_{2k}, (b',en)) \in \delta_j$ is the k-th return-and-call transition from the exit node $ex_i$, with $u \in N_j$, $Y_j(b)=j'$ and $Y_j(b)=j''$, then there are the following transitions:
  
  1) $(\sigma_0, \sigma_k, (j,ex_i), (j,ex_i), (j,ex_i), True, \{x\}, Reset, q_0) \in T$;
  
  2) $(\sigma_0, \sigma_k, (j,ex_i), (j,ex_i), (j,ex_i), True, \{x\}, Reset, q_0) \in T$;

The weak bisimulation relation $\prec$ is defined as follows. For $gs \in GS$ and $gc \in GC$, $gs \prec gc$ if and only if one of the following holds:

1) $gs = (b_1, \ldots, b_s, u_1, \ldots, v_s, v_{s+1})$ and $gc = (\sigma_0, v_{s+1}, (j, u_i)(j_0, b_i), \ldots, v_s, v_{s+1})$, where $v_k(x) = v_k(x)$ for all $x \in X$ and $1 \leq k \leq s + 1$;

2) $gs = (b_1, \ldots, b_s, u_1, \ldots, v_s, v_{s+1})$ and $gc = (\sigma_0, v_{s+1}, (j, u_i)(j_0, b_i), \ldots, v_s, v_{s+1})$, where $v_k(x) = v_k(x)$ for all $x \in X$ and $1 \leq k \leq s$, and $v_{s+1}(x) = v_{s+1}(x)$ for all $x \in X \setminus r_{1n}$, where $r_{1n}$ is the set of clocks reset by the n-th return transition from $(b_{s+1},ex_i)$ to $u$ and $v_{s+1}(x) = 0$;

3) $gs = (b_1, \ldots, b_s, u_1, \ldots, v_s, v_{s+1})$, with $u \in En$ and $v_{s+1} = v_s \downarrow r_{1n}$, where $r_{1n}$ is the set of clocks reset by the n-th return-and-call transition from $(b_{s+1},ex_i)$ to $(b_s, u)$ and $gc = (\sigma_0, v_{s+1}, (j, u_i)(j_0, b_i), \ldots, v_s, v_{s+1})$, with $v_k(x) = v_k(x)$ for all $x \in X$ and $1 \leq k \leq s$.

We show now that $\prec$ is a bisimulation.

Let us consider the initialization requirement. An initial state of $T$ has the form $gs_0 = (en, \epsilon, v_0)$, with $en \in En_1$. The initial state of $P$ has the form $gc_0 = (q_0, v_0, (1,en), \epsilon)$. By definition of $\prec$, it is immediate to show that $gs_0 \prec gc_0$ and $gc_0 \prec^{-1} gc_0$.

Let us consider now the propagation requirement. Let us assume that $gs \prec gc$. There are the following cases:

- (progress transition): $gs \xrightarrow{t} gs'$, for some $t \in D^{\geq 0}$, where $gs' = (b_1, \ldots, b_s, u_1, \ldots, v_s, v_{s+1}+t)$. Then there exists in $L_{abs}(P)$ the progress transition $gc \xrightarrow{t} gc'$, where $gc' = (q_0, v_{s+1}+t, (j,u)(j_0, b_i), \ldots, v_s, v_{s+1})$. By definition, $gs' \prec gc'$;

- (reset transition): $gs \xrightarrow{\emptyset} gs'$, where $gs' = (b_1, \ldots, b_s, u_1, \ldots, v_s, v_{s+1}+1)$. Then there exists in $L_{abs}(P)$ the transition $gc \xrightarrow{\emptyset} gc'$, where $gc' = (q_0, v_{s+1}+1, (j,u)(j_0, b_i), \ldots, v_s, v_{s+1})$. By definition, $gs' \prec gc'$;

- (call transition): $gs \xrightarrow{(b, \sigma, \varphi, r_1, \emptyset, (b, en))} gs'$, where $gs' = (b_1, \ldots, b_s, b_{s+1}, en, v_1, \ldots, v_s, v_{s+1}) \cup \delta_j$, $(u, (\sigma, \varphi, r_1, \emptyset, (b, en))) \in \delta_j$ and $v_{s+1} \in \varphi$. Then there exists in $L_{abs}(P)$ the transition $gc \xrightarrow{(b, \sigma, \varphi, r_1, \emptyset, (b, en))} gc'$, where $gc' = (q_0, v_{s+1}+1, (j,u)(j_0, b_i), \ldots, v_s, v_{s+1})$. By definition, $gs' \prec gc'$;

- (return transition): $gs \xrightarrow{(b, u, \sigma, \varphi, \epsilon, \varphi_n \wedge (x' = 0), r_{2n})} gs'$, where $gs' = (b_1, \ldots, b_s, b_{s+1}, \emptyset, \varphi_n \wedge (x' = 0), r_{2n})$. By definition, $gs' \prec gc'$.
Now we show all the possible transitions in $L_{abs}(P)$ having $gc$ as source:

- (progress transition): $gc \xrightarrow{\ell} gc'$, for some $t \in D^{\geq 0}$, where $gc' = (q_0, v_{s+1} \downarrow \tau, (j,u)\langle j_1,b_1 \rangle \ldots \langle j_i,b_i \rangle, v_{s} \ldots v_1)$. Then there exists $L_{abs}(T)$ the progress transition $gs \xrightarrow{\ell} gs'$, where $gs' = (b_1, \ldots, b_{s-1}, u', v_1, \ldots, v_{s-1}, v_{s+1} \uparrow \tau, v_0)$. By definition, $gc' \xleftarrow{\ell} gs'$;

- (reset transition): $gc \xrightarrow{\phi} gc'$, where $gc' = (q_0, v_{s+1} \downarrow \tau, (j,u)\langle j_1,b_1 \rangle \ldots \langle j_i,b_i \rangle, v_{s} \ldots v_1), (q_0, \sigma, (j_1,u), (j_2,u), \emptyset, \{\text{Reset} \langle q' \rangle \}) \in T$ and $v_{s+1} \in \varphi$. Then there exists $L_{abs}(T)$ the transition $gs \xrightarrow{\phi} gs'$, where $gs' = (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v_{s+1} \downarrow \tau)$ and $(u, \sigma, \varphi, r, 0, u') \in \delta_j$. By definition, $gc' \xleftarrow{\phi} gs'$;

- (store transition): $gc \xrightarrow{\phi} gc'$, where $gc' = (q_0, v_{s+1} \downarrow \tau, (j_1,u), \langle j_1,b_1 \rangle \ldots \langle j_i,b_i \rangle, v_{s} \ldots v_1), (q_0, \sigma, (j,u), \langle j_ex \rangle, \emptyset, \{\text{Reset} \langle q' \rangle \}) \in T$ and $v_{s+1} \in \varphi$. Then there exists $L_{abs}(T)$ the transition $gs \xrightarrow{\phi} gs'$, where $gs' = (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v_{s+1} \downarrow \tau)$ and $(u, \sigma, \varphi, r, 0, u') \in \delta_j$. By definition, $gc' \xleftarrow{\phi} gs'$;

- (reset transition): $gc \xrightarrow{\phi} gc'$, where $gc' = (q_0, v_{s+1} \downarrow \tau, (j_1,u), \langle j_1,b_1 \rangle \ldots \langle j_i,b_i \rangle, v_{s} \ldots v_1), (q_0, \sigma, (j,u), \langle j_ex \rangle, \emptyset, \{\text{Reset} \langle q' \rangle \}) \in T$ and $v_{s+1} \in \varphi$. Then there exists $L_{abs}(T)$ the transition $gs \xrightarrow{\phi} gs'$, where $gs' = (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v_{s+1} \downarrow \tau)$ and $(u, \sigma, \varphi, r, 0, u') \in \delta_j$. By definition, $gc' \xleftarrow{\phi} gs'$;

- (call transition): $gs \xrightarrow{\phi} gs'$, where $gs' = (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \uparrow \tau)$. By definition, $gs' \xleftarrow{\phi} gc'$;
1) \( \langle q_{i(u^0)}, \tau, (j_{x+1}, b_{s+1}), (j, u), x' = 0, r_{1n}, \text{Reset, } q_0 \rangle; \)
2) \( \langle q_0, \tau, (j, u), (j, u), \text{True, } \{x'\}, \text{Reset, } q_0 \rangle; \)
3) \( \langle q_0, \sigma_k, (j, u), \epsilon, \varphi_k \wedge (x' = 0) \rangle_{[k]}, \text{Reset, } q_{(i, u')} \rangle \)

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle, \ldots \langle j_1, b_1 \rangle, \nabla_s \ldots \nabla_1 \rangle \).

By definition, \( gs' < gc' \).

- (return-and-call transition): \( gs \xrightarrow{\sigma} gs', \) where \( gs' = \langle b_1, \ldots, b_{s-1}, b_{s'}, en, v_1, \ldots, v_{s-1}, v_s, v_{s+1}, v_{s+1} \downarrow_{r_{2k}, v_2} \downarrow_{r_{2k}} \rangle, ((b, u), \sigma_k, \varphi_k, r_{1k}, r_{2k}, (b_{s'}, en)) \) in \( \delta_j \) and \( v_{s+1} \in \varphi_k \). Then there exists in \( L_{abs}(P) \) the transition \( gc \xrightarrow{\tau} gc' \),抽象着以下的过渡:

1) \( \langle q_{(i, u')}, \tau, (j_{x+1}, b_{s+1}), (j, u), x' = 0, r_{1n}, \text{Reset, } q_0 \rangle; \)
2) \( \langle q_0, \sigma, (j, u), (j', u') \cdot (j, b_{s'}), \varphi, r, \text{Store, } q_0 \rangle \)

The resulting configuration is \( gc' = \langle q_0, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle, \ldots \langle j_1, b_1 \rangle, \nabla_s \ldots \nabla_1 \rangle, \) with \( \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \in \varphi \). Then there exists in \( L_{abs}(P) \) the transition \( gs \xrightarrow{\tau} gs', \) where \( gs' = \langle b_1, \ldots, b_{s-1}, b_{s'}, \ldots \rangle \) in \( \delta_j \) and \( v_{s+1} \in \varphi_k \). By definition, \( gc' < gc \).

- (return transition with \( \tau \)): \( gc \xrightarrow{\tau} gc' \), 抽象着以下的过渡:

1) \( \langle q_{(i, u')}, \tau, (j_{x+1}, b_{s+1}), (j, u), x' = 0, r_{1n}, \text{Reset, } q_0 \rangle; \)
2) \( \langle q_0, \tau, (j, u), (j, u), \text{True, } \{x'\}, \text{Reset, } q_0 \rangle; \)
3) \( \langle q_0, \sigma_k, (j, u), \epsilon, \varphi_k \wedge (x' = 0) \rangle_{[k]}, \text{Reset, } q_{(i, u')} \rangle \)

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle, \ldots \rangle, \) \( \text{in } T \).

The resulting configuration is \( gc' = \langle q_0, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle, \ldots \rangle, \) \( \text{in } T \).

* \( \langle q_0, \sigma, (j, u), (j', u') \cdot (j_{x+1}, b_{s+1}), \varphi, r, \text{Store, } q_0 \rangle \)

The resulting configuration is \( gc' = \langle q_0, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j_{x+1}, b_{s+1} \rangle, \ldots \rangle \).

When there exists in \( P \) the transition \( gs \xrightarrow{\tau} gs', \) where \( gs' = \langle b_1, \ldots, b_{s-1}, en, v_1, \ldots, v_{s+1}, v_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \rangle \) and \( (u, \sigma, \varphi, r, \emptyset, (b_{s'}, en)) \in \delta_j \). By definition, \( gc' < gc' \) - gs' transition.

- (store transition with \( \tau \)): \( gc \xrightarrow{\tau} gc' \), 抽象着以下的过渡:

1) \( \langle q_{(i, u')}, \tau, (j_{x+1}, b_{s+1}), (j, u), x' = 0, r_{1n}, \text{Reset, } q_0 \rangle; \)
2) \( \langle q_0, \sigma, (j, u), (j, u), \text{True, } \{x'\}, \text{Reset, } q_0 \rangle; \)
3) \( \langle q_0, \sigma_k, (j, u), \epsilon, \varphi_k \wedge (x' = 0) \rangle_{[k]}, \text{Reset, } q_{(i, u')} \rangle \)

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j_{x+1}, b_{s+1} \rangle, \ldots \rangle \).

Let \( gs = \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s \rangle \), where \( r_{1n} \) is the set of clocks reset by the n-th return-and-call transition from \( (b_1, \ldots, c_{x1}) \) to \( (b_s, u) \), a global state and \( gc = \langle q_{(i, u'), \nabla_s, (j, u'), (j, u) \cdot (j, b_s), x' = 0, r_{1n}, \text{Store, } q_0 \rangle \) and \( v_{s+1} = v_s \).

- \( \text{Let } gs = \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle, \) with \( u \in E_n \) and \( v_{s+1} = v_s \downarrow_{r_{1n}}, \) where \( r_{1n} \) is the set of clocks reset by the n-th return-and-call transition from \( (b_1, \ldots, c_{x1}) \) to \( (b_s, u) \), a global state and \( gc = \langle q_{(i, u'), \nabla_s, (j, u') \cdot (j, b_s), x' = 0, r_{1n}, \text{Store, } q_0 \rangle \) and \( v_{s+1} = v_s \).

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle \).

By definition, \( gc' < gc \).

- \( \text{Let } gs = \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s \rangle, \) with \( u \in E_n \) and \( v_{s+1} = v_s \).

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle \).

By definition, \( gc' < gc \).

- \( \text{Let } gs = \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s \rangle, \) with \( u \in E_n \) and \( v_{s+1} = v_s \).

The resulting configuration is \( gc' = \langle q_{(i, u')}, \nabla_{s+1} \downarrow_{r_{1n} \cup \{x'\}} \langle j, b_s \rangle \).

By definition, \( gc' < gc \).
definition, \( gs' < gc' \): 
- (call transition): \( gs \Rightarrow gs' \), where \( gs' = (b_1, \ldots, b_{s+1}, en, v_1, \ldots, v_s, \overline{v}_{s+1}, \overline{v}_{s+1} v_1, (u, \sigma, \varphi, r, 0, \emptyset, (b_{s+1}, en)) \} \in \delta_j \) and \( v_{s+1} \in \varphi \). Then there exists in \( \mathcal{P} \) the transition \( gc \Rightarrow gc' \), abstracting the following transitions:
  * \( (q_i,(b_{s+1}), (j,t'), (j,u) \cdot (j_s,b_s), x' = 0, r_{in}, \text{Store}, q_0) \in T \) ;
  * \( (q_0, \sigma, (j,u),(j',en) \cdot (j_{s+1},b_{s+1}), \varphi, r, \text{Store}, q_0) \) in \( T \).

The resulting configuration is \( gc' = (q_0, v_s \downarrow \downarrow \uparrow, (j',en) \cdot (j_{s+1},b_{s+1}), \ldots (j_1,b_1), v_s \ldots v_1) \). By definition, \( gs' < gc' \).

Now we show all the possible transitions in \( \mathcal{L}_{abs}(\mathcal{P}) \) having \( gc \) as source:

- (progression transition): \( gc \Rightarrow gc' \), abstracting the following transitions:
  1. \( (q_i(b_{s+1}), (j,t'), (j,u) \cdot (j_s,b_s), x' = 0, r_{in}, \text{Store}, q_0) \) and
  2. the progress transition in \( q_0 \) of every unit of time.

The resulting configuration is \( gc' = (q_0, v_s \downarrow, (j,u) \cdot (j_s,b_s), v_s \ldots v_1) \). By definition, \( gc' < gc'' \).

- (reset transition): \( gc \Rightarrow gc' \), abstracting the following transitions:
  1. \( (q_i,(b_{s+1}), (j,t'), (j,u) \cdot (j_s,b_s), x' = 0, r_{in}, \text{Store}, q_0) \) ;
  2. \( (q_0, \sigma, (j,u),(j',en) \cdot (j_s,b_s), x' = 0, r_{in}, \text{Store}, q_0) \) ;

- (call transition): \( gc \Rightarrow gc' \), abstracting the following transitions:
  * \( (q_i,(b_{s+1}), (j,t'), (j,u) \cdot (j_s,b_s), x' = 0, r_{in}, \text{Store}, q_0) \) in \( T \) ;
  * \( (q_0, \sigma, (j,u),(j',en) \cdot (j_{s+1},b_{s+1}), \varphi, r, \text{Store}, q_0) \) in \( T \).

The resulting configuration is \( gc' = (q_0, v_s \downarrow, (j',en) \cdot (j_{s+1},b_{s+1}), \ldots (j_1,b_1), v_s \ldots v_1) \), with \( v_{s+1} \in \varphi \). Then there exists in \( \mathcal{L}_{abs}(\mathcal{T}) \) the transition \( gs \Rightarrow gs' \), where \( gs' = (b_1, \ldots, b_{s+1}, u, v_1, \ldots, v_s, v_{s+1} + t) \). By definition, \( gc' < -1 gs' \).

The weak bisimulation relation \( \prec \) between \( \mathcal{L}_{\mathcal{F}} \) and \( \mathcal{L}_{\mathcal{T}} \) is defined as follows. For gc in \( GC \) and gs in \( GS \), gc \( \prec \) gs if and only if:

1. gc = \( (q, v_s, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \ldots v_1) \) and gs = \( (b_0, b_1, \ldots, b_{s+1}, u, v_1, \ldots, v_s, v_{s+1}) \), with \( u \in \{ (q_1, a_0, \overline{a}_0, en(q_0, a)) \} \).

We show now that \( \prec \) is a bisimulation. Let us consider the initialization requirement. The initial state of \( \mathcal{P} \) has the form \( gc_0 = (q_0, v_0, a, c) \). An initial state of \( \mathcal{T} \) has the form \( gs_0 = (en(q_0, a), c, v_0) \), with \( en(q_0, a) \in \mathcal{E}_{\mathcal{F}} \). By definition of \( \prec \), it is immediate to show that \( gc_0 < gc_0 \) and \( gs_0 < -1 gs_0 \).

Let us consider now the propagation requirement. Let us assume that \( gc \prec gs \), there are the following cases:

- Let \( gc = (q, v_s, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \ldots v_1) \) a configuration and \( gs = (b_0, b_1, \ldots, b_{s+1}, u, v_1, \ldots, v_s, v_{s+1}) \), with \( u \in \{ en(q_0, a), en(q_0, a) \} \) a global state. By definition, \( gc < gs \). Now we show all the possible transitions in \( \mathcal{L}_{abs}(\mathcal{P}) \) having \( gc \) as source:

  - (progression transition): \( gc \Rightarrow gc' \), where \( gc' = (q, v_s + t, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \ldots v_1) \) and \( t \in D_{\emptyset} \). Then there exists in \( \mathcal{L}_{abs}(\mathcal{T}) \) the transition \( gs \Rightarrow gs' \), where \( gs' = (b_0, b_1, \ldots, b_{s+1}, u, v_1, \ldots, v_s, v_{s+1} + t) \). By definition, \( gc' < -1 gs' \).

  - (reset transition): \( gc \Rightarrow gc' \), where \( gc' = (q', v_s \downarrow, a' \cdot a_{s-1} \ldots a_1, v_{s-1} \ldots v_1) \), \( q, a_s, a', \varphi, r, \text{Reset}, q' \) in \( \mathcal{T} \) and \( v_s \in \varphi \). Then there exists in \( \mathcal{L}_{abs}(\mathcal{T}) \) the transition \( gs \Rightarrow gs' \), where \( gs' = (b_0, b_1, \ldots, b_{s+1}, v_1, \ldots, v_s, v_{s+1} \downarrow, (u, \sigma, \varphi, r, 0, 0) \in \delta_j \) and \( v_{s+1} \in \varphi \). By definition, \( gc' < -1 gs' \).

      - (store transition): \( gc \Rightarrow gc' \), where \( gc' = (q', v_s \downarrow, a'a' \cdot a_{s-1} \ldots a_1, v_{s-1} \ldots v_1) \), \( q, a_s, a', \varphi, r, \text{Store}, q' \) in \( \mathcal{T} \) and \( v_s \in \varphi \). Then there...
exists in $L_{abs}(T)$ the transition $gs \Rightarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-1}}, b_{a_s}, u', v_1, \ldots, v_{s-1}, v_s \uparrow\rangle$, $(u, \sigma, \varphi, r, \emptyset, (b, u'))$ in $T$ and $u' \in \{e, (q,a), e_{(q,a)}\}$. By definition, $gc < gc'$;  

- (restore transition): $gc \Rightarrow gc'$, where $gc' = \langle q', v_{s-1}, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

- (call transition with $\tau$): $(u \in En)$. $gs \Rightarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-2}}, u, v_1, \ldots, v_{s-2}, v_s \uparrow\rangle$, $(u, \sigma, \varphi, r, \emptyset, \tau, q')$ in $T$ and $v_s \in \varphi$. Then there exists in $L_{abs}(P)$ the transition $gc \Rightarrow gc'$, where $gc' = \langle q', v_{s-1}, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

- (return transition): $(u \in Ex)$. $gs \Rightarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-2}}, u, v_1, \ldots, v_{s-2}, v_s \uparrow\rangle$, $(u, \sigma, \varphi, r, \emptyset, \tau, q')$ in $T$ and $v_s \in \varphi$. Then there exists in $L_{abs}(P)$ the transition $gc \Rightarrow gc'$, where $gc' = \langle q', v_{s-1}, \emptyset, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

Now we show all the possible transitions in $L_{abs}(T)$ having $gs$ as source.  

- (progress transition): $gs \Downarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-1}}, u, v_1, \ldots, v_{s-1}, v_s \uparrow, t \rangle$ and $t \in D^{\geq 0}$. Then there exists in $L_{abs}(P)$ the transition $gc \Rightarrow gc'$, where $gc' = \langle q, v_s + t, a_s, a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

- (progress transition with $\tau$): $(u \in En)$. $gs \Downarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-1}}, u', v_1, \ldots, v_{s-1}, v_s \uparrow, t \rangle$, $(u, \sigma, \varphi, r, \emptyset, u') \in T$, $v_s \in \varphi$ and $u' \in En \cup Ex$. Then there exists in $L_{abs}(P)$ the transition $gc \Rightarrow gc'$, where $gc' = \langle q', v_{s-1}, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

- (reset transition): $gs \Rightarrow gs'$, where $gs' = \langle b_{a_1}, \ldots, b_{a_{s-2}}, u', v_1, \ldots, v_{s-1}, v_s \uparrow\rangle$, $(u, \sigma, \varphi, r, \emptyset, u')$ in $T$, $v_s \in \varphi$ and $u' \in En \cup Ex$. Then there exists in $L_{abs}(P)$ the transition $gc \Rightarrow gc'$, where $gc' = \langle q', v_{s-1}, a_s \cdot a_{s-1} \ldots a_1, v_{s-1} \downarrow \rangle$. By definition, $gs' < gc'$;  

V. The Reachability Problem: decidability and complexity results  

In this section we study the problem of reachability for TRSMs and EPTAs. In particular, we prove that the problem is undecidable for the general class of TRSMs and EPTAs, but that is decidable for TRSM_1 and EPTA_1. We state also the complexity of the problem for the class of TRSM_0 and EPTA_0.

Given a global state $gs = \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v \rangle$ of a TRSM $T$, we call the tuple $(b_1, \ldots, b_s, u)$ an untimed global state of $T$. The reachability problem for an untimed global state $(b_1, \ldots, b_s, u, v_1, \ldots, v_s, v)$ is reachable from an initial configuration.
state. A similar notion of reachability can be given for EPTAs.

The undecidability for TRSMs is stated by showing the undecidability of the reachability problem for the class EPTA-2. In fact, it is possible to show (see the complete work) that EPTA-2s allow to simulate increment and decrement of clocks. For instance, Fig. 3 shows an EPTA-2 which decrements the clock z, leaving the clock y unchanged (z and y can be viewed as the two counters). The path leading from \((p, q_0)\) to \((p, q_3)\), is used to store the value of \(x\) into \(x''\), while \(y\) maintains the original value. In the next step, these values are stored in the valuation stack, and after an amount of time equal to \(x'' - 1\) is elapsed, the original value of \(y\) is restored. In this way, it is possible to define an EPTA-2 which simulates a two counter machine (clocks with increment and decrement can be used as counters). This allows to reduce the halting problem of two counter Minsky machine, which is known to be undecidable [5], to the reachability problem for an EPTA-2.

**Theorem 2:** The reachability problem for EPTA-2 is undecidable.

**Proof:**

We recall that a two counter machine is a finite set of labeled instructions over two counters \(c_1\) and \(c_2\). There are two type of instructions:

- **increment** of \(x \in \{c_1, c_2\}\), of the form
  
  \[ p : x := x + 1; \text{goto } q; \]

- **decrement** of \(x \in \{c_1, c_2\}\), of the form
  
  \[ p : x > 0 \text{ then } x := x - 1; \text{goto } q_1 \text{ else goto } q_2 \]

The machine starts at an instruction labeled \(p_0\) with \(c_1 = c_2 = 0\) and stops at a special instruction labeled \(\text{HALT}\). The halting problem for a two counter machine consists in deciding whether the machine reaches the instruction \(\text{HALT}\).

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 EPTA-2 which exploit the ability of storing and restoring clock valuations on the valuation stack. The states of the automaton are the labels of the instruction augmented with additional control states, used to simulated increments and decrements. We denote by \(C_{(+)}\) (resp. \(C_{(-)}\)) the set of the labels associated to increment (resp. decrement) instructions.

Given a machine with two counters \(x\) and \(y\), the simulating EPTA-2 is \((Q, (p_0, s), X, \Gamma, T)\), where

- \(Q = C_{(+)} \times \{q_{0_+}, q_{1_+}, q_{2_+}\} \cup C_{(-)} \times \{q_{-i} : 0 \leq i \leq 6\} \cup \{\text{HALT}\}\)
- \(s = q_{0_+}\) if \(p_0 \in C_{(+)}\) and \(s = q_{0_-}\) if \(p_0 \in C_{(-)}\)
- \(X = \{x, y, x', x''\} \text{ and } \Gamma = \{\gamma\}\)
- for every \(p \in C_{(+)}\) associated with the increment of the counter \(z \in \{x, y\}\), \(T\) has the transitions:
  1. \(\langle(p, q_{0_+}), \gamma, \epsilon, \gamma, x' = 0, \emptyset, \text{Store}, (p, q_{1_+})\rangle\)
  2. \(\langle(p, q_{1_+}), \gamma, \gamma, \gamma, y' = 1, X \setminus \{z\}, \text{Restore}, (p, q_{2_+})\rangle\)
  3. \(\langle(p, q_{2_+}), \gamma, \gamma, x'' - z = 0 \land x' = 0, \{z\}, \text{Reset}, (p, q_{3_+})\rangle\)
  4. \(\langle(p, q_{3_+}), \gamma, \gamma, x' = 0, \emptyset, \text{Store}, (p, q_{4_+})\rangle\)
  5. \(\langle(p, q_{4_+}), \gamma, \gamma, y' = 1 \land x' = 0, \{x', x''\}, \text{Reset}, (p, q_{5_+})\rangle\)
  6. \(\langle(p, q_{5_+}), \gamma, \gamma, x'' = 0 \land z < 1, \{x', x''\}, \text{reset}, (p, q_{6_+})\rangle\)

7. let \(q\) be the target label pointed by the goto instruction (it depends on the chosen branch), then
   - a) if \(q \in C_{(+)}\), then \(\langle(p, q_{6_+}), \gamma, \epsilon, \gamma, x' = 0, \{x''\}, \text{Reset}, (p, q_{0_+})\rangle\) and \(\langle(p, q_{0_+}), \gamma, \gamma, x' = 0 \land z < 1, \{x''\}, \text{Reset}, (p, q_{0_+})\rangle\)
   - b) if \(q \in C_{(-)}\), then \(\langle(p, q_{6_+}), \gamma, \epsilon, x' = 0, \{x''\}, \text{Reset}, (p, q_{0_-})\rangle\) and \(\langle(p, q_{0_-}), \gamma, \gamma, x' = 0 \land z < 1, \{x''\}, \text{Reset}, (p, q_{0_-})\rangle\)
   - c) if \(q = \text{HALT}\), then \(\langle(p, q_{6_+}), \gamma, \epsilon, x' = 0, \{x', x''\}, \text{Reset}, \text{HALT}\rangle\) and \(\langle(p, q_{0_+}), \gamma, \epsilon, x' = 0 \land z < 1, \{x', x''\}, \text{Reset}, \text{HALT}\rangle\)

As consequence of Theorem 1 we have the following corollary.

**Corollary 1:** The reachability problem for TRSMs is undecidable.
We shall now prove that the reachability problem for TRSM_1 is decidable. The proof combines two techniques: the former is the standard regionalization technique, used to prove reachability in Timed Automata [2], while the latter is derived from the algorithm, based on a fix point construction, proposed in [1] to solve the reachability problem for RSMs.

Let us first recall the notion of clock region and region automaton of a Timed Automaton. Following the standard construction [6], we assume that constants occurring in the clock constraints of the automaton are integers. For any \( t \in D^{\geq 0} \), \( \{ t \} \) denotes the fractional part of \( t \), and \( \lfloor t \rfloor \) denotes the integral part of \( t \) (i.e. \( t = \{ t \} + \lfloor t \rfloor \)). For each clock \( x \in X \), let \( c_x \) be the largest integer constant \( c \) such that \( x \) is compared with \( c \) in some clock constraint appearing in a transition. The equivalence relation \( \equiv, \) (region equivalence), is defined over the set of clock valuations for \( X \). For two clock valuations \( v_1 \) and \( v_2 \), we write \( v_1 \equiv v_2 \) iff the following conditions hold:

1. for all clocks \( x \in X \), either \( \{ v_1(x) \} \) and \( \{ v_2(x) \} \) are the same, or both \( v_1(x) \) and \( v_2(x) \) exceed \( c_x \);
2. for all clocks \( x, y \) with \( v_1(x) \leq c_x \) and \( v_1(y) \leq c_y \), \( \{ v_1(x) \} \leq \{ v_1(y) \} \) iff \( \{ v_2(x) \} \leq \{ v_2(y) \} \);
3. for all clocks \( x \in X \) with \( v_1(x) \leq c_x \), \( \{ v_1(x) \} = 0 \) iff \( \{ v_2(x) \} = 0 \).

A clock region is an equivalence class of clock valuations induced by \( \equiv \). If \( k \) is the number of the clocks, there are at most \( k! \cdot 4^{k} \cdot \prod_{x \in X} (c_x + 1) \) regions (see [6]). We denote by \( \mathit{Reg} \) the set of all regions with respect to the set of clocks \( X \) and an indexed family \( \{ c_x \}_{x \in X} \).

A clock region \( \mathit{reg}' \) is a time successor of a clock region \( \mathit{reg} \) if and only if, for all clock valuations \( v \in \mathit{reg} \), there exists a \( t \in D^{\geq 0} \) such that \( v+\mathit{reg} \in \mathit{reg}' \). Let \( \varphi \in \mathcal{C}(X) \) be a clock constraint, we write \( \mathit{reg} \in \varphi \) if and only if \( \forall v \in \mathit{reg} \). Note that for a clock constraint \( \varphi \) of a TA, if \( v \equiv v' \), then \( \forall v \in \mathit{reg} \) iff \( \forall v' \in \mathit{reg} \). Let \( r \subseteq X \) a set of clocks, \( \mathit{reg} \downarrow_r = \{ v \in \mathit{reg} \mid v \downarrow_r \} \) denotes the region resulting from \( \mathit{reg} \) by resetting the clocks in \( r \).

It is well known that the reachability problem of a Timed Automaton can be reduced to the reachability problem over its region automaton, namely the automaton whose states are obtained by coupling control states and regions, and transitions are obtained by suitably coupling the transition relation of the timed automaton and the successor relation defined on regions.

Given a TRSM \( T = \langle A_1, \ldots, A_n, X \rangle \) belonging to TRSM_1, with \( A_i = \langle N_i \cup B_i, Y_i, E_{N_i}, E_{X_i}, \delta_i \rangle \), in order to compute reachability in \( T \) we build a region RSM \( R \) for \( T \), over the alphabet \( \Sigma \), whose components are the region automata obtained from the components of the original TRSM. More formally, \( R = \langle A'_1, \ldots, A'_n \rangle \), where:

\[
A'_i = \langle \{ N_i \times \mathit{Reg} \} \cup B_i, Y_i, E_{N_i} \times \mathit{Reg}, E_{X_i} \times \mathit{Reg}, \delta_i \rangle,
\]

with \( 1 \leq i \leq n \) and \( \mathit{Reg} \) is the finite set of regions for the set of clocks \( X \) and the indexed family \( \{ c_x \}_{x \in X} \), where \( c_x \) is the maximal constant \( c \) which is compared with clock \( x \) in some clock constraint \( \varphi \) of a transition of \( T \). \( \mathit{Reg} \) is computed as the fix point of an iterative process, which builds a chain of region RSMs \( \mathit{Reg}^{10}, \ldots, \mathit{Reg}^{k} \). The idea is that, for the sake of reachability, given a run of a TRSM_1, it is possible to abstract away all the sub-runs bounded by matching pairs of call and return transitions. This abstraction can be performed by augmenting the region RSM \( R \) with summary transitions, which can be used, during the reachability analysis, to skip the invocation of and the return from the component, provided that the exit node of the component is actually reachable from the corresponding entry node. The process described below iteratively computes these summary transitions.

The initial RSM is \( \mathit{Reg}^{0} = \langle A'^{0}_1, \ldots, A'^{0}_n \rangle \), where \( A'^{0}_i = \langle \{ N_i \times \mathit{Reg} \} \cup B_i, Y_i, E_{N_i}, E_{X_i}, \delta'_i \rangle \), with \( 1 \leq i \leq n \). The transitions in \( \delta'_i \) are:

1. \( (u_1, \mathit{reg}1), \mathit{sigma}(u_1, reg_2) \), if \( reg_2 \) is a time successor of \( \mathit{reg}1 \);
2. \( (u_1, \mathit{reg}1), \mathit{sigma}(u_2, reg_2) \), if \( \{ u_1, \sigma, \phi, r_1, \mathit{empty}, u_2 \} \in \delta_1 \), \( \mathit{reg}1 \in \phi \) and \( \mathit{reg}2 = \mathit{reg}1 \downarrow_{r_1} \);
3. \( (u_1, \mathit{reg}1), \mathit{sigma}(b, reg_2) \), if \( \{ u_1, \sigma, \phi, r_1, \emptyset, (b, u_2) \} \in \delta_1 \), \( \mathit{reg}1 \in \phi \) and \( \mathit{reg}2 = \mathit{reg}1 \downarrow_{r_1} \).

Notice that \( \delta'_i \) does not contain return transitions. At the \( (k+1) \)-th iteration \( (k \geq 0) \), we compute \( \delta^{(k+1)}_i \), by adding appropriate summary transitions. Suppose that there is a call transition in the \( i \)-th component from a node \( u_1 \) to a box \( b \) invoking the \( j \)-th component into the entry node \( en \), and there is a return transition from the exit node \( \mathit{ex} \) of the \( j \)-th component to a node \( u_2 \) of the \( i \)-th component. In order to add a summary transition from \( \langle u_1, \mathit{reg}1 \rangle \) to \( \langle u_2, \mathit{reg}2 \rangle \), we need to know whether \( (ex, \mathit{reg}' \rangle \) is locally reachable (i.e. without exploiting call or return transitions) from \( \langle en, \mathit{reg} \rangle \) in the \( j \)-th component of \( R^{(k)} \), where \( \mathit{reg} \) is the region resulting from \( \mathit{reg}_1 \) after resetting the clocks according the call transition, \( \mathit{reg}_2 \) is the region resulting from \( \mathit{reg}_1 \) after resetting the clocks according the return transition in the TRSM and \( \mathit{reg}'' \) satisfy the constraint in the return transition.

Formally, the relation transition \( \delta^{(k+1)}_i \) is \( \delta^{(k)}_i \) augmented with the following transitions. For each entry node \( (b, (en, \mathit{reg})) \), where \( j=Y_1(b) \), and for each exit node \( (ex, \mathit{reg}' \rangle \in A'^{(k)}_j \) reachable from \( \langle en, \mathit{reg} \rangle \) in \( A'^{(k)}_j \), if there is a transition \( \langle (u_1, \mathit{reg}_1), \mathit{sigma}_1, (b, (en, \mathit{reg}' \rangle) \rangle \) in \( \delta^{(k)}_i \), we add to \( \delta^{(k+1)}_i \) the transitions:

\[
\left\{ (u_1, \mathit{reg}_1), \mathit{sigma}_1, (u_2, \mathit{reg}_1 \downarrow_{r_1}) \right\}, \text{whenever } (b, ex), \mathit{sigma}_2, \varphi, r, (X, u_2) \in \delta_1 \text{ and } \mathit{reg}'' \in \varphi.
\]

The iterative construction terminates when it is not possible to add new transitions. Termination of the procedure is ensured since the number of the states of \( \mathit{Reg} \) and the number of the transitions in each \( \delta_i \) is finite.
The region RSM R, resulting from the construction, can be proved equivalent, from the reachability viewpoint, to the original TRMS (see the complete work). Since the reachability problem for RMSs is decidable [1], this establishes decidability of reachability for the class TRSM_1.

Theorem 3: The reachability problem for the class TRSM_1 is decidable.

Proof:
Given a TRSM \( T = \{ A_1, \ldots, A_n, X \} \), the region RSM \( R \) for \( T \), over the alphabet \( \Sigma \), is \( R = \{ A'_1, \ldots, A'_n \} \), where for all \( (1 \leq i \leq n) \), \( A'_i \) is a tuple \( \langle (N_i \times \text{Reg}) \cup B_i, Y_i, E\text{r}_i', E\text{x}_i', \delta_i' \rangle \) and:

- \( \text{Reg} \) is the finite set of all regions with respect to the set of the clocks \( X \) of \( T \) and the indexed family \( \{ c_x \}_{x \in X} \), where \( c_x \) is the maximal values \( c \) such that the clock \( x \) is compared with \( c \) in some clock constraint appearing in a guard \( \varphi \) of all \( \delta_i \);
- \( \text{En}_i' = E\text{n}_i \times \text{Reg} \) is the set of initial states;
- \( \text{Ex}_i' = E\text{x}_i \times \text{Reg} \) is the set of final states.

The desired RSM \( R \) is built as the limit of a (finite) sequence of RSM \( R^{(i)} \) \((i \geq 0)\), where \( R^{(0)} = \{ A_1^{(0)}, \ldots, A_n^{(0)} \} \) and, for all \( (1 \leq i \leq n) \), \( A_i^{(i)} \) is the tuple \( \langle (N_i \times \text{Reg}) \cup B_i, Y_i, E\text{r}_i, E\text{x}_i, \delta_i \rangle \). The transitions in \( \delta_i \) are of the form:

1. \( \langle (u_1, \text{reg}_1), \sigma(u_1, \text{reg}_2) \rangle \) where \( \text{reg}_2 \) is a time successor of \( \text{reg}_1 \);
2. \( \langle (u_1, \text{reg}_1), \sigma(u_2, \text{reg}_2) \rangle \), where \( (u_1, \sigma, \varphi, r_1, \emptyset, u_2) \) \( \in \delta_i \) and \( \text{reg}_2 \) is such that for all clock valuation \( v \) \( \in \text{reg}_1 \) with \( v \in \partial \) and \( v \downarrow r_1 \in \text{reg}_2 \);
3. \( \langle (u_1, \text{reg}_1), \sigma(b, (u_2, \text{reg}_2)) \rangle \), where \( (u_1, \sigma, \varphi, r_1, \emptyset, (b, u_2)) \) \( \in \delta_i \) and \( \text{reg}_2 \) is such that for all clock valuation \( v \in \text{reg}_1 \) with \( v \in \partial \) and \( v \downarrow r_1 \in \text{reg}_2 \).

Given \( R^{(k)} \), with \( k \geq 0 \), \( R^{(k+1)} = \{ A_1^{(k+1)}, \ldots, A_n^{(k+1)} \} \), where for all \( (1 \leq i \leq n) \), \( A_i^{(k+1)} \) is the tuple \( \langle (N_i \times \text{Reg}) \cup B_i, Y_i, E\text{r}_i, E\text{x}_i, \delta_i^{(k+1)} \rangle \). The relation transition \( \delta_i^{(k+1)} \) is \( \delta_i^{(k)} \) augmented with the following transitions. For each enter node \( (b, \text{en}, \text{reg}) \), where \( j = Y_i(b) \), and for each exit node \( (ex, \text{reg}' \rangle \in A_j^{(k)} \), reachable from \( (en, \text{reg}) \) in \( A_j^{(k)} \), if there is a transition \( \langle (u', \text{reg}' \rangle, \sigma, (b, \text{en}, \text{reg}') \rangle \) in \( \delta_i^{(k)} \) we add to \( \delta_i^{(k+1)} \) the transitions:

- \( \langle (u', \text{reg}' \rangle, \sigma, (u, \text{reg}_2) \rangle \) for all the transitions \( (b, \text{ex}) \), \( \sigma_1, \varphi, r_1, X, u \) \( \in \delta_i \), where \( \text{reg}_2 \) is such that for all clock valuation \( v \in \text{reg}' \rangle \), with \( v \in \partial \) and \( v \downarrow r_1 \in \text{reg}_2 \).

The iterative construction terminates when it is not possible to add new transitions, and we obtain the relations \( \delta_i' \) (termination is ensured since the number of the states and the number of the transitions in \( \delta_i \) is finite, for all \( i \)).

For proving the Theorem 3 we preliminary state the following property:
Let \( (b_1, \ldots, b_s, u) \) be an untimed global state of \( T \), the global state \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) is reachable from an initial global state, for some clock evaluations \( v_1, \ldots, v_s, v \), iff the global state \( (b_1, \ldots, b_s, (u, \text{reg})) \) of \( R \) is reachable from an initial global state of \( R \), for some \( \text{reg} \) and \( v \in \text{reg} \).

Lemma 1: For all \( (b_1, \ldots, b_s, b'_1, \ldots, b'_s) \in B \), for all clock evaluations \( v_1, \ldots, v_s, v_1', \ldots, v_s, v'_k \), with \( 0 \leq s \leq k \), for all \( \text{reg}, \text{reg}' \in \text{Reg} \) and for all \( u, u' \in N_i \), such that \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v), (b_1, \ldots, b_s, (u, \text{reg})), (b_1, \ldots, b_s, u, v_1, \ldots, v_k, v) \) and \( (b_1, \ldots, b'_s, (u', \text{reg}')) \) are well-defined global states: if \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v') \), then \( (b_1, \ldots, b'_s, u, v_1, \ldots, v_k, v) \) \( \Rightarrow \) \( ((b_1, \ldots, b'_s, u, v_1, \ldots, v_k, v), (b_1, \ldots, b'_s, (u', \text{reg}')), (b_1, \ldots, b'_s, (u, \text{reg})), \Rightarrow \) \( (b_1, \ldots, b'_s, (u', \text{reg}')) \).

Proof:
We starting proving the Lemma for the TRMS \( T \).

The prove is by induction on the length \( k \) of the run \( \lambda_T = (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) of the run of length \( k+1 \). The \( (k+1) \) – th transition can have one of the following forms:

1. (progress transition): \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \);
2. (reset transition): \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \); and has an effect of the reset transition, we have that \( (b_1, \ldots, b'_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b'_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b'_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b'_s, u, v_1, \ldots, v_s, v) \);
3. (return transition): \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \) \( \Rightarrow \) \( (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v) \);
Let \( \lambda_T = g_{s_0} \delta_0 \ldots \delta_1 \) be a run of the TRSM \( T \). The nesting depth \( \lambda_T \) is defined as the maximum \( s \) such that \( g_{s_j} = (b_j, \ldots, b_j, u, v_j, \ldots, v_j) \) for some \( 0 \leq j \leq i \).

**Lemma 2:** Let \( u, u' \in N_j \) then \( (u, u') \rightarrow^* \langle u', v' \rangle \) iff for all \( v \in \text{reg} \) there is \( v' \in \text{reg}' \) and \( (u, v) \rightarrow^* \langle u', v' \rangle \).

**Definition 3:** Let \( \lambda_T = g_{s_0} \delta_0 \ldots \delta_1 \) be a run of the TRSM \( T \). The nesting depth of \( \lambda_T \) is defined as the maximum \( s \) such that \( g_{s_j} = (b_j, \ldots, b_j, u, v_j, \ldots, v_j) \) for some \( 0 \leq j \leq i \).

**Lemma 2:** Let \( u, u' \in N_j \) then \( (u, u') \rightarrow^* \langle u', v' \rangle \) iff for all \( v \in \text{reg} \) there is \( v' \in \text{reg}' \) and \( (u, v) \rightarrow^* \langle u', v' \rangle \).

**Proof:**

Let \( \lambda_T = (u_1, v_1) \rightarrow^* \langle u_n, v_n \rangle \) be a run of \( T \), with \( u_1, u_n \in N_j, v_1 \in \text{reg}_1 \) and \( v_n \in \text{reg}_n \). The proof is by induction on nesting depth of \( \lambda_T \).

**Base case, \( k = 0 \):** In this case, every transition \( (u_1, v_1) \rightarrow^* \langle u_{i+1}, v_{i+1} \rangle \) in \( \lambda_T \), with \( 1 \leq i \leq n \) and \( u_i, u_{i+1} \in N_j \), can have two kinds of sub-runs. The first one of the form \( (u_i, v_i) \rightarrow^* \langle u_i, v_i' \rangle \), with nesting depth \( s < k \) and \( i < i' \leq n \). All sub-runs of this form have, by inductive hypothesis, the corresponding sub-runs \( (u_i, v_i) \rightarrow^{*A'_j} (u_i, v_i') \), with \( v_i \in \text{reg}_i \) and \( v_i' \in \text{reg}_i' \). The second one of the form \( (u_i, v_i) \rightarrow^* \langle u_i', v_i' \rangle \) with nesting depth \( k+1 \) and \( i < i' \leq n \). Any such run can be split in the following way:

\( (u_i, v_i) \rightarrow^* \langle b, en, v_i, v_i' \rangle \rightarrow^* (b, ex, v_i, v_i') \rightarrow^* \langle u_i', v_i' \rangle \).

Therefore:

- since \( (u_i, v_i) \rightarrow^* \langle b, en, v_i, v_i' \rangle \) is in \( L(T) \), then the call transition \( (u_i, \sigma_1, r_1, 0, (b, en)) \) is in \( \delta_j \), with \( Y_j(b) = b, en \in E_n, v_i \in \text{reg}_i, v_i' = v_i \downarrow_1 \).
- By construction of \( \delta_j \), we have that \( \langle u_i, v_i \rangle \rightarrow^* \langle u_i, v_i' \rangle \), with \( v_i \in \text{reg}_i \) and \( v_i' \in \text{reg}'_i \).
- since \( (b, ex, v_i, v_i') \) is in \( L(T) \), then the return transition \( (b, ex, v_i, v_i') \rightarrow^* \langle \sigma_2, \varphi_2, r_2, X, v_i' \rangle \) is in \( \delta_j \), with \( v_i'' \in \text{reg}_i \) and \( v_i = v_i \downarrow_2 \) and \( v_i'' \in \text{reg}_i' \).
- from \( (b, en, v_i, v_i') \rightarrow^* \langle b, ex, v_i, v_i'' \rangle \). By inductive hypothesis, we have \( \langle (en, v_i') \rightarrow^{A'_j} \langle ex, v_i'' \rangle \rangle \in \text{reg}'_{i''} \).

Since \( ex \) is reachable from \( en \) (locally in the component machine \( A'_j \)) and there are a call and a return transitions as above (in \( T \)), by construction of \( R \) there is a summary transition \( \langle (u_i, v_i), \sigma_1, (u_i', v_i') \rangle \in \delta_j' \) and we can write \( \langle (u_i, v_i) \rightarrow^* A'_j (u_i', v_i') \rangle \).

Since \( ex \) is reachable from \( en \) (locally in the component machine \( A'_j \)) and there are a call and a return transitions as above (in \( T \)), by construction of \( R \) there is a summary transition \( \langle (u_i, v_i), \sigma_1, (u_i', v_i') \rangle \in \delta_j' \) and we can write \( \langle (u_i, v_i) \rightarrow^* A'_j (u_i', v_i') \rangle \).

The proof is by induction on the index \( k \) of the transition relations \( \delta_j^{(k)} \).

**Base case, \( k = 0 \):** Immediate, by construction, since \( \delta_j^{(0)} \in \delta_j' \).

**Inductive step, \( k > 0 \):** Let us suppose, by inductive hypothesis, that the property holds for the run that have all the transitions in \( \delta_j^{(k)} \). Let \( \lambda_{A'_j}=\langle (u_1, v_1) \rangle \rightarrow^* A'_j \langle (u_n, v_n) \rangle \) be a run where at least one transition belongs to \( \delta_j^{(k+1)} \setminus \delta_j^{(k)} \). For all the transitions of the form \( \langle (u_i, v_i), \sigma, (u_{i+1}, v_{i+1}) \rangle \in \delta_j^{(k)} \), by inductive hypothesis we can write \( \langle (u_i, v_i) \rightarrow^* \langle u_{i+1}, v_{i+1} \rangle \rangle \), with \( v_i \in \text{reg}_i \) and \( v_i \downarrow_{i+1} \in \text{reg}_{i+1} \).

For all the transitions of the form \( \langle (u_i, v_i), \sigma, (u_{i+1}, v_{i+1}) \rangle \in \delta_j^{(k+1)} \setminus \delta_j^{(k)} \), by construction of \( \delta_j^{(k+1)} \) we have that:

- there is in \( T \) a call transition \( (u_i, \sigma_1, r_1, 0, (b, en)) \) in \( \delta_j \), with \( Y_j(b) = b, en \in E_n, v_i \in \text{reg}_i, \varphi_i \). Taking a clock valuation \( v_i \in \text{reg}_i \), we have \( \langle (u_i, v_i) \rightarrow^* \langle b, en, v_i, v_i' \rangle \rangle \rightarrow^* \langle b, ex, v_i, v_i'' \rangle \rangle \) with \( v_i \in \text{reg}_i \) and \( v_i' \in \text{reg}'_i \).
- there is a sub-run \( \langle (en, v_i) \rightarrow^* \langle ex, v_i' \rangle \rangle \) and then there is the sub-run \( \langle (en, v_i) \rightarrow^* \langle ex, v_i'' \rangle \rangle \) by inductive hypothesis, we can write \( \langle (en, v_i) \rightarrow^* \langle ex, v_i' \rangle \rangle \), with \( v_i' \in \text{reg}_i \) and \( v_i'' \in \text{reg}'_i \).
• there is in \( T \) the matching-return transition \((b, ex), \sigma_2, \varphi_2, r_2, X, u_{i+1} \in \delta_i\), with \(ex \in Ex_j \) and \(reg' \in \varphi_2\) (then \(v' \in \varphi_2\)). Then, we can write \((b, ex, v_i, v') \rightarrow_T (u_{i+1}, v_i \downarrow r_2)\).

Combining the facts above, we have that \((u_1, v_1) \rightarrow_T (u_n, v_n)\).

Now, we show that, for an untimed global state \((b_1, \ldots, b_s, u)\) of \( T \), the global state \((b_1, \ldots, b_s, u, v_1, \ldots, v_s, v)\) is reachable from an initial global state, for some clock evaluations \(v_1, \ldots, v_s, v\), iff the global state \((b_1, \ldots, b_s, (u, reg))\) of \( R \) is reachable from an initial global state of \( \mathcal{R} \), for some region \( reg \) with \( v \in reg\).

**Proof:**

\(\Rightarrow\).

By induction on the length \( k \) of the run.

(Base case, \( k = 0 \)). Immediate.

(Inductive step, \( k > 0 \)). Let \( \lambda_T = (en, v_0) \rightarrow_T (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_s+1)\), with \( u \in N_j \), be a run of length \( k + 1 \). The \((k+1) - th\) transition can have one of the following forms:

- (progress transition): \((b_1, \ldots, b_s, u, v_1, \ldots, v_s, v') \rightarrow_T (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_s+1)\), with \( v_{s+1} = v' + t \) for some \( t \in D^{\geq 0} \). By construction of \( R \), there is \((u, reg), \sigma, (u, reg_{s+1}) \in \delta_i\), with \(v' \in reg\) and \(v_{s+1} \in reg_{s+1}\) (\(reg_{s+1}\) is a time successor of \(reg\)). By inductive hypothesis, we can write \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg))\). By the effect of the transition above, we have \((b_1, \ldots, b_s, (u, reg)) \rightarrow_R (b_1, \ldots, b_s, (u, reg_{s+1}))\). Therefore, we can conclude \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg_{s+1}))\).

- (reset transition): \((b_1, \ldots, b_s, u', v_1, \ldots, v_s, v') \rightarrow_T (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_s+1)\), with \(u', \sigma, \varphi, r, \emptyset, u \in \delta_i\), \(v' \in \varphi\), \(v_{s+1} = v' \downarrow r\), and \(u' \in N_i\). By construction of \( R \), there is the transition \((u', reg), \sigma, (u, reg_{s+1}) \in \delta_i\), with \(v' \in reg\) and \(v_{s+1} \in reg_{s+1}\). By inductive hypothesis, we can write \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u', reg))\). By the effect of the transition above, we have \((b_1, \ldots, b_s, (u', reg)) \rightarrow_R (b_1, \ldots, b_s, (u, reg_{s+1}))\). Therefore, we can conclude \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg_{s+1}))\).

- (call transition): \((b_1, \ldots, b_{s-1}, u', v_1, \ldots, v_{s-1}, v_s) \rightarrow_T (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_s+1)\), with \(u', \sigma, \varphi, r, \emptyset, (b_s) \in \delta_i\), \(V_i(b_s) = j\), \(v_s \in \varphi\), \(v_{s+1} = v_s \uparrow r\), and \(u' \in N_i\). By construction, there is in \( R \) a call transition \((u', reg), \sigma, (b_s, (u, reg_{s+1})) \), with \(v_s \in reg\) and \(v_{s+1} \in reg_{s+1}\) (\(reg_{s+1}\) is a time successor of \(reg\)). By inductive hypothesis, we can write \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u', reg))\). By the effect of the transition above, we have \((b_1, \ldots, b_s, (u', reg)) \rightarrow_R (b_1, \ldots, b_s, (u, reg_{s+1}))\). Therefore, we can conclude \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg_{s+1}))\).

- (return transition): if the \((k+1) - th\) transition is a return transition, then we can split the run \(\lambda_T\) in the following way

1) a sub-run of length \(k' \leq k\) in the form \((en, v_0) \rightarrow^*_T (b_1, \ldots, b_s, u', v_1, \ldots, v_s, v')\), with \(u' \in N_j\). By inductive hypothesis, we have \((en, reg_0) \rightarrow^*_R (b_1, \ldots, b_s, (u', reg'))\), with \(v_s \in reg_0\) and \(v' \in reg'\).

2) \((b_1, \ldots, b_s, u, v_1, \ldots, v_s, v') \rightarrow_T (b_1, \ldots, b_{s+1}, v', v_1, \ldots, v_s, v', v''), \) where \(v', \sigma_2, r_2, \emptyset, (b_{s+1}, v'') \in \delta_j\) is a call transition, \(Y_j(b_{s+1}) = i\), \(v' \in \varphi_1\) and \(v'' = v' \downarrow r_1\). By construction of \( R \), there is the transition \((u', reg'), \sigma_2, (b_{s+1}, (en, reg'')) \), with \(v' \in reg'\), \(v'' \in reg''\) (\(reg'' = reg'_r\)) and (since for a clock constraint \(\varphi\) belongs to a transition in \( T \), if \( v \equiv v'\), then \( v \in \varphi \)) \(v'' \in \varphi_1\).

3) \((b_1, \ldots, b_{s+1}, v', v_1, \ldots, v_s, v', v'' \rightarrow_T (b_1, \ldots, b_{s+1}, v', v_1, \ldots, v_s, v', v''), \) with \(v' \in \varphi'\), \(v'' \equiv v''\), the exit node \((v', \varphi''\) is reachable from the enter node \((en, \varphi''\) in the component \(A_i\).

4) \((b_1, \ldots, b_{s+1}, v', v_1, \ldots, v_s, v', v'' \rightarrow_T (b_1, \ldots, b_{s+1}, v', v_1, \ldots, v_s, v', v'')\), where \((b_{s+1}, v'\), \(v_2, \varphi_2, r_2, X, u \in \delta_j\) is a return transition, \(v'' \equiv v''\), \(v' \in \varphi_2 \) (then \(reg'' \equiv v''\)), \(v_{s+1} = v' \downarrow r_2\), and \(v_{s+1} \in reg_{s+1}\).

By construction of \( R \), as a consequence of the three points above, there exists a summary transition \((u', reg'), \sigma, (u, reg_{s+1}) \in \delta_i\). Combining this transition with the inductive hypothesis as above, we can conclude \((en, v_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg_{s+1}))\) \(\Rightarrow\).

The prove is by induction on the length \( k \) of the run.

(Base case, \( k = 0 \)). Immediate.

(Inductive step, \( k \geq 0 \)). Let \( \lambda_R = (en, reg_0) \rightarrow^*_R (b_1, \ldots, b_s, (u, reg_{s+1})) \) be a run of length \( k + 1 \), with \( u \in N_i\). The \((k+1) - th\) transition can have one of the following forms:

- \((b_1, \ldots, b_s, (u, reg)) \rightarrow_R (b_1, \ldots, b_s, (u, reg_{s+1}))\), where \((u, reg), \sigma, (u, reg) \in \delta_i\) and \(reg_{s+1}\) is a time successor of \(reg\). By construction of \( R \), there exists in \( L(T) \) (LTS of \( T \)) a progress transition \((b_1, \ldots, b_s, u, v_1, \ldots, v_s, v') \rightarrow_T (b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1})\), with \(v_{s+1} = v' + t, t \in D^{\geq 0}\). By construction of \( R \), there exists in \( L(T) \) a reset transition \((b_1, \ldots, b_s, v_1, \ldots, v_s, v_{s+1})\). Combining the facts above with the inductive hypothesis, we can conclude \((en, v_0) \rightarrow^*_T (b_1, \ldots, b_s, (u, reg_{s+1}))\).
of TRSM_1 is a proper subclass of TRSM (resp. EPTA_2).

We conclude the section by considering the complexity of the reachability problem, by showing that the problem is PSPACE-complete for the subclass TRSM_0. Notice that, as a consequence of Theorem 1, that establish an effective equivalence of TRSM and EPTA_2, and of Theorems 2 and 3, we can state the following expressiveness results.

**Corollary 2:** TRSM_1 (resp. EPTA_1) is a proper subclass of TRSM (resp. EPTA_2).

The idea underlying the construction for reachability in a TRSM_0 T, is similar to the construction given above for TRSM_1. In this case, however, we do not need to build the region graph for T, as we can add summary transitions directly to T. Indeed, any call transition in a TRSM_0 resets all the clocks when entering the invoked component. Hence, the only relevant local reachability problem from an enter to an exit node in the invoked component is the one which assumes the clock valuation set to zero at the entry node. This is the main difference with respect to TRSM_1, where different clock valuations for the entry node have to be considered, thus forcing to explicitly take into account regions in the definition of a summary transition.

Since the number of summary transitions for TRSM_0 is clearly polynomial in the number of transitions of T, and local reachability in a component boils down to reachability in a Timed Automaton, which is known to be a PSPACE-complete problem [2], we can construct, using only polynomial space, a TRSM_0 T’ equivalent to T from the reachability viewpoint (see the complete work). This is done, again, using a fix point construction, which builds a chain of TRSM_0 T^(0), . . . , T^(m), with T^(0) = T, and the transition relation δ^k_j (k ≥ 0) is δ^k_j augmented with summary transitions as follows.

For every box b ∈ B_j, with ∀j(b) = j, for every entry node en ∈ En_j and for every return transition t_ex = (b, ex, σ_j, ϕ_j, r_j, X_j, u_j) ∈ δ_j, we first build the component machine A^k_j ↓_t_ex = (N_j ∪ B_j ∪ {u_j}, Y_j, En_j, {u_j}, δ^k_j), where

1. δ^k_j = δ^k_j ∪ \{ex, σ_j, ϕ_j, 0, 0, u_j’\}, and u’ ∉ N_j.

Notice that, for any clock valuation v, (en, v_0) → A^k_j ↓_t_ex (u’, v) if and only if (en, v_0) → A^k_j ↓_t_ex (ex, v) and v’ ∈ ϕ_j. Therefore, local reachability of u’ in A^k_j ↓_t_ex ensures both that ex is locally reachable in A^k_j and that t_ex is enabled at the exit node, since the constraint ϕ_j, occurring in t_ex, is satisfied by the reached clock valuation.

Then, if u’ is locally reachable from en in A^k_j ↓_t_ex, then for all call transition (u_1, σ_1, ϕ_1, X_1, 0, (b, en)) ∈ δ_j, the summary transition (u_1, σ_1, ϕ_1, r_j, 0, u_2) is added to δ^k_j (k + 1).

Once computed T’, the reachability problem for an unlimited global state (b_1, . . . , b_s, u) in T can be solved by checking reachability of (b_1, . . . , b_s, u) in T’, according to the following procedure.

Let A_0, A_1, . . . , A_s be the sequence of the invoked components, with i_0 = 1 and Y_j, (b_j) = i_j. We start by guessing a sequence of entry nodes en_0, . . . , en_s, with en_j ∈ En_j. For each j ≥ 0, we then check whether (b_j+1, en_j+1) is locally reachable from en_j in the component A_j, where, for the sake of local reachability, call transitions are treated as reset transitions. Finally, we check whether u is locally reachable from en_s in A_s.
reachable from \(en_i\) in the component \(A_i\). If all the checks are fulfilled, then reachability is ensured. Notice that each local reachability check is a reachability analysis in a Timed Automaton, which can be done using polynomial space. Hence, we can conclude the following theorem:

**Theorem 4:** The reachability problem for TRSM\(_0\) is PSPACE-complete.

**Proof:**

Let us consider a TRSM\(_0\) \(T = \langle A_1, \ldots, A_n, X \rangle\) where, for all \(1 \leq i \leq n\), \(A_i = \langle N_i \cup B_i, Y_i, E_{ni}, Ex_i, \delta_i \rangle\). We define an algorithm which enriched the transition relation by adding summary transition. We that prove that, if \(T’\) is the output of the below reported algorithm, then \(T’\) and \(T\) are equivalent for reachability viewpoint. The algorithm for augmented \(T\) with summary transitions is the following:

**REPEAT**

summarized = FALSE

FOR ALL \(A_i\)

FOR ALL \(b \in B_i\)

\(j = Y_i(b)\)

FOR ALL \(en \in E_{nj}\)

FOR ALL \(t_{ex} = \langle (b, ex), \sigma_2, \varphi_2, r_2, X, u_2 \rangle \in \delta_i\)

\(\delta'_j = \delta_j \cup \{(ex, \sigma_2, \varphi_2, \emptyset, \emptyset, u_2)\}\)

\(A_j \downarrow_{t_{ex}, en, u'}\)

\((u' \notin N_j)\)

IF \(treach(A_j \downarrow_{t_{ex}, en, u'})\)

summarized = TRUE

FOR ALL \(t_{en} = \langle u_1, \sigma_1, \varphi_1, X, \emptyset, (b, en) \rangle \in \delta_i\)

\(\delta_i = \delta_i \cup \{(u_1, \sigma_1, \varphi_1, r_2, \emptyset, (b, \emptyset, u_2)\}\}

UNTIL \(\neg\) summarized

In the algorithm, the function \(treach\) is the standard procedure for the reachability problem for a Timed Automaton: for instance, if exists clock valuation \(v\) such that the configuration \(\langle u', v \rangle\) is reachable from the initial configuration \(en, v_0\) in the Timed Automaton \(A_j \downarrow_{t_{ex}}\), then the output of \(treach(A_j \downarrow_{t_{ex}, en, u'})\) is TRUE; else the output is FALSE.

Let \(T\) a TRSM\(_0\) and \(T'\) the TRSM\(_0\) output of the algorithm. In order to prove that \(T'\) is equivalent to \(T\) for reachability, we need to show the following lemma.

**Lemma 3:** \(\langle u, v \rangle \rightarrow^{*}_T \langle u', v' \rangle\), with \(u, u' \notin N_j\), iff \(\langle u, v \rangle \rightarrow^{*}_{A_j} \langle u', v' \rangle\).

**Proof:**

\((\Rightarrow)\).

The proof is by induction on nesting depth \(k\) of the run \(\lambda_T = \langle u_1, v_1 \rangle \rightarrow^{*}_T \langle u_n, v_n \rangle\).

(Base case, \(k = 0\)). In this case, every transition \(\langle u_i, v_i \rangle \rightarrow^{*}_T \langle u_{i+1}, v_{i+1} \rangle\) in \(\lambda_T\), with \(1 \leq i \leq n\), can be only either a progress or a reset transition:

- (progress transition): we have that \(u_{i+1} = u_i \in N_j, v_{i+1} = v_i + t\), with \(t \in D^{\geq 0}\). By construction of \(A_j\), there is a progress transition \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_i, v_{i+1} \rangle\);

- (reset transition): we have that \(v_{i+1} = v_i \downarrow r_i\), with \(\langle u_i, \sigma_1, \varphi_1, r_1, \emptyset, u_{i+1} \rangle \in \delta_j\) and \(v_i \in \varphi_1\). By construction of \(A_j\), there is a reset transition \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_{i+1}, v_{i+1} \rangle\).

(Inductive step, \(k > 0\)). Let us suppose that the nesting depth of the run \(\langle u_1, v_1 \rangle \rightarrow^{*}_T \langle u_n, v_n \rangle\) is \(k+1\). The run can have two kinds of sub-runs. The first one has the form \(\langle u_i, v_i \rangle \rightarrow^{*}_T \langle u_j, v_j \rangle\), with nesting depth \(s \leq k\) and \(1 \leq i < j \leq n\). All sub-runs of this form have, by inductive hypothesis, the corresponding sub-runs \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_j, v_j \rangle\). The last one has the form \(\langle u_i, v_i \rangle \rightarrow^{*}_T \langle u_i, v_i \rangle\) with nesting depth \(k+1\) and \(1 \leq i < j \leq n\). It is possible to split all sub-runs of this form in such a way that:

- a call transition \(\langle u_i, v_i \rangle \rightarrow_T \langle b, ex, v_i, v_i \downarrow r_i \rangle\), with \(\langle u_i, \sigma_1, \varphi_1, r_1, \emptyset, (b, en) \rangle \in \delta_j\), \(Y_j(b) = i\), \(en \in E_{ni}\) and \(v_i \in \varphi_1\). By construction of \(\delta_j\), \(\langle u_i, \sigma_1, \varphi_1, r_1, \emptyset, (b, \emptyset, v_i) \rangle \in \delta'_j\);

- a sub-run having the form \(\langle b, ex, v_i, v_i \downarrow r_i \rangle \rightarrow_T \langle b, ex, v'' \rangle\), with \(ex \in E_{xi}\). By Lemma 1, we can write \(\langle b, ex, v_i, v_i \downarrow r_i \rangle \rightarrow_T \langle b, ex, v'' \rangle\), with nesting depth \(s = k\). By inductive hypothesis there is a corresponding sub-run \(\langle b, ex, v_i, v_i \downarrow r_i \rangle \rightarrow^{*}_{A_j} \langle b, ex, v'' \rangle\), with all transitions in \(\delta_i^{(k)}\);

- a return transition \(t_{ex} = \langle b, ex, v_i, v'' \rangle \rightarrow_T \langle u_i', v_i' \rangle\), with \(\langle b, ex,\sigma_2, \varphi_2, r_2, X, u_{i+1} \rangle \in \delta_j\), \(v_i' = v_i \downarrow r_2\) and \(v'' \notin \varphi_2\).

As a consequence of the facts above, by construction of \(A_j^{(k)} \downarrow_{t_{ex}} = \langle N_j \cup B_j \cup \{u'\}, Y_j, E_{nj}, \{u'\}, \delta'_j \rangle\), since \(ex\) is reachable from \(en\) and \(v'' \notin \varphi_2\), we have that \(u'\) is reachable from \(en\) (since the valuation \(v''\) in \(ex\) satisfies the constraint \(\varphi\) of the return transition \(t_{ex}\)) and then \(treach(A_j \downarrow_{t_{ex}, en, u'}) = TRUE\). By construction of \(T'\) there is a summary transition \(\langle u_i, \sigma_1, \varphi_1, u_i, v_i \rangle \in \delta_i'\) and then we can write \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_i', v_i' \rangle\). Combining the facts above, we have that \(\langle u_1, v_1 \rangle \rightarrow T' \langle u_n, v_n \rangle\).

\((\Leftarrow)\).

The proof is by induction on the index \(k\) of \(\delta_i^{(k)}\).

(Base case, \(k = 0\)). If \(k = 0\), there is no summary transition in \(\delta_i^{(0)}\), and the property holds immediately.

(Inductive step, \(k > 0\)). Let \(\langle u_1, v_1 \rangle \rightarrow^{*}_{A_j} \langle u_s, v_s \rangle\) be a run with at least one transition belonging to \(\delta^{(k+1)}_i \backslash \delta^{(k)}_i\). The run \(\lambda_{A_j}\) can be split as follows:

- for each sub-run \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_i', v_i' \rangle\), with \(1 \leq i < i' \leq s\) and all transitions in \(\delta^{(k)}_i\), by inductive hypothesis we have that \(\langle u_i, v_i \rangle \rightarrow_T \langle u_i', v_i' \rangle\);

- for each summary transition \(\langle u_i, \sigma_1, \varphi_1, r, \emptyset, u_{i+1} \rangle \in \delta^{(k+1)}_i\), such that \(\langle u_i, v_i \rangle \rightarrow^{*}_{A_j} \langle u_{i+1}, v_{i+1} \rangle\), with \(1 \leq i < s\), we have that:

  - there is a call transition \(\langle u_i, \sigma_1, \varphi_1, X, \emptyset, (b, en) \rangle\) in \(\delta'_j\), with \(Y_j(b) = j'\), such that \(v_i \in \varphi_1\);
– there is a return transition \( t_{ex} = (\langle b, \text{ex}, \sigma_2, \varphi_2, r, X, u_{i+1} \rangle) \subseteq \delta \); 
– \( \text{treach}(A_j' \downarrow_{ex}, en, u') \) is TRUE, namely \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} (ex, v) \xrightarrow{\sigma_j, r, \varphi_j} \langle u', v' \rangle \), for some clock valuation \( v \) such that \( v \in \varphi_j \). By construction of \( A_j' \downarrow_{ex} \) and by induction, we have that \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle u_{i+1}, v_i \rangle \).

As a consequence of the facts above, we have that \( \langle u, v \rangle \xrightarrow{\sigma_j, r, \varphi_j} (u', v') \).

Now we can show that, for an untimed global state \( \langle b_1, \ldots, b_s, u, v_1, v_2, \ldots, v_s \rangle \) of \( T \), the global state \( \langle b_1, \ldots, b_s, u, v_1, v_2, \ldots, v_s, v_{s+1} \rangle \) is reachable in \( T \), from an initial global state, for some clock evaluations \( v_1, v_2, \ldots, v_s, v \), iff the global state \( \langle b_1, \ldots, b_s, u \rangle \) of \( T' \) is reachable from the initial global state of \( T' \).

\( \implies \).

**Proof:**

The proof is by induction on the length of the run \( \lambda_T = \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, v_2, \ldots, v_s, v_{s+1} \rangle \), with \( u \in N_j \).

**Base case, \( k = 0 \):** Immediate.

**Inductive step, \( k \geq 0 \):** Let \( \lambda_T \) be a run of length \( k \) + 1. The \((k+1) - \text{th} \) transition can have one of the following forms:

- **(progress transition):** \( \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v' \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \), with \( v_{s+1} = v' + t \), \( t \in D^{\geq 0} \). By construction we have that \( \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v' \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \). Combining the previous fact with the inductive hypothesis, we have \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \).

- **(reset transition):** \( \langle b_1, \ldots, b_s, u', v_1, \ldots, v_s, v' \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \), with \( \langle u', \sigma, \varphi, r, \emptyset, u \rangle \in \delta_j, u' \in N_i, v' \in \varphi \) and \( v_{s+1} = v' \downarrow_{r} \). By construction of \( T' \), \( \langle u', \sigma, \varphi, r, \emptyset, u \rangle \in \delta_j \). Combining this transition with the inductive hypothesis, we have \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \).

- **(call transition):** \( \langle b_1, \ldots, b_s, u', v_1, \ldots, v_s, v_s, v_{s+1} \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \), with \( \langle u', \sigma, \varphi, X, \emptyset, (b, u) \rangle \in \delta_j, Y_{\delta_j}(b_s) = i, u' \in N_i, v' \in \varphi \) and \( v_{s+1} = v' \downarrow_{r} \). By construction of \( T' \), \( \langle u', \sigma, \varphi, X, \emptyset, (b, u) \rangle \in \delta_j \). Combining this transition with the inductive hypothesis, we have \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, v_{s+1} \rangle \).

- **(return transition):** If the \((k+1) - \text{th} \) transition is a return transition \( (t_{ex}) \), then we can split the run \( \lambda_T \) in the following way:
  1. a sub-run of length \( k' \leq k \) in the form \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u', v_1, \ldots, v_s, v' \rangle \), with \( u' \in N_i \). By inductive hypothesis we can write \( \langle en, v_0 \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, u', v_1, \ldots, v_s, v' \rangle \);
  2. a matching call transition \( \langle b_1, \ldots, b_s, u', v_1, \ldots, v_s, v' \rangle \xrightarrow{\sigma_j, r, \varphi_j} \langle b_1, \ldots, b_s, b_{s+1}, en', v_1, \ldots, v_s, v' \rangle \), where \( \langle u', \sigma_1, \varphi_1, X, \emptyset, (b, u, en') \rangle \in \delta_j, Y_{\delta_j}(b_{s+1}) = j, en' \in En_j, v' \in \varphi_1 \) and \( v'' = v' \downarrow_{r} \).
then by construction of $T'$, we have that:

1) there is a call transition $\langle u', \sigma_1, \varphi_1, X, \emptyset, (b_{s+1}, e_{n'}) \rangle \in \delta'_i$, with $Y_i(b_{s+1}) = j$ and $e_{n'} \in \mathcal{E}_{n,j}$. Since $v' \in \varphi_2$ (see above), we have that $(b_1, \ldots, b_s, u', v_1, \ldots, v_s, v') \rightarrow_T \langle b_1, \ldots, b_{s+1}, e_{n'}, v_1, \ldots, v_s, v', v_0 \rangle$;

2) $\text{reach}(A_j \downarrow_{t_{ex'}}) = \text{TRUE}$, where $A_j \downarrow_{t_{ex'}} = \langle N_j \cup B_j \cup \{u''\}, Y_j, \mathcal{E}_{n,j}, \{u''\}, \delta'_j \rangle$ and $t_{ex'} = \langle (b_{s+1}, ex'), \sigma_2, \varphi_2, r_2, \emptyset, u \rangle \in \delta_2$. But $\text{reach}(A_j \downarrow_{t_{ex'}}) = \text{TRUE}$ implies that $ex'$ is reachable from the configuration $\langle e_{n'}, v_0 \rangle$ with some clock valuation $v$, with $v \in \varphi_2$. Then, we can write $\langle e_{n'}, v_0 \rangle \rightarrow^{*}_{\delta_1} \langle ex', v \rangle$. By Lemma 3, we can write $\langle e_{n'}, v_0 \rangle \rightarrow_{\delta} \langle ex', v \rangle$. By Lemma 1, we have that $(b_1, \ldots, b_{s+1}, e_{n'}, v_1, \ldots, v_s, v', v_0) \rightarrow_{\delta} \langle b_1, \ldots, b_{s+1}, e_{n'}, v_1, \ldots, v_s, v', v \rangle$;

3) since $v \in \varphi_2$, we can write $(b_1, \ldots, b_{s+1}, ex', v_1, \ldots, v_s, v', v) \rightarrow_T \langle b_1, \ldots, b_s, u, v_1, \ldots, v_s, \sigma+1, \delta_{ex'} \rangle$, by the effect of the return transition $t_{ex'}$, with $v_{s+1} = v' \downarrow r_2$.

As a consequence, by applying the inductive hypothesis, we can write $\langle e_{n}, v_0 \rangle \rightarrow_{\delta} \langle b_1, \ldots, b_s, v_1, \ldots, v_s, v_{s+1} \rangle$.

Once computed $T'$, the reachability problem for an untimed global state $\langle b_1, \ldots, b_s, u \rangle$ in $T$ can be solved by checking reachability of $\langle b_1, \ldots, b_s, u \rangle$ in $T'$, according to the following procedure.

Let $A_{i_0}, A_{i_1}, \ldots, A_{i_s}$ be the sequence of the invoked components, with $i_0 = 1$ and $Y_{i_{j-1}}(b_j) = i_j$. We start by guessing a sequence of enter nodes $en_0, \ldots, en_s$, with $en_j \in \mathcal{E}_{n,j}$. For each $j \geq 0$, we then check whether $(b_{j+1}, en_{j+1})$ is locally reachable from $en_j$ in the component $A_{i_j}$, where, for the sake of local reachability, call transitions are treated as reset transitions. Finally, we check whether $u$ is locally reachable from $en_0$ in the component $A_{i_0}$. If all the checks are fulfilled, then reachability is ensured. Notice that each local reachability check is a reachability analysis in a Timed Automaton, which can be done using polynomial space.

VI. CONCLUSION

In this paper we have introduced TRSMs, a real time extension of RSMs, able to model real time recursive systems. We have shown that TRSMs allow to specify interesting context-free properties, both on the untimed and the timed dimensions. Though reachability of the general framework is undecidable, we have shown that the problem is still decidable for the meaningful classes TRSM_1 and TRSM_0, and that it is PSPACE-complete for the class TRSM_0, the same complexity of reachability in standard Timed Automata.

A number of interesting issues are still to be investigated. In particular, tight complexity results for the reachability problem in TRSM_1 are yet to be established.

The paper focuses on the relationship with the most related formalisms, namely PTAs and RSMs. On the other hand, a comparison with other related formalisms need to be settled. In particular, the undecidability proof for EPTAs shows that they are able to simulate clock updates (increment and decrement) similar to those provided by Updatable Timed Automata [3]. Also the formalisms of Stopwatch Automata [8] requires a comparison. It seems that Stopwatch Automata can be simulated by the class EPTA_2, while the opposite seems not to hold. Another relevant issue concerns the study of the class of context-free timed languages accepted by EPTAs and by their subclasses.

REFERENCES


