Theoretical Computer Science 411 (2010) 4291–4322

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Weak bisimulation for Probabilistic Timed Automata☆

Ruggero Lanotte [a], Andrea Maggiolo-Schettini [b], Angelo Troina [c,*]

[a] Dipartimento di Informatica e Comunicazione, Università dell'Insubria, Via Carloni 78, 22100 Como, Italy

[b] Dipartimento di Informatica - Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy

[c] Dipartimento di Informatica - Università di Torino, Corso Svizzera 185, 10149 Torino, Italy

ARTICLE INFO

Article history:
Received 26 March 2007
Received in revised form 15 June 2010
Accepted 1 September 2010
Communicated by V. Sassone

Keywords:
Probabilistic timed automata
Weak bisimulation

ABSTRACT

We are interested in describing timed systems that exhibit probabilistic behaviour. To this purpose, we consider a model of Probabilistic Timed Automata and introduce a concept of weak bisimulation for these automata, together with an algorithm to decide it. The weak bisimulation relation is shown to be preserved when either time, or probability is abstracted away. As an application, we use weak bisimulation for Probabilistic Timed Automata to model and analyze a timing attack on the dining cryptographers protocol.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The application of formal methods to the development of validated systems mainly concentrates on specification languages, with semantics adequate to support implementation and verification, and methods and tools for verifying specifications with respect to properties expressed in suitable formalisms.

Models based on the concept of states and transitions, also called automata, have turned out to be particularly intuitive. States of the automaton represent snapshots of the described system, while transitions represent state changes.

Basic formalisms allow the description of the nondeterministic approximation (sometimes called possibilistic) behaviour of a system. This is enough to model the functionality of systems, and hence to capture the qualitative behaviour, but if one wants to capture also quantitative aspects, such as time- or frequency-dependent properties, formalisms must be extended with real-time and probabilistic features. Hence, in systems that model quantitative processes, steps are associated with a given quantity, such as the probability that the step will happen or the resources (e.g. time or cost) needed to perform that step.

Timed Automata have been introduced by Alur and Dill [3] as an extension of ω-Automata to describe real-time systems. Timed Automata are equipped with variables measuring time, called *clocks*. Transitions are guarded by *clock constraints*, which compare the value of a clock with some constant, and by *reset updates*, which reset a clock to the initial value 0. The range of application of analysis through Timed Automata is particularly wide [4,6,12,14,40,41]. Extensions with probability have been proposed (e.g. in [11,35,36]) and may be used to describe real-time systems exhibiting a probabilistic behaviour.

In general, behavioural equivalences may be used to verify a property of a system by assessing the equivalence of the considered system with a system one knows to enjoy the property. Just as an example, the problem of formalising the notion of confidentiality could be boiled down to that of verifying equivalence of processes (see [24,52], where equivalence

☆ This work has been partially carried out during the third author's postdoc at Laboratoire d'Informatique (École Polytechnique) and Laboratoire Spécification et Vérification (École Normale Supérieure de Cachan) supported by the INRIA/ARC project ProNoBiS: Probability and Nondeterminism, Bisimulations and Security.

* Corresponding author. Tel.: +39 011 6706851; fax: +39 011 751603.
E-mail addresses: ruggero.lanotte@uninsubria.it (R. Lanotte), maggiolo@di.unipi.it (A. Maggiolo-Schettini), troina@di.unito.it (A. Troina).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.09.003

relations are used to test whether two systems cannot be distinguished by an intruder). This is a central and difficult question at the heart of computer science to which there is no unique answer. Which notion of equivalence is appropriate depends on the context and application. For example, one should not be surprised that the information security community has failed to come up with a consensus on what constitutes confidentiality. In [24], however, bisimulation equivalences have been successfully used for the analysis of information flow security properties in multilevel systems. They are fine enough to capture any unsecure behaviour that can give rise to information flow, but not so strict as to classify as unsecure behaviours which are instead correct.

Most of the time one wants to assess bisimilarity considering as silent (non observable) system internal moves. This kind of equivalence relations, called *weak bisimulations*, offer several advantages: they exploit abstraction from internal computation, in the process algebraic domain they are usually compositional with respect to parallel composition and other operators, and they can be exploited to minimize components with respect to their internal behaviour.

In this paper, we introduce a notion of weak bisimulation which combines some results from real-time and probabilistic models. We consider a model of Probabilistic Timed Automata inspired by the models in [11,35,2], where *schedulers* [35] are used in order to resolve the nondeterministic choices. We define a weak bisimulation equivalence relation for this model and we prove its decidability. The algorithm we propose is based on the well established partitioning technique [48,33], where large classes of potentially equivalent states are refined into smaller ones. More precisely, we partition the equivalence classes composed by the timed regions [2] of a Probabilistic Timed Automaton when they contain configurations from which it is possible to reach a certain class with different probabilities. In a certain sense, such a methodology extends the technique of [20] for real-time systems, in order to combine it with the one for probabilistic systems in [7].

On the one hand, a notion of weak bisimulation for Probabilistic Timed Automata allows us to prove behaviour inclusion (hence to define behavioural properties) for systems where both probability and time play a role. As we will show in the last section, one may, for example, define bisimulation based properties in order to capture timing attacks in probabilistic protocols. As another example, in [39,42] we developed a framework based on behavioural equivalence to analyze information flow security properties for Probabilistic Timed Automata. In particular, we were able to describe systems exhibiting a covert channel due to the combination of probability and time, that neither a formalism with only probability nor a formalism with only time can express.

On the other hand, weak bisimulation can be used as a minimization technique for reducing the state space of a system. Actually, automata based formalisms are amenable to formal analysis such as model checking, and state space reduction is one of the main research issues in that field.

From the computational point of view, we prove that strong and weak bisimulations are decidable, respectively, in exponential and double exponential time on the size of the Probabilistic Timed Automaton. These results are in agreement with the results for the untimed version. Actually strong and weak bisimulations are decidable in polynomial and exponential time, respectively, for the class of Probabilistic Automata, but the region graph has an exponential size with respect to the size of the considered Timed Automaton. Moreover, the algorithm we propose uses a symbolic representation for checking bisimilarity. To the best of our knowledge this is the first paper that gives an algorithm for deciding weak bisimulation that makes use of a symbolic representation.

1.1. Summary

The remainder of this paper is organized as follows. In Section 2 we recall some basic definitions of Labeled Transition Systems, Probabilistic Automata and Timed Automata, together with notions of weak bisimulations in the different models. We show that weak bisimulation for Probabilistic Automata and Timed Automata is preserved in the purely nondeterministic model obtained by abstracting away from probabilities and time. In Section 3 we introduce the framework of Probabilistic Timed Automata, we define their semantics, introduce the weak bisimulation relation, and again show that weak bisimulation is preserved with respect to the models of Timed Automata and Probabilistic Automata. In Section 4 we give a decision procedure based on backward analysis for verifying the weak bisimulation relation for the model of Probabilistic Timed Automata, and we develop the theory needed for assessing the correctness of our procedure. As a toy example, in Section 6 we use the means we have developed to model and capture a timing attack on the dining cryptographers protocol. Finally, in Section 8 we draw some conclusions.

2. Preliminaries

The models used in this paper are all based on *Labeled Transition Systems* (LTSs), also called *automata*. These have turned out to be an intuitive and powerful framework for the analysis of concurrent systems, and they have been extended with probability and time.

Different types of LTSs give rise to different notions of external behaviour. Informally, the *external behaviour* of an LTS, also called *visible behaviour*, is given by its sequences of external actions. A special invisible action τ is considered. The external behaviour of a timed LTS (Timed Automaton) also considers the passage of time as an externally observable action. In probabilistic LTSs (Probabilistic Automata) the probability of performing each action is taken into account. Actually, the measure is associated with each measurable set of sequences of actions.

$e_1 = (q_0, b, q_1)$
$e_2 = (q_0, a, q_2)$
$e_3 = (q_0, a, q_3)$

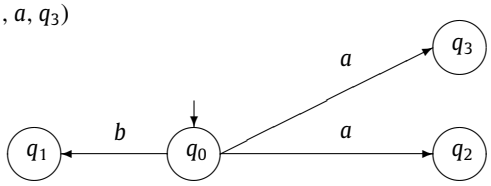

Fig. 1. Example of LTS.

We may show that the external behaviour of an automaton is contained in or is equal to the external behaviour of another one by proving behaviour inclusion or equality. However, this is a rather complex task. In this case, simulation and bisimulation relations can be extremely useful. These relations compare the stepwise behaviour of systems and when two systems are shown to be bisimilar, then there is also behaviour inclusion. Intuitively, the idea behind bisimilar states is that each step one of them may take can be mimicked by the other.

2.1. The possibilistic model

We recall some basic notions of finite *Labeled Transition Systems* together with a notion of weak bisimulation.

Definition 1. A *Labeled Transition System* (LTS) is a tuple $A = (\Sigma, Q, q_0, \delta)$, where Σ is a set of labels, Q is a finite set of states with $q_0 \in Q$ the initial one. The set of transitions is given by $\delta \subseteq Q \times (\Sigma \cup \{\tau\}) \times Q$, where τ represents an internal silent move.

Notice that we use the special symbol τ (not contained in Σ) to denote internal (unlabeled) actions.

Given two states $q_i, q_j \in Q$ of $A = (\Sigma, Q, q_0, \delta)$, there is a step from q_i to q_j labeled with a (denoted $q_i \xrightarrow{a} q_j$) if $(q_i, a, q_j) \in \delta$. With S_T we denote the set of terminal states of A, namely $S_T = \{q \in Q \mid \forall q' \text{ and } \forall a \in \Sigma \cup \tau, (q, a, q') \notin \delta\}$.

An *execution fragment* of A is a finite sequence of steps $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} q_k$, where $q_0, \ldots, q_k \in Q$ and $a_i \in \Sigma \cup \{\tau\}$. With $ExecFrag_A$ we denote the set of execution fragments of A, and with $ExecFrag_A(q)$ we denote the set of execution fragments of A starting from q. We define $last(\sigma) = q_k$ and $|\sigma| = k$. For any $j \leq |\sigma|$, with σ^j we define the sequence of steps $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_j} q_j$. The execution fragment σ is called *maximal* iff $last(\sigma) \in S_T$.

An *execution* of A is either a maximal execution fragment or an infinite sequence of steps $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$, where $q_0, q_1 \ldots \in Q$ and $a_1, a_2, \ldots \in \Sigma \cup \{\tau\}$. We denote with $Exec_A$ the set of executions of A and with $Exec_A(q)$ the set of executions of A starting from q. Finally, with $\sigma\uparrow$ we denote the set of executions σ' such that $\sigma \leq_{prefix} \sigma'$, where $\leq_{prefix}$ is the usual prefix relation over sequences.

Example 1. In Fig. 1 we show an example of LTS. From the initial state q_0, the LTS may perform transition e_1 labeled with b reaching state q_1 or it can nondeterministically perform transitions e_2 and e_3, labeled with a and leading to states q_2 and q_3, respectively. An example of execution of the LTS in Fig. 1 is $\sigma = q_0 \xrightarrow{a} q_2$.

Behavioural equivalence. As a relation of observational equivalence for LTS, we now introduce the notion of weak bisimulation [46].

The *bisimilarity* of two systems is based on the idea of mutual step-by-step simulation. Intuitively, two systems A and A' are bisimilar, if whenever one of the two systems executes a certain action and reaches a state q, the other system is able to simulate this single step by executing the same action and reaching a state q' which is again bisimilar to q. A *weak bisimulation* is a bisimulation which abstracts away from τ (internal) moves. In this sense, whenever a system simulates an action of the other system, it can also execute some internal τ actions before and after the execution of that action.

In order to abstract away from τ moves, Milner [46] introduced the notion of observable step, which consists of a single *visible* action a preceded and followed by an arbitrary number (including zero) of internal moves. Such moves are described by a *weak* transition relation $\Longrightarrow$, defined as $\stackrel{a}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^*$, where $\longrightarrow$ is the classical strong relation, and $\stackrel{\tau}{\Longrightarrow} = (\xrightarrow{\tau})^*$. It is worth noting that, with such a definition, a weak internal transition $\stackrel{\tau}{\Longrightarrow}$ is possible even without performing any internal action.

Definition 2. Let $A = (\Sigma, Q, q_0, \delta)$ be a LTS. A *weak bisimulation* on A is an equivalence relation $\mathcal{R} \subseteq Q \times Q$ such that for all $(p, q) \in \mathcal{R}$ and $\forall a \in \Sigma \cup \{\tau\}$ it holds that:

- if $p \xrightarrow{a} p'$, then there exists q' such that $q \stackrel{a}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$;
- conversely, if $q \xrightarrow{a} q'$, then there exists p' such that $p \stackrel{a}{\Longrightarrow} p'$ and $(p', q') \in \mathcal{R}$.

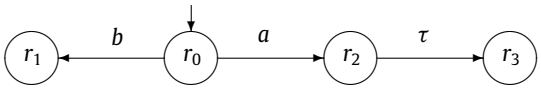

Fig. 2. Example of weak bisimulation for LTSs.

Two states p, q are called *weakly bisimilar* on A (denoted $p \approx_A q$) iff $(p, q) \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$.

Two LTSs $A = (\Sigma, Q, q_0, \delta)$ and $A' = (\Sigma', Q', q_0', \delta')$, such that $Q \cap Q' = \emptyset$, are called weakly bisimilar (denoted by $A \approx A'$) if, given the LTS

$$\hat{A} = (\Sigma \cup \Sigma', Q \cup Q' \cup \{\hat{q}\}, \hat{q}, \delta \cup \delta' \cup \{(\hat{q}, \tau, q_0), (\hat{q}, \tau, q_0')\}),$$

it holds $q_0 \approx_{\hat{A}} q_0'$.

Note that it is always possible to obtain $Q \cap Q' = \emptyset$ by state renaming.

Example 2. Let A be the LTS in Fig. 1 and A' be the LTS in Fig. 2. It holds that $A \approx A'$. Intuitively, from the initial states q_0 and r_0 the two LTSs may either perform a step labeled with a or b and then reach a state where no other visible steps may be performed.

2.2. The probabilistic model

We introduce the formalism for probabilistic systems together with a notion of weak bisimulation. We also show how to remove probabilities in order to get a nondeterministic system, and that weak bisimulation is preserved when reducing to the possibilistic model. We shall use the same terminology for operators and bisimulation in the different models when this does not give rise to ambiguity.

Our probabilistic model is a slight variant of the Markov Decision Processes (MDPs) of [9,30] and the Probabilistic Automata of [53,56].

Actually, we give a definition of Probabilistic Automata which is a bit closer to the one of MDPs, where probability distributions are defined over the set of transitions. Intuitively, our definition derives from the effort to find the more natural way to add probabilities to a given LTS (we take the definition for LTSs and spread probabilities over transitions). This choice, while requiring some particular attention in the definition of the semantics, allows for a simple and intuitive backward reconstruction of the original LTS by simply removing probabilities from a PA.

In [53,56], instead, probability distributions are defined over the set of target states and associated with each transition. It is possible, however, to give translations from these models to ours, and vice versa.

Definition 3. A *Probabilistic Automaton* (PA) is a tuple $A = (\Sigma, Q, q_0, \delta, \Pi)$, where:

- Σ is a finite alphabet of actions;
- Q is a finite set of states and $q_0 \in Q$ is the initial state;
- $\delta \subseteq Q \times (\Sigma \cup \{\tau\}) \times Q$ is a finite set of transitions;
- $\Pi = \{\pi_1, \ldots, \pi_n\}$ is a finite set of probability distributions as functions $\pi_i : \delta \to [0, 1]$, for any $i = 1, \ldots, n$, where $\pi_i(e)$ is the probability of performing transition e according to distribution π_i.

For a state q of a PA, we denote with $\delta(q)$ the set of transitions with q as source state, i.e. the set $\{(q_1, a, q_2) \in \delta \mid q_1 = q\}$. We require that $\sum_{e \in \delta(q)} \pi_i(e) = 1$ for any i and q. Moreover, we assume that for all e_j there exists some π_i such that $\pi_i(e_j) > 0$.

Transition steps, execution fragments and executions of a PA are defined as for LTSs.

The probability of executing a transition step from a state q is chosen, among all the transitions in $\delta(q)$, according to the values returned by some distribution π.

Intuitively, the probability distribution of a transition step is chosen nondeterministically. Hence, executions and execution fragments of a PA arise by resolving both the nondeterministic and the probabilistic choices [35]. We need a notion of *scheduler* to resolve the nondeterminism that arises when choosing a distribution π within the set Π.

A *scheduler* of a PA A is a partial function F assigning a distribution $\pi \in \Pi$ to each finite sequence σ in $ExecFrag_A$. Namely, $F : ExecFrag_A \to \Pi$. Given a scheduler F and an execution fragment σ, we assume that F is defined for σ if and only if $\exists q \in Q$ and $a \in \Sigma \cup \{\tau\}$ such that $last(\sigma) \xrightarrow{a} q$.

For a scheduler F we define $ExecFrag_A^F$ (resp. $Exec_A^F$) as the set of execution fragments (resp. executions) $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots$ of A such that, for any i, $\pi_i((q_{i-1}, a_i, q_i)) > 0$ where $\pi_i = F(\sigma^{i-1})$.

Given a scheduler F and an execution fragment $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q_k$ in $ExecFrag_A^F$, if $k = 0$ we put $P_A^F(\sigma) = 1$, else, if $k \geq 1$, we define $P_A^F(\sigma) = P_A^F(\sigma^{k-1}) \cdot F(\sigma^{k-1})(e)$, where $e = (q_{k-1}, a_k, q_k)$.

$e_1 = (q_0, b, q_1)$ $\quad \pi(e_1) = \frac{1}{6}$
$e_2 = (q_0, a, q_2)$ $\quad \pi(e_2) = \frac{1}{3}$
$e_3 = (q_0, a, q_3)$ $\quad \pi(e_3) = \frac{1}{2}$

$\sum_{e \in \delta(q_0)} \pi(e) = 1$

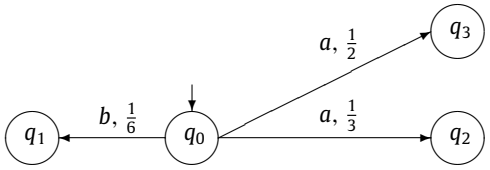

Fig. 3. Example of PA.

Assuming the basic notions of probability theory (see e.g. [25,53]), we define the probability space on the executions starting from a given state $q \in Q$ as follows. Given a scheduler F, let $Exec_A^F(q)$ be the set of executions in $Exec_A^F$ starting from q, $ExecFrag_A^F(q)$ be the set of execution fragments in $ExecFrag_A^F$ starting from q, and $\sigma Field_A^F(q)$ be the smallest sigma field on $Exec_A^F(q)$ that contains the basic cylinders $\sigma\uparrow$, where $\sigma \in ExecFrag_A^F(q)$. The probability measure $Prob_A^F$ is the unique measure on $\sigma Field_A^F(q)$ such that $Prob_A^F(\sigma\uparrow) = P_A^F(\sigma)$.

Given a scheduler F, a state q and a set of states $Q' \subseteq Q$, with $Exec_A^F(q, Q')$ we denote the set of executions starting from q that cross a state in the set Q'. Namely, $Exec_A^F(q, Q') = \{\sigma \in Exec_A^F(q) \mid last(\sigma^i) \in Q', \text{ for some } i\}$.

If a PA does not allow nondeterministic choices it is said to be *fully probabilistic*.

Definition 4. Given a PA $A = (\Sigma, Q, q_0, \delta, \Pi)$, we say that A is *fully probabilistic* if $|\Pi| = 1$.

Example 3. In Fig. 3 we show an example of a PA with $\Pi = \{\pi\}$. Intuitively, from the initial state q_0, the PA performs probabilistically transitions e_1, e_2 or e_3 with probabilities $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{1}{2}$, respectively. Note that $|\Pi| = 1$, thus the PA is *fully probabilistic*. As a consequence, nondeterministic choices are not performed since every scheduler can return the only distribution π.

Examples of executions of the PA in Fig. 3 are $\sigma_1 = q_0 \xrightarrow{a} q_3$ and $\sigma_2 = q_0 \xrightarrow{b} q_1$ with $P(\sigma_1) = \frac{1}{2}$ (we may omit indexes A and F from $P_A^F(\sigma)$ when this does not give rise to ambiguity) and $P(\sigma_2) = \frac{1}{6}$. To make the presentation easier to understand, when there is just one probability distribution, we put probabilities also on the arcs of the automaton. Note, however, that probabilities are not a part of the transition label.

The next proposition derives from results in [15].

Proposition 1. *Let A_1 and A_2 be two PAs and Q_1 and Q_2 be two subsets of states of A_1 and A_2, respectively. It is decidable in exponential time whether for any scheduler F of A_1 there exists a scheduler F' of A_2 such that $Prob_{A_1}^F(Exec_{A_1}^F(q_1, Q_1)) = Prob_{A_2}^{F'}(Exec_{A_2}^{F'}(q_2, Q_2))$, where q_1 and q_2 are states of A_1 and A_2, respectively. If for any $\sigma \in Exec_{A_1}^F(q_1, Q_1) \cup Exec_{A_2}^{F'}(q_2, Q_2)$ it holds that $last(\sigma^1) \in Q_1 \cup Q_2$, then the problem is decidable in polynomial time.*

Behavioural equivalence. For the definition of weak bisimulation in the fully probabilistic setting, Baier and Hermanns [7] replace Milner's weak internal transitions $q \stackrel{\tau}{\Longrightarrow} q'$ by the probability $Prob(q, \tau^*, q')$ of reaching state q' from q via internal moves. Similarly, for visible actions a, Baier and Hermanns define $q \stackrel{a}{\Longrightarrow} q'$ by means of the probability $Prob(q, \tau^* a \tau^*, q')$. As already mentioned, the probabilistic model we obtain for PAs, when a scheduler is given that resolves the nondeterministic choices, is that of fully probabilistic systems.

Hence, the definition of weak bisimulation for PAs is inspired by the ones in [7,1]. The only difference is given by the introduction of schedulers in order to obtain a fully probabilistic model.

In the following, $q \in Q$ is a state of A, $\mathcal{C} \subseteq Q$ a set of states and $\hat{a}$ stands for a if $a \in \Sigma$ and for ε (the empty string) if $a = \tau$.

Let $Exec_A^F(q, \tau^*\hat{a}\tau^*, \mathcal{C})$ be the set of executions that lead to a state in $\mathcal{C}$ via a sequence belonging to the set of sequences $\tau^*\hat{a}\tau^*$ starting from state q and crossing the states equivalent to q. We define the probability $Prob_A^F(q, \tau^*\hat{a}\tau^*, \mathcal{C}) = Prob_A^F(Exec_A^F(q, \tau^*\hat{a}\tau^*, \mathcal{C}))$.

Definition 5. Let $A = (\Sigma, Q, q_0, \delta, \Pi)$ be a PA. A *weak bisimulation* on A is an equivalence relation $\mathcal{R}$ on Q such that, for all schedulers F and $(q, q') \in \mathcal{R}$, there exists a scheduler F' such that for all $\mathcal{C} \in Q/\mathcal{R}$ and $a \in \Sigma \cup \{\tau\}$ it holds:

$$Prob_A^F(q, \tau^*\hat{a}\tau^*, \mathcal{C}) = Prob_A^{F'}(q', \tau^*\hat{a}\tau^*, \mathcal{C})$$

and vice versa.
Two states q, q' are called *weakly bisimilar* on A (denoted $q \approx_A q'$) iff $(q, q') \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$.

$$e_1 = (r_0, b, r_1) \qquad \pi(e_1) = \tfrac{1}{3}$$
$$e_2 = (r_0, a, r_2) \qquad \pi(e_2) = \tfrac{1}{3}$$
$$e_3 = (r_0, a, r_3) \qquad \pi(e_3) = \tfrac{1}{3}$$

$$\textstyle\sum_{e \in \delta(r_0)} \pi(e) = 1$$

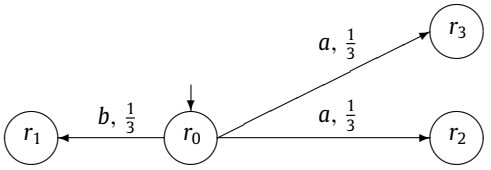

Fig. 4. Example of weak bisimulation for PAs.

As for LTSs, in order to define weak bisimulation for PAs we resort to a disjoint sum of automata.

Definition 6. Let $A = (\Sigma, Q, q_0, \delta, \Pi)$ and $A' = (\Sigma', Q', q_0', \delta', \Pi')$ be two PAs such that $Q \cap Q' = \emptyset$. Consider the PA

$$\hat{A} = (\Sigma \cup \Sigma', Q \cup Q' \cup \{\hat{q}\}, \hat{q}, \delta \cup \delta' \cup \{(\hat{q}, \tau, q_0), (\hat{q}, \tau, q_0')\}, \hat{\Pi}),$$

where $\hat{\pi_1}, \hat{\pi_2} \in \hat{\Pi}$ such that $\hat{\pi_1}(e) = 1$ if $e = (\hat{q}, \tau, q_0)$, 0 otherwise, and $\hat{\pi_2}(e) = 1$ if $e = (\hat{q}, \tau, q_0')$, 0 otherwise; moreover, for each couple $(\pi, \pi') \in \Pi \times \Pi'$, $\hat{\pi} \in \hat{\Pi}$ such that

$$\hat{\pi}(e) = \begin{cases} \pi(e) & \text{if } e \in \delta \\ \pi'(e) & \text{if } e \in \delta'. \end{cases}$$

A and A' are weakly bisimilar (denoted by $A \approx A'$) if it holds $q_0 \approx_{\hat{A}} q_0'$.

Note that each distribution $\hat{\pi}$ is well defined since $Q \cap Q' = \emptyset$ implies $\delta \cap \delta' = \emptyset$.

Example 4. Let A' be the PA in Fig. 4, and A the PA in Fig. 3. We have that $A \not\approx A'$ since from the initial state q_0, PA A may perform a step labeled with b and reach a terminal state with probability $\frac{1}{6}$, while from r_0, PA A' reaches a terminal state through a transition labeled with b with probability $\frac{1}{3}$.

Removing probabilities. Given a PA A, we call *possibilistic abstraction* of A (written $unprob(A)$) the LTS obtained from A by simply removing the set of probability distributions Π. This can be easily done since we assumed that for each transition of A there is at least one probability distribution which assigns to such a transition a probability greater than 0 (see Definition 3).

Definition 7. Given a PA $A = (\Sigma, Q, q_0, \delta, \Pi)$, $unprob(A) = (\Sigma, Q, q_0, \delta)$.

Example 5. Let A be the PA in Fig. 3. If we remove probabilities from A the possibilistic abstraction $unprob(A)$ can be found in Fig. 1. Actually, the PA in Fig. 3 could be seen as a probabilistic specification of the LTS in Fig. 1.

The following conservativeness result holds.

Lemma 1. *Given PAs A and A', $A \approx A' \Rightarrow unprob(A) \approx unprob(A')$.*

Proof. Let us assume $A = (\Sigma, Q, q_0, \delta, \Pi)$, $A' = (\Sigma', Q', q_0', \delta', \Pi')$ and $\hat{A}$ constructed as in Definition 6. Since $A \approx A'$ for a weak bisimulation $\mathcal{R}$, we have that for all schedulers F and $(q, r) \in \mathcal{R}$, there exists a scheduler F' such that for all $\mathcal{C} \in Q \cup Q'/\mathcal{R}$ and $a \in \Sigma \cup \{\tau\}$, $Prob_{\hat{A}}^{F}(q, \tau^* \hat{a} \tau^*, \mathcal{C}) = Prob_{\hat{A}}^{F'}(r, \tau^* \hat{a} \tau^*, \mathcal{C})$. Now, if $Prob_{\hat{A}}^{F}(q, a, q') > 0$ for some $q' \in \mathcal{C}$, then there exists a state r' and a scheduler F' such that $Prob_{\hat{A}}^{F'}(r, \tau^* \hat{a} \tau^*, r') = Prob_{\hat{A}}^{F}(q, a, q') > 0$. Therefore, for each step $q \xrightarrow{a} q'$ there exists r' such that $r \stackrel{a}{\Longrightarrow} r'$ and, since q' and r' are in the same equivalence class, $\mathcal{R}$ is also a weak bisimulation for $\hat{Q}_{np}$, where $\hat{Q}_{np}$ is the set of states of the LTS constructed as in Definition 2 starting from $unprob(A)$ and $unprob(A')$. The same holds if we exchange the roles of q and r. □

2.3. The timed model

We introduce the formalism for timed systems together with a notion of weak bisimulation. We also show how to remove time in order to get a nondeterministic system, and that weak bisimulation is preserved when reducing to the untimed model.

Let us assume a set X of nonnegative real variables called *clocks*. A *valuation* over X is a mapping $v : X \to \mathbb{R}^{\geq 0}$ assigning real values to clocks. For a valuation v and a time value $t \in \mathbb{R}^{\geq 0}$, let $v + t$ denote the valuation such that $(v + t)(x) = v(x) + t$, for each clock $x \in X$.

The set of *constraints* over X, denoted $\Phi(X)$, is defined by the following grammar, where ϕ ranges over $\Phi(X)$, $x \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, \leq, \geq, >\}$:

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

We write $v \models \phi$ when *the valuation v satisfies the constraint ϕ*. Formally, it holds that $v \models true$, $v \models x \sim c$ iff $v(x) \sim c$, $v \models \phi_1 \wedge \phi_2$ iff $v \models \phi_1$ and $v \models \phi_2$.

Let $B \subseteq X$; with $v[B]$ we denote the valuation resulting after resetting all clocks in B. More precisely, $v[B](x) = 0$ if $x \in B$, $v[B](x) = v(x)$, otherwise. Finally, with $\mathbf{0}$ we denote the valuation with all clocks reset to 0, namely $\mathbf{0}(x) = 0$ for all $x \in X$.

Now, we are able to recall the definition of Timed Automata [3].

Definition 8. A *Timed Automaton* (TA) is a tuple $A = (\Sigma, X, Q, q_0, Inv, \delta)$, where:

- Σ is a finite alphabet of actions;
- X is a finite set of nonnegative real variables called clocks;
- Q is a finite set of states and $q_0 \in Q$ is the initial state;
- $Inv : Q \to \Phi(X)$ is the invariant function assigning to each state a formula (called *state invariant*) that must hold in any instant in which the state is enabled;
- $\delta \subseteq Q \times (\Sigma \cup \{\tau\}) \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions;

For a state q, we denote with $\delta(q)$ the set of transitions with q as source state, i.e. the set $\{(q_1, a, \phi, B, q_2) \in \delta \mid q_1 = q\}$. We also require that $(q_0, \mathbf{0}) \models Inv(q_0)$.

A *configuration* of a TA A is a pair (q, v), where $q \in Q$ is a state of A, v is a valuation over X and $v \models Inv(q)$. The initial configuration of A is represented by $(q_0, \mathbf{0})$.

There is a discrete *transition step* from a configuration $s_i = (q_i, v_i)$ to a configuration $s_j = (q_j, v_j)$ through action $a \in \Sigma \cup \{\tau\}$, written $s_i \xrightarrow{a} s_j$, if there is a transition $e = (q_i, a, \phi, B, q_j) \in \delta$ such that $v_i \models \phi$, $v_j = v_i[B]$, $v_i \models Inv(q_i)$ and $v_j \models Inv(q_j)$.

There is a *time step* from a configuration $s_i = (q_i, v_i)$ to a configuration $s_j = (q_j, v_j)$ through time $t \in \mathbb{R}^{>0}$, written $s_i \xrightarrow{t} s_j$, if $q_j = q_i$, $v_j = (v_i + t)$ and $\forall t' \in [0, t]\ v_i + t' \models Inv(q_i)$.

An *execution fragment* of A is a finite sequence of steps $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} s_k$, where $s_0, \ldots, s_k$ are configurations, and $\alpha_i \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$. With $ExecFrag_A$ we denote the set of execution fragments of A, and with $ExecFrag_A(s)$ we denote the set of execution fragments of A starting from configuration s. We define $last(\sigma) = s_k$ and $|\sigma| = k$. For any $j \leq |\sigma|$, with σ^j we define the sequence of steps The execution fragment σ is called *maximal* iff there is not any configuration s and $\alpha \in \Sigma \cup \{\tau\}$ such that $\sigma \xrightarrow{\alpha} s$.

With $\mathcal{S}_A$ we denote the set of configurations reachable by A, more precisely, $\mathcal{S}_A = \{last(\sigma) \mid \sigma \in ExecFrag_A(s_0)\}$.

An *execution* of A is either a maximal execution fragment or an infinite sequence of steps $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots$, where $s_0, s_1 \ldots \in \mathcal{S}_A$ and $\alpha_1, \alpha_2, \ldots \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{\geq 0}$. We denote with $Exec_A$ the set of executions of A and with $Exec_A(s)$ the set of executions of A starting from s.

Example 6. In Fig. 5 we show an example of a TA. In this example, and in the following ones, we omit the condition on a transition when the condition is *true*. We write state invariants in bold face and, again, we omit them when they are equal to *true*.

From the initial state q_0, the TA may always perform some time step and update the value of clock x. Transition e_1 labeled with b can be performed by the TA if the value of clock x is less than or equal to 5. Transitions e_2 and e_3, labeled with a and leading to states q_2 and q_3, respectively, may be performed at any time (their guard condition is *true*). Note, however, that if transition e_2 is performed, then the value of the clock x is reset to 0.

An example of execution fragment of the TA in Fig. 5 is $(q_0, 0) \xrightarrow{9.7} (q_0, 9.7) \xrightarrow{a} (q_2, 0) \xrightarrow{3.2} (q_2, 3.2)$, where (q, t) represents the configuration composed by the state q and the valuation v such that $v(x) = t$.

Regions. We recall the definitions of clock equivalence [3] and the theory of clock zones [12]. Clock equivalence is an equivalence relation of finite index permitting us to group sets of evaluations and to have decidability results. Unfortunately, the number of equivalence classes is exponential w.r.t. the size of the TA. A more efficient symbolic representation by means of clock zones is introduced (see [26,12]), and this helps in representing parts of the state space in practice.

Let A be a TA; with C_A we denote the largest constant appearing in A.

Let us consider the equivalence relation $\approx$ over clock valuations containing precisely the pairs (v, v') such that:

- for each clock x, either $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, or both $v(x)$ and $v'(x)$ are greater than C_A;
- for each pair of clocks x and y with $v(x) \leq C_A$ and $v(y) \leq C_A$ it holds that $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$ (where $fract(\cdot)$ is the fractional part);
- for each clock x with $v(x) \leq C_A$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.

$e_1 = (q_0, b, 0 \leq x \leq 5, \emptyset, q_1)$
$e_2 = (q_0, a, true, \{x\}, q_2)$
$e_3 = (q_0, a, true, \emptyset, q_3)$

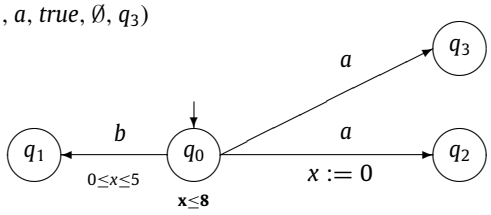

Fig. 5. Example of TA.

As proved in [3], $v \approx v'$ implies that, for any $\phi \in \Phi(X)$ with constants less than or equal to C_A, $v \models \phi$ iff $v' \models \phi$. With $[v]$ we denote the equivalence class $\{v' \mid v \approx v'\}$. The set of equivalence classes is finite. We recall the definition of *clock zone* and its properties. For more details see [12,26].

The set of clock zones on X (denoted with $\Psi(X)$) is the set of formulae ψ such that

$$\psi ::= true \mid false \mid x \sim c \mid x - y \sim c' \mid \psi_1 \wedge \psi_2$$

where $\sim \in \{<, \leq, \geq, >\}$, $c \in \mathbb{N}$, $c' \in \mathbb{Z}$ and $x, y \in X$.

With $\Psi_C(X)$ we denote the set of clock zones in $\Psi(X)$ that use integer constants in $[-C, C]$. We will write $v[x := c]$ to denote the valuation such that $(v[x := c])(x) = c$ and $(v[x := c])(y) = v(y)$, for any $y \neq x$. Moreover, we will write $v \models \psi$ if ψ is *true* when valuating each clock x with $v(x)$. Hence, two clock zones are equivalent if are satisfied by the same set of valuations.

Given a set $B = \{x_1, \ldots, x_m\} \subseteq X$, with $\exists B.\psi$ we denote the formula $\exists x_1. \cdots . \exists x_m.\psi$, with $\forall x.\psi$ the formula $\neg\exists t.\neg\psi$ and, moreover, with $B = 0$ we denote the formula $\bigwedge_{x \in B} x = 0$ and with $\psi[B := B + t]$ we denote the formula ψ where each occurrence of $x \in B$ is substituted with $x + t$.

Known properties of clock zones are expressed by the following propositions. We note that a clock zone is a convex space and can be represented by a Difference Bound Matrix (DBM). Forward operators pose problems if diagonal constraints are allowed (as shown in [12]). Hence we consider backward operators for which the approximation and symbolic verification are correct (see [12]).

Proposition 2. *Let ψ be a clock zone in $\Psi(X)$ and $x \in X$. A clock zone ψ' in $\Psi(X \setminus x)$ is computable in polynomial time such that ψ' is equivalent to the set of v such that $v[x := c] \models \psi$, for some $c \in \mathbb{R}^{\geq 0}$.*

With abuse of notation, from now on, we will write $\exists x.\psi$ to denote its equivalent clock zone.

The following proposition gives an upper bound to the number of clock zones.

Proposition 3. *There exists $\Psi' \subset \Psi_C(X)$ with exponential cardinality w.r.t. C and $|X|$ such that each clock zone in $\Psi_C(X)$ is equivalent to a clock zone in Ψ'.*

Hence from now on we can suppose that the set of clock zones in $\Psi_C(X)$ has finite cardinality.

Definition 9. Let A be a TA with states in Q and clocks in X; a *region* of A is a pair (q, ψ) where $q \in Q$ and $\psi \in \Psi(X)$.

Example 7. As an example the clock zone $x > 0 \wedge x \leq 10$ expresses the set of valuations assigning to x a real value in $(0, 10]$. Hence the region $(q_1, x > 0 \wedge x \leq 10)$ of the TA of Fig. 5 represents the set of configurations (q_1, v) such that $v(x) \in (0, 10]$.

The following proposition states that the set of configurations reachable by performing either a discrete or a time step starting from a set of configurations expressed by a region, is a region.

Proposition 4. *Given a region (q, ψ) and a transition $e = (q', a, \phi, B, q)$, the set of configurations $\{(q', v) \mid v \models \phi \wedge Inv(q')$ and $v[B] \models \psi \wedge Inv(q)\}$ from which it is possible to reach a configuration within the region (q, ψ) by a discrete step triggered by e, is equal to the region $(q', \phi \wedge Inv(q') \wedge \exists B.(\psi \wedge B = 0 \wedge Inv(q)))$.*

The set of configurations $\{(q, v) \mid v + t \models \psi$ and $v + t' \models Inv(q)$ for some $t \in \mathbb{R}^{>0}$ and for all $t' \in [0, t]\}$ from which it is possible to reach a configuration expressed by (q, ψ) by means of a time step, is equal to the region $(q, \exists t.t > 0 \wedge \psi[X := X + t] \wedge \forall t' \in [0, t].Inv(q)[X := X + t'])$. Moreover, if $\psi \in \Psi_C(X)$, then $\exists t.t > 0 \wedge \psi[X := X + t] \wedge \forall t' \in [0, t].Inv(q)[X := X + t'] \in \Psi_C(X)$.

Example 8. As an example the set of configurations that can reach with a transition step a configuration expressed by the region $(q_1, x > 0 \wedge x \leq 5)$ of the TA of Fig. 5, is expressed by the region $(q_0, x > 0 \wedge x \leq 5)$. Moreover, the set of configurations that can reach with a time step a configuration expressed by the region $(q_0, x > 5 \wedge x \leq 8)$ of the TA of Fig. 5 is expressed by the region $(q_0, x \geq 0 \wedge x < 8)$.

$e_1 = (r_0, b, 0 \le y \le 3, \emptyset, r_1)$
$e_2 = (r_0, a, true, \{y\}, r_2)$
$e_3 = (r_0, a, true, \emptyset, r_3)$

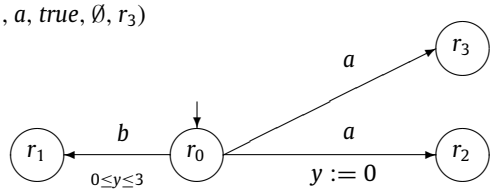

Fig. 6. Example of weak bisimulation for TAs.

Now, it is obvious that, due to the discrete step, we can reach regions that contain constants that are not in $[-C_A, C_A]$. Thus, we need an approximation for discrete steps.

If ψ is a clock zone of A, we denote with $Ap_A(\psi)$ the set $\{v \mid \exists v' \approx v \text{ s.t. } v' \models \psi\}$.

The following proposition, proved in [12], states that Ap_A returns a clock zone.

Proposition 5 (*Approximation*). $Ap_A(\psi) \in \Psi_{C_A}(X)$.

The following theorem, proved in [12], states the correctness of the operator Ap.

Theorem 1. *The sequence of steps* $(q_0, v_0) \xrightarrow{\alpha_0} (q_1, v_1) \xrightarrow{\alpha_1} \cdots$ *is an execution of A iff there exists a sequence of regions* $(q_0, \psi_0), (q_1, \psi_1) \ldots$ *such that, for all i,* $v_i \models \psi_i$ *and, if* $\alpha_i \in \Sigma \cup \{\tau\}$, *then* $\psi_i = Ap_A(\phi \wedge Inv(q_i) \wedge \exists B.\psi_{i+1} \wedge B = 0 \wedge Inv(q_{i+1}))$ *for some transition* $(q_i, \alpha_i, \phi, B, q_{i+1})$, *and, otherwise,* $\psi_i = \exists t.t > 0 \wedge \psi_{i+1}[X := X + t] \wedge \forall t' \in [0, t].Inv(q_i)[X := X + t']$ *and* $q_{i+1} = q_i$. *Moreover, each* ψ_i *is computable in polynomial time w.r.t. C and* $|X|$.

Behavioural equivalence. The definition of weak bisimulation introduced for LTSs (see Definition 2) can be naturally adapted for TAs.

Definition 10. Let $A = (\Sigma, X, Q, q_0, Inv, \delta)$ be a TA. A *weak bisimulation* on A is an equivalence relation $\mathcal{R} \subseteq \mathcal{S}_A \times \mathcal{S}_A$ such that for all $(s, r) \in \mathcal{R}$ and $\forall \alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$ it holds that:

- if $s \xrightarrow{\alpha} s'$, then there exists r' such that $r \stackrel{\alpha}{\Longrightarrow} r'$ and $(s', r') \in \mathcal{R}$;
- conversely, if $r \xrightarrow{\alpha} r'$, then there exists s' such that $s \stackrel{\alpha}{\Longrightarrow} s'$ and $(s', r') \in \mathcal{R}$.

Two configurations s, r are called *weakly bisimilar* on A (denoted $s \approx_A r$) iff $(s, r) \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$. (Note that a weak time transition is of the form $\xrightarrow{\tau} \cdots \xrightarrow{\tau} \xrightarrow{t} \xrightarrow{\tau} \cdots \xrightarrow{\tau}$ with $t \in \mathbb{R}^{>0}$.)
Two TAs $A = (\Sigma, X, Q, q_0, Inv, \delta)$ and $A' = (\Sigma', X', Q', q_0', Inv', \delta')$ such that $Q \cap Q' = \emptyset$ and $X \cap X' = \emptyset$ are called weakly bisimilar (denoted by $A \approx A'$) if, given the TA

$$\hat{A} = (\Sigma \cup \Sigma', X \cup X', Q \cup Q' \cup \{\hat{q}\}, \hat{q}, \hat{Inv}, \delta \cup \delta' \cup \{(\hat{q}, \tau, true, \emptyset, q_0), (\hat{q}, \tau, true, \emptyset, q_0')\}),$$

with

$$\hat{Inv}(q) = \begin{cases} true & \text{if } q = \hat{q} \\ Inv(q) & \text{if } q \in Q \\ Inv'(q) & \text{if } q \in Q' \end{cases}$$

it holds that $(q_0, \mathbf{0}) \approx_{\hat{A}} (q_0', \mathbf{0})$, where the valuation $\mathbf{0}$ is defined over all clocks of the set $X \cup X'$.

Again, note that it is always possible to obtain $Q \cap Q' = \emptyset$ and $X \cap X' = \emptyset$ by state and clock renaming.

Example 9. Let A' be the TA in Fig. 6, and A the TA in Fig. 5. We have that $A \not\approx A'$ since from the initial configuration $(q_0, x = 0)$, TA A may perform a time step of duration 5 and then, from configuration $(q_0, x = 5)$ a discrete transition step labeled with b. On the contrary, from $(r_0, y = 0)$, TA A', by performing a time step of duration 5, reaches configuration $(r_0, y = 5)$ from which a discrete transition step labeled with b is not possible anymore.

Removing time. Given a TA A, we call *untime*(A) the LTS obtained as the *region automaton* of A. Intuitively, the region automaton (see [3]) is obtained by considering timed regions as states. Note that in the region automaton there might be an admissible step between regions R and R' with symbol a also if there is an admissible run $s \xrightarrow{t} s'' \xrightarrow{a} s'$ of the TA such that $t \in \mathbb{R}^{>0}$ and where $s \in R$ and $s' \in R'$. We use the silent label τ to label all the transitions of the LTS *untime*(A) arising from time steps of the TA A. Intuitively, time steps are not visible anymore in the untimed setting.

Definition 11. Given a TA $A = (\Sigma, X, Q, q_0, Inv, \delta)$, we define the LTS *untime*(A) as the tuple $(\Sigma, Q \times \mathcal{V}, (q_0, [\mathbf{0}]), \delta')$, where $\mathcal{V}$ is the set of equivalence classes of the valuations of A:

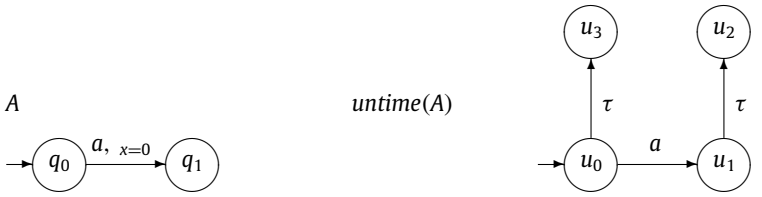

Fig. 7. Example of *untime*(*A*).

- $((q, [v]), \tau, (q, [v'])) \in \delta'$ iff $v' = v + t$ for some time $t \in \mathbb{R}^{>0}$ and $v' \models Inv(q)$ (note that we do not need to check each time in $[0, t]$ since we are considering convex spaces);
- $((q, [v]), a, (q', [v'])) \in \delta'$ iff $(q, a, \phi, B, q') \in \delta$, $v \models \phi \wedge Inv(q)$ and $v' = v[B] \models Inv(q')$.

Example 10. In Fig. 7 we show the TA A and its untimed version, the LTS *untime*(A). States u_0, u_1, u_2 and u_3 correspond, respectively, to the pairs $(q_0, [v_0])$, $(q_1, [v_0])$, $(q_1, [v_1])$ and $(q_0, [v_1])$, where $[v_0] = \{v \mid v(x) = 0\}$ and $[v_1] = \{v \mid v(x) > 0\}$. In the figure we omitted self-loop transitions (u_i, τ, u_i) for $i \in \{2, 3\}$.

Given an execution $\sigma = (q_0, v_0) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} (q_n, v_n)$ of a TA A, with $[\sigma]$ we denote the corresponding execution $(q_0, [v_0]) \xrightarrow{\alpha'_1} \cdots \xrightarrow{\alpha'_n} (q_n, [v_n])$ of *untime*(A) where $\alpha'_i = \alpha_i$ if $\alpha_i \in \Sigma \cup \{\tau\}$ and $\alpha'_i = \tau$ if $\alpha_i \in \mathbb{R}^{>0}$. We also say that σ *is a timed instance* of $[\sigma]$ (written $\sigma \in [\sigma]$).

As a consequence of Lemma 4.13 in [3] we have the following result.

Lemma 2. *Given a TA A, if σ is an execution fragment of A, then $[\sigma]$ is an execution fragment of untime*(A). *Vice versa, if $[\sigma]$ is an execution fragment of untime*(A), *then there exists $\sigma' \in [\sigma]$ such that σ' is an execution fragment of A.*

The following conservativeness result holds.

Lemma 3. *Given TAs A and A', $A \approx A' \Rightarrow untime(A) \approx untime(A')$.*

Proof. The implication holds by the construction of the region automaton *untime*(A) and by Lemma 2. Actually, for each sequence of steps of a TA, there exists an analogous sequence of the LTS obtained with *untime*(A) and vice versa. Weak bisimulation is, therefore, preserved. □

3. Probabilistic Timed Automata

The framework of Probabilistic Timed Automata (PTAs) allows the description of timed systems showing a probabilistic behaviour, in an intuitive and succinct way. Therefore, within the framework of PTAs, where time and probabilities are taken into consideration, the modeler can describe, on a single model, different aspects of a system, and analyze real-time properties, performance and reliability properties.

Our definition of PTAs is inspired by the definitions in [11,35,2]. Intuitively, our definition of PTA derives from the idea of putting a PA (obtained by adding probabilities to an LTS) into a timed context (which adds temporal constraints to transitions and states, thus guarding some of the possible steps). Note, again, that translations can be given from our model of Probabilistic Timed Automata to the others, and vice versa.

As for PAs, we give a definition of Probabilistic Timed Automata where probability distributions are defined over the set of transitions. This choice, while requiring some particular attention in the definition of the semantics, allows us to easily get rid of probabilities when reducing to the non probabilistic case. Such a definition requires, in particular, the re-normalization of probabilities when time guards prevent the execution of some transitions. In a sense, time becomes a resource the PTA might consume and adds new constraints on the possible execution of the automaton.

Definition 12. A *Probabilistic Timed Automaton* (PTA) is a tuple $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$, where:

- Σ is a finite alphabet of actions;
- X is a finite set of nonnegative real variables called clocks;
- Q is a finite set of states and $q_0 \in Q$ is the initial state;
- $Inv : Q \to \Phi(X)$ is a function assigning a constraint $\phi \in \Phi(X)$ (called *state invariant*) to each state in Q;
- δ is a finite set of transitions in $Q \times \Sigma \times \Phi(X) \times 2^X \times Q$;
- $\Pi = \{\pi_1, \ldots, \pi_n\}$ is a finite set of probability distributions as functions $\pi_i : \delta \to [0, 1]$, for any $i = 1, \ldots, n$, where $\pi_i(e)$ is the probability of performing transition e according to distribution π_i.

For a state $q \in Q$, we denote with $\delta(q)$ the set of transitions with q as source state, i.e. the set $\delta(q) = \{(q', a, \phi, B, q'') \in \delta \mid q' = q\}$. We require that $\sum_{e \in \delta(q)} \pi_i(e) = 1$ for any i and q. Moreover, we assume that for all e_j there exists some π_i such that $\pi_i(e_j) > 0$.

3.1. Semantics of Probabilistic Timed Automata

Configurations, steps, execution fragments, and executions of a PTA are defined as for TAs.

Given a configuration $s = (q, v)$, with $Adm(s) = \{(q, a, \phi, B, q') \in \delta \mid v \models \phi \text{ and } v[B] \models Inv(q')\}$ we represent the set of admissible transitions that an automaton could execute from configuration s, and we say that a transition in $Adm(s)$ is *enabled* in s. Given two configurations $s = (q, v)$, $s' = (q', v')$, and given $a \in \Sigma \cup \{\tau\}$, we represent with $Adm(s, a, s') = \{(q, a, \phi, B, q') \in \delta \mid v \models \phi \wedge v' = v[B] \models Inv(q')\}$ the set of transitions that lead from configuration s to configuration s' through a transition step labeled with a. A configuration $s = (q, v)$ is called *terminal* iff $Adm(s') = \emptyset$ for all $s' = (q, v + t)$ with $t \in \mathbb{R}^{\geq 0}$; we denote with S_T the set of terminal configurations.

The probability of executing a transition step from a configuration s is chosen, among all the transitions enabled in s, according to the values returned by some distribution π, while we set the probability of executing a time step labeled with $t \in \mathbb{R}^{>0}$ to the value 1. Intuitively, a PTA chooses nondeterministically the distribution of a transition step or to let time elapse by performing a time step, and, in this case, also the amount of time passed is chosen nondeterministically.

Executions and execution fragments of a PTA arise by resolving both the nondeterministic and the probabilistic choices [35]. To resolve the nondeterministic choices of a PTA, we introduce now *schedulers* of PTAs.

A *scheduler* of a PTA A is a partial function from $ExecFrag_A$ to $\Pi \cup \mathbb{R}^{>0}$. Given a scheduler F and an execution fragment σ, we assume that F is defined for σ if and only if $\exists s \in \mathcal{S}_A$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$ such that $last(\sigma) \xrightarrow{\alpha} s$.

For a scheduler F of a PTA A we define $ExecFrag_A^F$ (resp. $Exec_A^F$) as the set of execution fragments (resp. the set of executions) $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots$ of A such that, for any $i, \alpha_i \in \mathbb{R}^{>0}$ iff $F(\sigma^{i-1}) = \alpha_i$, and $\alpha_i \in (\Sigma \cup \{\tau\})$ iff $\exists e \in Adm(s_{i-1}, \alpha_i, s_i)$ and $\pi_i(e) > 0$ where $\pi_i = F(\sigma^{i-1})$.

Given a scheduler F and an execution fragment $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_k} s_k \in ExecFrag_A^F$, if $k = 0$ we put $P_A^F(\sigma) = 1$, else, if $k \geq 1$, we define $P_A^F(\sigma) = P_A^F(\sigma^{k-1}) \cdot p$ where

$$p = \begin{cases} \dfrac{\sum_{e \in Adm(s_{k-1}, \alpha_k, s_k)} (F(\sigma^{k-1}))(e)}{\sum_{e \in Adm(s_{k-1})} (F(\sigma^{k-1}))(e)} & \text{if } \alpha_k \in \Sigma \cup \{\tau\} \\ 1 & \text{if } \alpha_k \in \mathbb{R}^{>0}. \end{cases}$$

Notice that such a measure is consistent, since we are assuming that, given the execution fragment σ, there is a step from s_{k-1} to s_k labeled with α_k. Now, if $\alpha_k \in \mathbb{R}^{>0}$, then $p = 1$, otherwise the probability of going from s_{k-1} to s_k through a discrete transition labeled with α_k is re-normalized according to the transitions enabled in s_{k-1}. In this latter case, $Adm(s_{k-1}) \neq \emptyset$, since there exists at least the step $s_{k-1} \xrightarrow{\alpha_k} s_k$.

Given a scheduler F, let $Exec_A^F(s)$ be the set of executions in $Exec_A^F$ starting in s, $ExecFrag_A^F(s)$ be the set of execution fragments in $ExecFrag_A^F$ starting in s, and $\sigma Field_A^F(s)$ be the smallest sigma field on $Exec_A^F(s)$ containing the basic cylinders $\sigma\uparrow$, where $\sigma \in ExecFrag_A^F(s)$. The probability measure $Prob_A^F$ is the unique measure on $\sigma Field_A^F(s)$ such that $Prob_A^F(\sigma\uparrow) = P_A^F(\sigma)$.

Remark. Note that, given a PTA A and a scheduler F, the executions of A *driven* by F do not contain any nondeterministic choice. Hence, a PTA A driven by a scheduler F gives rise to a fully probabilistic behaviour.

Example 11. In Fig. 8 we show an example of a PTA with $\Pi = \{\pi\}$. Intuitively, from the initial configuration $(q_0, \mathbf{0})$, the PTA may nondeterministically choose whether to perform some time step and update the value of clock x or to perform, probabilistically, transitions e_1, e_2 or e_3 with probabilities $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{1}{2}$, respectively.

If some time step is performed in state q_0, such that the value of clock x becomes greater than 5, then the transition labeled with b cannot be performed anymore, and the probabilities of performing the other transitions should be redistributed. In this case, even if the transition was at some point enabled in state q_0, we might intuitively say that the automaton has consumed too much time resources to be able to perform transition e_1 anymore. Even if the case is quite simple in the depicted automaton, similar situations may arise whenever the automaton returns to a certain state at different times and some of the transitions may not be enabled anymore. Note that re-normalizing probability at run-time allows us to relax the condition of *admissible target states* used in [35].

For appropriately chosen F and F', examples of executions of the PTA in Fig. 8 are $\sigma_1 = (q_0, 0) \xrightarrow{9.7} (q_0, 9.7) \xrightarrow{a} (q_2, 0) \xrightarrow{3} (q_2, 3)$ and $\sigma_2 = (q_0, 0) \xrightarrow{3} (q_0, 3) \xrightarrow{b} (q_1, 3) \xrightarrow{1.2} (q_1, 4.2)$ with $P^F(\sigma_1) = \frac{2}{5}$ and $P^{F'}(\sigma_2) = \frac{1}{6}$, where (q, t) represents the configuration composed by the state q and the valuation v such that $v(x) = t$. Please notice the difference between F, which chooses to wait 9.7 time units before taking a discrete transition, and F', which chooses to wait only 3 time units.

In the following, A is a PTA, F is a scheduler of A, $\hat{\alpha}$ stands for α if $\alpha \in \Sigma \cup \mathbb{R}^{>0}$ and for ε (the empty string) if $\alpha = \tau$, $s \in \mathcal{S}_A$ and $\mathcal{C} \subseteq \mathcal{S}_A$.

Let $Exec_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ be the set of executions that lead to a configuration in $\mathcal{C}$ via a sequence belonging to the set of sequences $\tau^*\hat{\alpha}\tau^*$ starting from the configuration s and crossing configurations equivalent (according to our bisimulation relation) to s. Finally, we define the probability $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^F(Exec_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}))$.

$e_1 = (q_0, b, 0 \leq x \leq 5, \emptyset, q_1)$ $\quad \pi(e_1) = \frac{1}{6}$
$e_2 = (q_0, a, true, \{x\}, q_2)$ $\quad \pi(e_2) = \frac{1}{3}$
$e_3 = (q_0, a, true, \emptyset, q_3)$ $\quad \pi(e_3) = \frac{1}{2}$

$\sum_{e \in \delta(q_0)} \pi(e) = 1$

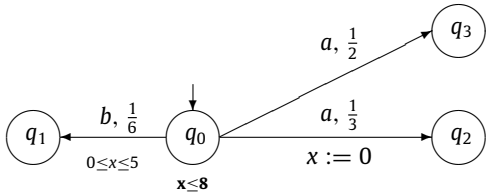

Fig. 8. Example of PTA.

3.2. Regions of PTAs

In [35] it as been shown that for forward symbolic reachability, the computation of probability could be an overapproximation due to the time successor operator. To solve this problem, we have split transition steps from time steps in the region graph. Moreover, we have defined a special function *Clean* that considers a time predecessor for a set of regions instead of a single region. Hence, for the timed regions of PTAs, we have exactly the same concepts and properties given for regions of TAs in Section 2.3.

3.3. Behavioural equivalence

We introduce a notion of weak bisimulation for PTAs.

As already mentioned, weak internal transitions $s \stackrel{\tau}{\Longrightarrow} s'$ are replaced by the probability $Prob(s, \tau^*, s')$ of reaching configuration s' from s via internal moves. Similarly, for visible actions α, transitions $s \stackrel{\alpha}{\Longrightarrow} s'$ are replaced by the probability $Prob(s, \tau^*\alpha\tau^*, s')$. The next definition is obtained by adapting Definition 5 to the model of PTAs.

Definition 13. Let $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$ be a PTA. A *weak bisimulation* on A is an equivalence relation $\mathcal{R}$ on $\mathcal{S}_A$ such that, for all schedulers F and $(s, s') \in \mathcal{R}$, there exists a scheduler F' such that for all $\mathcal{C} \in \mathcal{S}_A/\mathcal{R}$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$:

$$Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C})$$

and vice versa.

Two configurations s, s' are called *weakly bisimilar* on A (denoted $s \approx_A s'$) iff $(s, s') \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$.

Again, in order to define weak bisimulation among PTAs, we resort to the notion of a disjoint sum of automata.

Definition 14. Let $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$ and $A' = (\Sigma', X', Q', q_0', Inv', \delta', \Pi')$ such that $Q \cap Q' = \emptyset$ and $X \cap X' = \emptyset$. Let

$$\hat{A} = (\Sigma \cup \Sigma', X \cup X', Q \cup Q' \cup \{\hat{q}\}, \hat{q}, \hat{Inv}, \delta \cup \delta' \cup \{(\hat{q}, \tau, true, \emptyset, q_0), (\hat{q}, \tau, true, \emptyset, q_0')\}, \hat{\Pi}),$$

where $\hat{\pi_1}, \hat{\pi_2} \in \hat{\Pi}$ such that $\hat{\pi_1}(e) = 1$ if $e = (\hat{q}, \tau, true, \emptyset, q_0)$, 0 otherwise, and $\hat{\pi_2}(e) = 1$ if $e = (\hat{q}, \tau, true, \emptyset, q_0')$, 0 otherwise; moreover, for each couple $(\pi, \pi') \in \Pi \times \Pi'$, $\hat{\pi} \in \hat{\Pi}$ such that:

$$\hat{\pi}(e) = \begin{cases} \pi(e) & \text{if } e \in \delta \\ \pi'(e) & \text{if } e \in \delta'. \end{cases}$$

The invariant conditions of $\hat{A}$ are given by:

$$\hat{Inv}(q) = \begin{cases} true & \text{if } q = \hat{q} \\ Inv(q) & \text{if } q \in Q \\ Inv'(q) & \text{if } q \in Q'. \end{cases}$$

We say that A and A' are weakly bisimilar (denoted by $A \approx A'$) if $(q_0, \mathbf{0}) \approx_{\hat{A}} (q_0', \mathbf{0})$, where the valuation $\mathbf{0}$ is defined over all clocks of the set $X \cup X'$.

Example 12. Consider the PTAs of Fig. 9. Intuitively, they both can perform action a or action b before 5 time units, with equal probability $\frac{1}{2}$. By applying the notion of weak bisimulation introduced above, the two PTAs turn out to be equivalent. Please note that the probabilities in A_2 are re-normalized according to the transitions enabled in the different configurations. Let $\hat{A}$ be the automaton built from the automata A_1 and A_2 by following the procedure described in Definition 14.

We call π_1 the only probability distribution of A_1 and π_2 the only probability distribution of A_2.

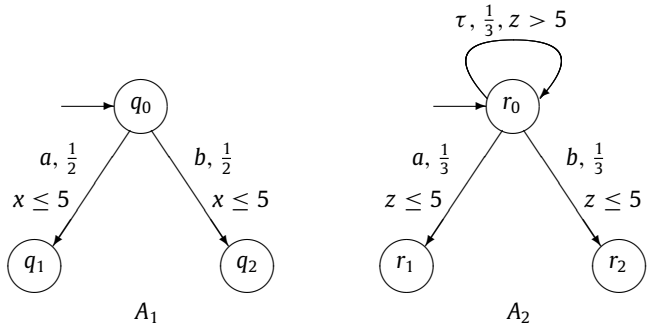

Fig. 9. $A_1 \approx A_2$.

With $\mathcal{R}$ we denote the equivalence relation on $\mathcal{S}_{\hat{A}}$ such that $((q, v), (r, v')) \in \mathcal{R}$ if one of the following requirements holds:

- $q = q_0, r = r_0$ and $0 \leq v(x) = v'(z) \leq 5$.
- $q = q_0, r = r_0$ and $v(x) = v'(z) > 5$.
- $q \in \{q_1, q_2\}$ and $r \in \{r_1, r_2\}$ and $v(x) = v'(z)$.

Note that the case $x \neq z$ is not considered since no reachable configuration allows this case.

Moreover, the set of classes has infinite cardinality. Actually, each value in $[0, 5]$ generates a class. Hence, for any $m \in [0, 5]$, we call $\mathcal{C}_m$ the class composed by the two configurations $\{(q_0, x = m), (r_0, z = m)\}$. As we will see, to solve the problem of the infiniteness of classes, the algorithm we propose groups the set of $\{\mathcal{C}_m\}_{m\in[0,5]}$ in the triple $(q_0, r_0, x \leq 5 \wedge x = z)$.

With $\mathcal{C}$ we denote the set of classes containing each configuration (q_0, v) and (r_0, v) such that $v(x) = v(z) > 5$. This set of classes can be represented by the triple $(q_0, r_0, x = z \wedge x > 5)$. Finally, with $\mathcal{C}'$ we denote the class containing each configuration (q, v) such that $q \in \{q_1, q_2\}$ and (r, v) such that $r \in \{r_1, r_2\}$ and $v(x) = v(z)$.

In the following we assume $\alpha \in \{a, b, \tau\} \cup \mathbb{R}^{>0}$, where α is chosen according to a scheduler F and a distribution π_i.

We consider the case in which the configuration is in a state of A_2, the other case is easier since from q_0 there is not any τ transition.

Given $s = (r_0, v) \in \mathcal{C}_m$ and $s' = (q_0, v') \in \mathcal{C}_m$ and a scheduler F. We have the following cases:

- If $Prob^F(s, \tau^*, \mathcal{C}_m) = 1$, then $Prob^{F'}(s', \tau^*, \mathcal{C}_m) = 1$, for any scheduler F'. Namely, $\epsilon \in \tau^*$.
- If $Prob^F(s, \tau^*\alpha\tau^*, \mathcal{C}_{m+\alpha}) = 1$ where $\alpha \in \mathbb{R}^{>0}$ and $\alpha \leq 5 - m$, then it is possible to choose F' such that $Prob^{F'}(s', \tau^*\alpha\tau^*, \mathcal{C}_{m+\alpha}) = 1$ where $F'(s') = \alpha$.
- If $Prob^F(s, \tau^*\alpha\tau^*, \mathcal{C}) = 1$ where $\alpha \in \mathbb{R}^{>0}$ and $\alpha > 5 - m$, then it is possible to choose F' such that $Prob^{F'}(s', \tau^*\alpha\tau^*, \mathcal{C}) = 1$ where $F'(s') = \alpha$.
- If $Prob^F(s, \tau^*\alpha\tau^*, \mathcal{C}') = \frac{1}{2}$ where $\alpha \in \{a, b\}$, then it is possible to choose F' such that $Prob^{F'}(s', \tau^*\alpha\tau^*, \mathcal{C}') = \frac{1}{2}$ where $F'(s') = \pi_1$.

Moreover, for each $s \in \mathcal{C}$ we have that if $Prob^F(s, \tau^*\alpha\tau^*, \mathcal{C}) = 1$ where $\alpha \in \mathbb{R}^{>0}$, then $Prob^{F'}(s', \tau^*\alpha\tau^*, \mathcal{C}) = 1$ where $F'(s') = \alpha$. Similarly for each $s \in \mathcal{C}'$.

The probability of any other case we have not considered here is 0 for any scheduler F. In this case, in order to show the weak bisimulation, it has been sufficient to consider $F' = F$.

Now, $\mathcal{R}$ is a weak bisimulation on $\hat{A}$ and, since $(q_0, \mathbf{0})$ and $(r_0, \mathbf{0})$ are in the same class, A_1 and A_2 are weakly bisimilar.

The next example shows a subtle feature of weak bisimulation for PTAs. Namely, internal actions, even if not visible, may alter the probability of observing the passage of time.

Example 13. Consider the PTAs of Fig. 10. Intuitively, both of them, eventually, perform action l with probability 1 and then reach a terminal configuration. At a first glance the two automata appear to be bisimilar, however, the internal move in A_2 allows a scheduler to make different the probability of observing passage of time with respect to A_1. Namely, A_1 and A_2 are not bisimilar because there exists no scheduler F of A_1 able to simulate the behaviour induced by the following scheduler F':

$$
\begin{array}{ll}
F'(\sigma_0) = \pi & \sigma_0 = (r_0, 0) \\
F'(\sigma_1) = 1 & \sigma_1 = (r_0, 0) \xrightarrow{\tau} (r_0, 0) \\
F'(\sigma_2) = \pi & \sigma_2 = (r_0, 0) \xrightarrow{\tau} (r_0, 0) \xrightarrow{1} (r_0, 1) \\
\ldots &
\end{array}
$$

where π is the only probability distribution of A_2 depicted in Fig. 10.

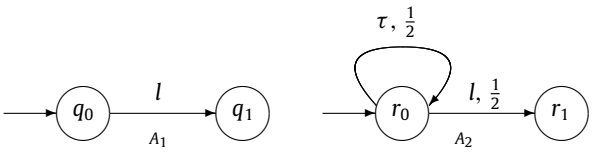

Fig. 10. $A_1 \not\approx A_2$.

If $\mathcal{C}$ is the class containing the configuration $(r_0, 1)$ we have that $Prob^{F'}_{A_2}((r_0, 0), \tau^*1\tau^*, \mathcal{C}) = \frac{1}{2}$, and no scheduler exists for A_1 with the same property.

If we consider this aspect from a security analysis point of view (in the context of noninterference), the action τ in A_2 could represent an *invisible* high level action h one wants to hide from a low level observer. In such a probabilistic timed system, however, as shown above, a scheduler may exploit the probabilistic execution of the action h (τ) to alter the observation of the passage of time. In this case, the probabilistic and timed features of the system could give rise to a probabilistic/timed covert channel (an high level activity affects the observable behaviour of the system): since action h (τ) competes probabilistically with action l, the low level observation of the passage of time could be probabilistically different in the two automata.

If we replace the probabilistic distribution π in A_2 with two distributions π_1 and π_2 assigning, respectively, probability 1 to each of the two transitions, the two PTAs turn out to be weakly bisimilar. In such a case, in fact, the internal action in A_2 is not any more in competition with the observable action l.

3.4. Removing probability and time from PTAs

Given a PTA A, we call *unprob*(A) the TA obtained from A by simply removing probabilities from A. This can be done since we assumed that for each transition of A there is at least a probability distribution which assigns a probability greater than 0 to such a transition (see Definition 12).

Definition 15. Given a PTA $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$, $unprob(A) = (\Sigma, X, Q, q_0, Inv, \delta)$.

Example 14. Let A be the PTA in Fig. 8. If we remove probabilities from A the TA *unprob*(A) can be found in Fig. 5.

Given a PTA A, we call *untime*(A) the PA obtained as the *region automaton* of A, with probability functions chosen according to Π. Intuitively, the region automaton is obtained by considering timed regions as states. Note that in the region automaton there might be a step between regions R and R' with symbol a also if there is an admissible run $s \xrightarrow{t} s'' \xrightarrow{a} s'$ of the PTA such that $t \in \mathbb{R}^{>0}$ and where $s \in R$ and $s' \in R'$. Since time steps are no more visible in the untimed setting, we use the silent action τ to label all the transitions of the PA *untime*(A) arising from time steps of the PTA A.

Definition 16. Given a PTA $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$, we define the PA $untime(A) = (\Sigma, Q \times \mathcal{V}, (q_0, [\mathbf{0}]), \delta', \Pi')$ where $\mathcal{V}$ is the set of equivalence classes of the valuations of A:

- $e = ((q, [v]), \tau, (q, [v'])) \in \delta'$ iff $v' = v + t$ for some time $t \in \mathbb{R}^{>0}$; moreover, there exists $\pi_e \in \Pi'$ such that $\pi_e(e) = 1$;
- $e = ((q, [v]), a, (q', [v'])) \in \delta'$ iff $(q, a, \phi, B, q') \in \delta$, $v \models \phi$ and $v' = v[B]$; moreover, for all $\pi_l \in \Pi$, $\pi'_l \in \Pi'$ such that $\pi'_l(e) = \frac{\sum_{e_i \in Adm((q,v),a,(q',v'))} \pi_l(e_i)}{\sum_{e_j \in Adm((q,v))} \pi_l(e_j)}$ if $\sum_{e_j \in Adm((q,v))} \pi_l(e_j) \neq \emptyset$ and $\pi'_l(e) = 0$ otherwise.

Example 15. In Fig. 11 we show the PTA A and its untimed version, the PA $untime(A) = (\Sigma', Q', u_0, \delta', \Pi')$. States u_0, u_1 and u_2 correspond, respectively, to the pairs $(q_0, [v_0])$, $(q_1, [v_0])$ and $(q_2, [v_0])$, where $[v_0] = \{v \mid v(x) = 0\}$. States u'_0, u'_1 and u'_2 correspond, respectively, to the pairs $(q_0, [v_1])$, $(q_1, [v_1])$ and $(q_2, [v_1])$, where $[v_1] = \{v \mid v(x) > 0\}$. Again, we omitted the self-loop transitions labeled with τ that arise from time steps of the PTA that do not change the region.

Note that, since in state u'_0 it holds $x > 0$, the transition labeled with a from q_0 to q_2 cannot be executed (it has constraint $x = 0$). Such a transition is lost in the PA *untime*(A) and probabilities are redistributed (actually transition with label b gets probability 1). Namely, there is a distribution π' in Π' (obtained by re-normalizing the only probability distribution of the PTA A) such that $\pi'((u_0, b, u_1)) = \frac{2}{3}$, $\pi'((u_0, a, u_2)) = \frac{1}{3}$ and $\pi'((u'_0, b, u'_1)) = 1$, while for any other transition e (corresponding to a time step in A), there is a distribution $\pi_e \in \Pi$ such that $\pi_e(e) = 1$.

Given an execution $\sigma = (q_0, v_0) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} (q_n, v_n)$ of A, with $[\sigma]$ we denote the corresponding execution $(q_0, [v_0]) \xrightarrow{\alpha'_1} \cdots \xrightarrow{\alpha'_n} (q_n, [v_n])$ of *untime*(A) where $\alpha'_i = \alpha_i$ if $\alpha_i \in \Sigma \cup \{\tau\}$ and $\alpha'_i = \tau$ if $\alpha_i \in \mathbb{R}^{>0}$. We also say that σ *is a timed instance* of $[\sigma]$ (written $\sigma \in [\sigma]$).

As a consequence of Lemma 4.13 in [3] and Lemma 4.8 in [37], we have the following result.

Lemma 4. *Given a PTA A, we have that, for any scheduler F of A, there exists a scheduler F′ of untime(A) such that, for any* $\sigma \in ExecFrag^F_A$, $Prob^F_A(\sigma) = Prob^{F'}_{untime(A)}([\sigma])$. *Vice versa, for any scheduler F of untime(A), there exists a scheduler F′ of A such that, for any* $[\sigma] \in ExecFrag^F_{untime(A)}$, $Prob^F_{untime(A)}([\sigma]) = Prob^{F'}_A(\sigma')$ *for some* $\sigma' \in [\sigma]$.

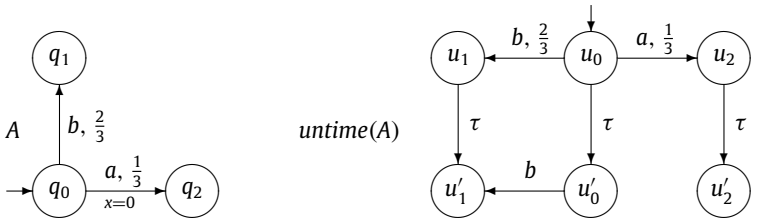

Fig. 11. Example of *untime*(*A*).

The following proposition states that given a PTA, we may obtain a LTS by removing time and probability in two successive steps, no matter about the order.

Proposition 6. *Given a PTA A, unprob*(*untime*(*A*)) = *untime*(*unprob*(*A*)).

Proof. The proof derives trivially from the construction of the region automaton in the *untime* operators for PTAs and TAs. Actually, by definition of the *untime* operators, the sets of transitions of *untime*(*A*) and *unprob*(*untime*(*A*)) are exactly the same. Now, the LTS obtained from a PTA does not change if we remove probabilities through the *unprob* operator either before or after applying the *untime* construction. Consistency holds since we assumed that for each transition of *A* there is at least a probability distribution which assigns to such a transition a probability greater than 0 (see Definition 12). □

The following conservativeness result holds.

Lemma 5. *Given PTAs A and A′ such that A* ≈ *A′, the following statements hold:*

1. *unprob*(*A*) ≈ *unprob*(*A′*);
2. *untime*(*A*) ≈ *untime*(*A′*).

Proof. For case 1, let us assume $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$, $A' = (\Sigma', X', Q', q'_0, Inv', \delta', \Pi')$ and $\hat{A}$ constructed as in Definition 14. Since $A \approx A'$ for a weak bisimulation $\mathcal{R}$, we have that for all schedulers F and $(s, r) \in \mathcal{R}$, there exists a scheduler F' such that for all $\mathcal{C} \in \mathcal{S}_{\hat{A}}/\mathcal{R}$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$, $Prob^F_{\hat{A}}(s, \tau^*\alpha\tau^*, \mathcal{C}) = Prob^{F'}_{\hat{A}}(r, \tau^*\alpha\tau^*, \mathcal{C})$. Now, if $Prob^F_{\hat{A}}(s, \alpha, s') > 0$ for some $s' \in \mathcal{C}$ there exists a configuration r' and a scheduler F' such that $Prob^{F'}_{\hat{A}}(r, \tau^*\alpha\tau^*, r') = Prob^F_{\hat{A}}(s, \alpha, s') > 0$. Therefore if $s \xrightarrow{\alpha} s'$, then there exists r' such that $r \stackrel{\alpha}{\Longrightarrow} r'$ and, since s' and r' are in the same equivalence class, $\mathcal{R}$ is also a weak bisimulation on $\mathcal{S}_{\hat{A}_{np}}$, where $\hat{A}_{np}$ is the TA constructed as in Definition 10 starting from *unprob*(*A*) and *unprob*(*A′*). The same holds if we exchange the roles of *s* and *r*.

For case 2, the implication holds by the construction of the region automaton and by Lemmata 4 and 2. Actually, for each run of a PTA (or TA), there exists an analogous run for the PA (or LTS) obtained with *untime*(*A*), and vice versa. Weak bisimulations are, therefore, preserved. □

4. Decidability of weak bisimulation for PTAs

In this section we develop an algorithm which computes the classes of the weak bisimulation equivalence and decides whether two configurations are weakly bisimilar by checking that they are in the same class. To do this, we have to check the condition of Definition 14.

In particular, we resort to disjoint sets of clocks in order to describe pairs of configurations $((q, v), (q', v'))$ within a certain equivalence relation. We use X to represent the evaluations of the clocks in X for configuration (q, v), and $\overline{X}$ to represent the evaluations for configuration (q', v') where $\overline{X} = \{\overline{x} \mid x \in X\}$.

Example 16. Consider the PTAs in Fig. 12. In the untimed version, the probability of reaching q_4 from the state q_2 is $\frac{1}{2}$. Now, in the timed version, we observe that in state q_2, when clock x has value smaller than 3, the automaton may execute both transitions with probability $\frac{1}{2}$. Otherwise, if clock x has value greater than 3, the transition labeled with a cannot be executed, and hence the probability has to be redistributed; in such a case the probability of executing the transition with action a is 0, whereas the transition labeled with b gets probability 1. Therefore, we need to consider the different cases in which a subset of transitions are enabled or not.

Moreover, we might consider using the algorithm for the untimed version on the region graphs of the two automata, i.e. the graph of regions resulting by applying the *untime* operator. However, this is not a good solution; in fact, if we consider the clock zone reached in state q_1 we have $x \geq 0$ and in state q_6 we have $x \geq 0$. Let us suppose that one wants to compare the probability of reaching q_2 from q_1 with the probability of reaching q_7 from q_6. Now, we must check the two probabilities for each time $\alpha \in \mathbb{R}^{\geq 0}$, and these are equal for every time if and only if the value of x in q_1 is equal to the value of x in q_6. This means that we cannot consider the clocks separately, but we must have formulae on all the pairs of states.

Hence, we have to consider formulae that express conditions on the value of clocks at state q_1 together with the value of clocks at state q_6. As an example the triple $(q_1, q_6, x = \overline{x})$ means that the value of clock x at state q_1 is equal to the value of clock x at state q_6. Thus, a set of bisimilar configurations (called *class*) can be expressed by a set of triples.

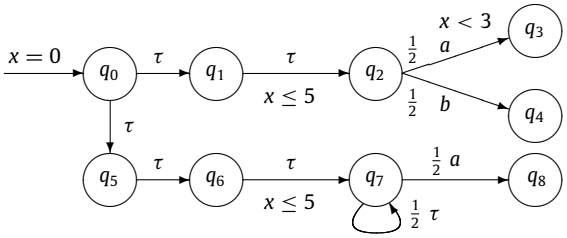

Fig. 12. An example on our methodology.

For deciding our notion of weak bisimulation, we follow the classical approach of refining relations between configurations [48,33,7,15]. In particular, starting from the initial relation where all the configurations of a PTA are equivalent, we stepwise specialize relations until we obtain a weak bisimulation.

At each step we refine the set of classes by deleting the relations between configurations s^1 and s^2 that do not satisfy the condition that, for all schedulers F, there exists a scheduler F' such that $Prob_A^F(s^1, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s^2, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ and vice versa.

To compute the probabilities $Prob_A^F(s^1, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ and $Prob_A^{F'}(s^2, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$, with $\mathcal{C}$ belonging to the current partition, we construct two PAs A^1 and A^2. PA A^i has triples (q, q', ψ) as states (where ψ is a formula on $\Psi(X)$), while transitions (computed by means of predecessor operators) reflect the possibility and the probability of performing certain steps from configurations reached starting from s^i, for $i = 1, 2$.

When $\alpha \in \mathbb{R}^{>0}$ labels a time step, we require that the unsatisfiability of bisimulation requirements is not caused by the fact that a time step α cannot be performed. Actually, even if a time step α can be performed from s^1 but not from s^2, both A^1 and A^2 do not have a transition representing that step. This is because, since states of A^1 and A^2 are triples, steps from s^1 are affected by configuration s^2, and hence there is no time successor from the triple representing s^1 and s^2. This does not hold for transition steps since the guard of a transition triggered from s^1 is not affected by the valuation in s^2. We solve this problem by defining an algorithm *Clean* that refines the classes in such a manner that the unsatisfiability of bisimulation requirements is not caused by the fact that we cannot perform a step labeled with $\alpha \in \mathbb{R}^{>0}$. Intuitively, algorithm *Clean* removes the relations between configurations from which it is not possible to perform the same time step to reach a certain class of bisimilar configurations.

The correctness of the methodology we propose is set up on the following inductive definition of equivalence relations $\sim_n$ on $\mathcal{S}_A$.

Definition 17. Let $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$ be a PTA. We set $\sim_0 = \mathcal{S}_A \times \mathcal{S}_A$ and, for $n = 0, 1, \ldots, s \sim_{n+1} s'$ iff for all schedulers F there exists a scheduler F' such that $\forall \mathcal{C} \in \mathcal{S}_A/\sim_n$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$ it holds that $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ and vice versa.

The next lemma allows us to define an algorithm for computing weak bisimulation equivalence classes by following the technique discussed above.

Lemma 6. *Let $A = (\Sigma, X, Q, q_0, Inv, \delta, \Pi)$ be a PTA and $s, s' \in \mathcal{S}_A$. Then, $s \approx s' \Leftrightarrow \forall n \geq 0 \quad s \sim_n s'$.*

Proof. The proof can be adapted from the one in Baier's habilitation thesis [8] for probabilistic systems. This can be done since configurations of a PTA can be grouped in a finite set of classes (see Section 2.3).

Let $\sim' = \bigcap_{n\geq 0} \sim_n$. We have to show that $\approx = \sim'$. Since the approximations $\sim_n$ are equivalence relations, it is easy to see that $\sim'$ is an equivalence relation too. By induction on n we can show that $\sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \approx$. Hence, $\sim' \supseteq \approx$. In order to show that $\sim' \subseteq \approx$ we prove that $\sim'$ is a weak bisimulation.

Since $\sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \approx$, then, for any $n \geq 0$, $s \sim_{n+1} s'$ implies $s \sim_n s'$. Therefore, for any $n \geq 0$ and for any $B \in \mathcal{S}_A/\sim_n$, there exists $B' \in \mathcal{S}_A/\sim_{n+1}$, such that $B' \subseteq B \ldots$.

Now, for each $B \in \mathcal{S}_A/\sim'$ and for each $n \geq 0$, it holds that there exists a unique element $B_n \in \mathcal{S}_A/\sim_n$ with $B \subseteq B_n$. Actually $\sim' = \bigcap_{n\geq 0} \sim_n$.

Now, $B \subseteq B_n$ and $B \subseteq B_{n+1}$ implies that $B_n \cap B_{n+1} \neq \emptyset$.

But we have proved that for any $n \geq 0$ and for any $B \in \mathcal{S}_A/\sim_n$, there exists $B' \in \mathcal{S}_A/\sim_{n+1}$, such that $B' \subseteq B$, hence the class B_n contains the class B_{n+1}, for any n.

Then, $B_0 = \mathcal{S}_A \supseteq B_1 \supseteq B_2 \supseteq \cdots$ and $B = \bigcap_{n\geq 0} B_n$.

Claim 1: We want to prove that, for all schedulers F of A, if $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, B) > 0$ and $B \in \mathcal{S}_A/\sim'$, then $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, B) = \inf_{n\geq 0} Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, B_n)$. For short, let us call $P[B_n]$ the probability $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, B_n)$. Since $B = \bigcap_{n\geq 0} B_n$ and $B_n \supseteq B_{n+1}$, we have $1 = P[B_0] \geq P[B_1] \geq \cdots \geq P[B_n]$. We put $r = inf_{n\geq 0} P[B_n]$. Clearly $r \geq P[B]$. We suppose, by contradiction, that $r > P[B]$. Let $\Delta = r - P[B]$, then $\Delta > 0$. Since $\sum_{t\notin B} P[t]$ is convergent, then there exists a finite subset M of $\mathcal{S}_A \setminus B$ such that $P[N] < \Delta$ where $N = \mathcal{S}_A \setminus (B \cup M)$. For all $n \geq 0$, $B_n = B \cup (N \cap B_n) \cup (M \cap B_n)$. The sets B, $N \cap B_n$ and $M \cap B_n$ are pairwise disjoint. Hence, $P[B_n] = P[B] + P[N \cap B_n] + P[M \cap B_n] < P[B] + \Delta + P[M \cap B_n] = r + P[M \cap B_n]$. Since $r \leq P[B_n]$ we get $M \cap B_n \neq \emptyset$. Since $B_n \supseteq B_{n+1}$ and $M \cap B_n \neq \emptyset$, for any n, then there exists t such that $t \in B_n$, for any n. Hence, $t \in (\bigcap_{n\geq 0} B_n) = B$. But $t \in B$ and $t \in M$, implies $M \cap B \neq \emptyset$, giving a contradiction (by definition M is a subset of $S \setminus B$).

Claim 2: Now, we want to prove that $\sim'$ is a weak bisimulation. Let $s \sim' s'$ and $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) > 0$ for some scheduler F and $\mathcal{C} \subseteq \mathcal{S}_A$. By Claim 1 it suffices to show that there exists a scheduler F' such that $Prob_A^F(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$, for all $n \geq 1$ and $\mathcal{C} \in \mathcal{S}_A/\sim_n$. But this directly derives from the definition of $\sim_n$. In fact, since for all F and $n \geq 1$ $s \sim_{n+1} s'$, we have that there exists a scheduler F' such that $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ $\forall C \in \mathcal{S}_A/\sim_n$. □

In the remainder of this section we assume A to be the PTA $(\Sigma, X, Q, q_0, Inv, \delta, \Pi)$.

4.1. Classes

In this section we define the notion of *classes* of a PTA. A class represents a set of pairs of configurations that belong to the relation we are considering. Hence, a set of classes $\mathcal{G}$ defines a relation $\approx_{\mathcal{G}}$.

Definition 18. A *class* g of the PTA A is a finite set in $Q \times Q \times \Psi_{C_A}(X \cup \overline{X})$. A class g defines the relation $\approx_g \subseteq \mathcal{S}_{\mathcal{A}} \times \mathcal{S}_{\mathcal{A}}$ containing pairs $((q, v), (q', v'))$ such that there exists $(q, q', \psi) \in g$ with $v'' \models \psi$, where $v''(x) = v(x)$ and $v''(\overline{x}) = v'(x)$, for any $x \in X$.

Given a triple (q, q', ψ), we will refer to *the first component of* (q, q', ψ) when we refer to the configurations expressed by the region $(q, \exists\overline{X}.\psi)$, and we will refer to the *second component of* (q, q', ψ) when we refer to the configurations expressed by the region $(q', \exists X.\psi)$.

From now on, without loss of generality, we assume that for all $(q, q', \psi) \in g$, it holds that $\psi \not\equiv false$.

Given two classes g_1 and g_2, we define the class resulting from the intersection $g_1 \cap g_2$. Namely, $g_1 \cap g_2 = \{(q, q', \psi \wedge \psi') \mid (q, q', \psi) \in g_1$ and $(q, q', \psi') \in g_2$ and $\psi \wedge \psi' \not\equiv false\}$.

We call *atomic constraints*, the constraints of the form $x \sim c$ and $x - y \sim c$. Obviously, the negation of an atomic constraint is expressible as an atomic constraint. As example $\neg(x \leq 5)$ is equal to $x > 5$. With $\neg g$ we denote the class $\{(q, q', \neg\phi) \mid (q, q', \psi) \in g)$ s.t. $\psi \not\equiv true$ and ϕ is an atomic constraint appearing in $\psi\}$.

Moreover, with $Ap_A(g)$ we denote the set $\{(q, q', Ap_A(\psi)) \mid (q, q', \psi) \in g)\}$.

With $Set(A)$ we denote the set of sets of classes $\mathcal{G}$ of A such that, for any $g_1, g_2 \in \mathcal{G}$ with $g_1 \neq g_2$, it holds that, for any $(q, q', \psi) \in g_1$ and $(q, q', \psi') \in g_2$ it holds that $\psi \wedge \psi' \equiv false$. A set of classes $\mathcal{G}$ defines the relation $\approx_{\mathcal{G}}$ that is the relation $\bigcup_{g \in \mathcal{G}} \approx_g$.

Now, the number of classes is finite since the number of clock zones is finite (see Proposition 3). Moreover, the number of clocks is in the order of $|X|$. Hence, the number of triples (q, q', ψ) is in the order of the number of clock zones of A.

Example 17. The set $\mathcal{G} = \{(q_0, q_2, 0 \leq x \leq \overline{x})\}$ is in $Set(A)$, where A is the PTA of Example 16. The set $\mathcal{G}$ defines the relation $\approx_{\mathcal{G}}$ such that $(q, v) \approx_{\mathcal{G}} (q', v')$ iff $q = q_0, q' = q_2$ and $v(x) \leq v'(x)$.

4.2. The algorithm Clean

We introduce the algorithm $Clean^A(\mathcal{G})$ that eliminates the relations between configurations such that the unsatisfiability of bisimulation requirements is not caused by the fact that we cannot perform the time step α. Intuitively, we use the algorithm *Clean* in order to remove the relations between configurations from which it is not possible to perform the same time step to reach a certain class of bisimilar configurations.

We denote with ψ_g the formula

$$\forall t' \in [0, t], \quad \bigwedge_{i \in [1,2]} Inv(q_i)[X := X + t'] \wedge \bigwedge_{(q_1, q_2, \psi) \in g} \psi[X := X^{q_1} + t][\overline{X} := \overline{X}^{q_2} + t]$$

where t is a new variable representing the time elapsed, and x^{q_1} and $\overline{x}^{q_2}$ are new clocks, for any state q and clock $x \in X$ (namely, x and $\overline{x}$ are substituted in ψ with $x^{q_1} + t$ and $\overline{x}^{q_2} + t$, respectively).

Definition 19. Given a class g, with $prec_T(g)$ we denote the class

$$\bigcup_{(q_1, q_2, \psi) \in g} \{(q_1, q_2, \exists X^Q \cup \overline{X}^Q \cup \{t\}.t > 0 \wedge x = x^{q_1} \wedge \overline{x} = \overline{x}^{q_2} \wedge \psi_g)\}$$

where $X^Q = \{x^q \mid x \in X,\ q \in Q\}$ and $\overline{X}^Q = \{\overline{x}^q \mid x \in X,\ q \in Q\}$.

The class $prec_T(g)$ contains the configurations from which g can be reached with the same time step. We consider $t > 0$ since the scheduler returns time steps in $\mathbb{R}^{>0}$. Actually a time step with $t = 0$ changes neither the location nor the valuation. Hence, it can be trivially excluded.

The following lemma states that deleting the relations between configurations such that the unsatisfiability of the bisimulation requirements is not caused by the fact that one cannot perform the time step α, is equivalent to deleting the relations between configurations from which it is not possible to perform the same time step to reach a certain class of bisimilar configurations.

Lemma 7. *Given an equivalence relation $\approx$, the two following statements are equivalent:*

- *For any $s_1 \approx s_2$ and $\alpha \in \mathbb{R}^{>0}$, if $s_1 \xrightarrow{\alpha} s_1'$, then $s_2 \xrightarrow{\alpha} s_2'$ with $s_1' \approx s_2'$.*
- *For any $s_1 \approx s_2$, then, for all schedulers F and F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in S_A/\approx$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$.*

Proof. We prove the two implications:

1. For any $s_1 \approx s_2$ and $\alpha \in \mathbb{R}^{>0}$, if $s_1 \xrightarrow{\alpha} s_1'$, then $s_2 \xrightarrow{\alpha} s_2'$ with $s_1' \approx s_2'$ implies the property that for any $s_1 \approx s_2$, then, for all schedulers F and F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in S_A/\approx$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$.

   Let us suppose by contradiction that there exist two schedulers F and F', a time $\alpha \in \mathbb{R}^{>0}$ and a $\mathcal{C} \in S_A/\approx$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ and $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) = 0$ or vice versa.

   Let us consider the case $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ and $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) = 0$.

   Therefore there exists $\alpha \in \mathbb{R}^{>0}$ such that $s_1 \xrightarrow{\alpha} s_1'$ and, for any s_2' such that $s_1' \approx s_2'$, $s_2 \not\xrightarrow{\alpha} s_2'$. This is a contradiction since, for any $\alpha \in \mathbb{R}^{>0}$, $s_1 \xrightarrow{\alpha} s_1'$ iff $s_2 \xrightarrow{\alpha} s_2'$.

   The other case is analogous, since $\approx$ is an equivalence relation, and hence $s_1 \approx s_2$ implies $s_2 \approx s_1$.
2. For any $s_1 \approx s_2$, then, for all schedulers F and F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in S_A/\approx$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ implies the property that, for any $s_1 \approx s_2$ and $\alpha \in \mathbb{R}^{>0}$, if $s_1 \xrightarrow{\alpha} s_1'$, then $s_2 \xrightarrow{\alpha} s_2'$ with $s_1' \approx s_2'$.

   By contradiction there exists $\alpha \in \mathbb{R}^{>0}$ such that, for any $s_1' \approx s_2'$ it holds that $s_1 \xrightarrow{\alpha} s_1'$ and $s_2 \not\xrightarrow{\alpha} s_2'$.

   This means that there exists a scheduler F such that $Prob_A^F(s_1, \tau^*\alpha\tau^*, [s_1']) > 0$, while for all schedulers F' it holds that $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, [s_2']) = 0$. But this is a contradiction since $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ for $\mathcal{C} = [s_1'] = [s_2']$. □

We give now the algorithm $Clean^A(\mathcal{G})$ which refines $\mathcal{G}$ by using $prec_T$. By Lemma 7, $Clean^A(\mathcal{G})$ deletes the relations between configurations from which it is not possible to perform the same time step to reach a certain class of bisimilar configurations.

$$
\begin{array}{l}
Clean^A(\mathcal{G} : Set(A)) : Set(A) \\
\quad g' := \bigcap_{g \in \mathcal{G}} \neg g \\
\quad \mathcal{G} := \bigcup_{g \in \mathcal{G}} g \cap \neg \left(\bigcup_{(q,q',\psi) \in g'} prec_T(\{(q, q', \psi)\}) \right) \\
\quad \text{return } \mathcal{G}
\end{array}
$$

Class g' represents the set of pairs (s, s') such that $s \not\approx_{\mathcal{G}} s'$, $\left(\bigcup_{(q,q',\psi) \in g'} prec_T(\{(q, q', \psi)\}) \right)$ represents the set of configurations from which we can reach non bisimilar configurations through a time step. Therefore, $\neg\left(\bigcup_{(q,q',\psi) \in g'} prec_T(\{(q, q', \psi)\}) \right)$ represents the set of configurations from which, through any time step, bisimilar configurations are reached. As a consequence, we refine the relation $\approx_{\mathcal{G}}$ by deleting the pairs (s, s') such that it is not possible to perform the same time step from s and s' to reach a certain class.

The following lemma states the correctness of the algorithm $Clean^A$.

Lemma 8. *Relation $\approx_{Clean^A(\mathcal{G})}$ is the biggest relation enclosed in $\approx_{\mathcal{G}}$ such that if $s_1 \approx_{Clean^A(\mathcal{G})} s_2$ and there exists a scheduler F such that for any scheduler F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in \mathcal{S}_A/\approx_{\mathcal{G}}$, it holds that $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ and vice versa. Moreover, $Clean^A(\mathcal{G})$ is computable in exponential time w.r.t. the size of A and, if $\approx_{\mathcal{G}}$ is an equivalence relation, then $\approx_{Clean^A(\mathcal{G})}$ is an equivalence relation.*

Proof. Since we refine a triple (q, q', ψ) with a triple (q, q', ψ') such that $\psi' \Rightarrow \psi$, it is obvious that if $s \approx_{Clean^A(\mathcal{G})} s'$, then $s_1 \approx_{\mathcal{G}} s_2$. Therefore the relation $\approx_{Clean^A(\mathcal{G})}$ is enclosed in the relation $\approx_{\mathcal{G}}$.

We prove now that if $s_1 \approx_{Clean^A(\mathcal{G})} s_2$ and there exists a scheduler F such that for any scheduler F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in S_A/\approx_{Clean^A(\mathcal{G})}$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ and vice versa.

By Lemma 7, it is sufficient to prove that for any $s_1 \approx_{Clean^A(\mathcal{G})} s_2$ and $\alpha \in \mathbb{R}^{>0}$, if $s_1 \xrightarrow{\alpha} s_1'$, then $s_2 \xrightarrow{\alpha} s_2'$, for some $s_1' \approx_{Clean^A(\mathcal{G})} s_2'$.

Given $s_1 \approx_{Clean^A(\mathcal{G})} s_2$, let us suppose, by contradiction, that $s_1' \not\approx_{\mathcal{G}} s_2'$, for some s_1' and s_2' such that $s_i \xrightarrow{\alpha} s_i'$, for $i = 1, 2$.

Since $s_1' \not\approx_{\mathcal{G}} s_2'$, then $s_1' \not\approx_{g'} s_2'$ (recall that $g' = \bigcap_{g \in \mathcal{G}} \neg g$). Moreover, by Proposition 4 and since $\alpha \in \mathbb{R}^{>0}$, we have that $s_1 \not\approx_{g''} s_2$ where $g'' = \left(\bigcup_{(q,q',\psi) \in g'} prec_T(\{(q, q', \psi)\}) \right)$.

The equivalence $s_1 \approx_{g''} s_2$ implies that $s_1 \not\approx_{\neg g''} s_2$, and, since $Clean^A(\mathcal{G}) = \bigcup_{g \in Clean^A(\mathcal{G})} g \cap \neg g''$, we have that $s_1 \not\approx_{Clean^A(\mathcal{G})} s_2$, that is a contradiction.

Finally, $\approx_{Clean^A(\mathcal{G})}$ is the biggest relation, namely, we just have to prove that for each relation $\approx'$ satisfying the same requirements of $\approx_{Clean^A(\mathcal{G})}$ we have that $\approx'$ is contained in $\approx_{Clean^A(\mathcal{G})}$.

Let us suppose, by contradiction, that there exists such a relation $\approx'$ with $\approx' \supset \approx_{Clean^A(\mathcal{G})}$. As a consequence, there exist s_1 and s_1' such that $s_1 \approx' s_1'$ and $s_1 \not\approx_{Clean^A(\mathcal{G})} s_1'$.

Now, $s_1 \approx' s_1'$ implies $s_1 \approx_{\mathcal{G}} s_1'$, thus $s_1 \approx_{\mathcal{G}} s_1'$ and $s_1 \not\approx_{Clean^A(\mathcal{G})} s_1'$. This implies that $s_1 \not\approx_{Clean^A(\mathcal{G})} s_1'$ is introduced by the refinement done by $Clean^A$. Therefore, there is some g'' such that $s_1 \not\approx_{\neg g''} s_1$ where $g'' = \left(\bigcup_{(q,q',\psi) \in g'} prec_T(\{(q, q', \psi)\})\right)$ (recall that $g' = \bigcap_{g \in \mathcal{G}} \neg g$). Since $s_1 \not\approx_{\neg g''} s_1$ we have that $s_1 \approx_{g''} s_1$. Hence, by Proposition 4, there exists α, s_2 and s_2' such that $s_1 \xrightarrow{\alpha} s_2$ and $s_1' \xrightarrow{\alpha} s_2'$ and $s_2 \approx_{g'} s_2'$. As a consequence, since $g' = \bigcap_{g \in \mathcal{G}} \neg g$, we have that $s_2 \not\approx_{\mathcal{G}} s_2'$.

This implies that $s_1 \xrightarrow{\alpha} s_2$ and $s_1' \xrightarrow{\alpha} s_2'$ and $s_2 \not\approx_{\mathcal{G}} s_2'$. Since $\approx'$ satisfies the conditions of Lemma 7, it must hold that $s_1 \not\approx' s_1'$, but this is a contradiction since by hypothesis $s_1 \approx' s_1'$.

The cost of the operator $Clean^A(\mathcal{G})$ is the cost of assignments and $\cup$ and $\cap$ operators that are polynomial on the size of $\mathcal{G}$. The size of $\mathcal{G}$ is bounded on the size of the regions that are exponential on the size of A. Hence, $Clean^A(\mathcal{G})$ is computable in exponential time on the size on the size of A.

We prove that $\approx_{Clean^A(\mathcal{G})}$ is an equivalence relation. Obliviously, since $Clean^A(\mathcal{G})$ is a refinement of $\mathcal{G}$, for any $g \in \mathcal{G}$ there exists $g' \in Clean^A(\mathcal{G})$ such that $\approx_{g'} \subseteq \approx_g$. We call g' as $clean(g)$.

Hence the thesis holds since $s_1 \approx_{clean(g)} s_2$ implies $s_1 \approx_g s_2$ and, by Lemma 7, we have that, for any $s_1 \approx_{Clean^A(\mathcal{G})} s_2$, F and F', $\alpha \in \mathbb{R}^{>0}$ and $\mathcal{C} \in S_A/\approx_{Clean^A(\mathcal{G})}$, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$.

Actually:

- Reflexivity: we must prove that $s \approx_{Clean^A(\mathcal{G})} s$. Since $\approx_{\mathcal{G}}$ is an equivalence relation, we have that $s \approx_{\mathcal{G}} s$, hence there exists $g \in \mathcal{G}$ such that $s \approx_g s$. Hence it is sufficient to prove that $s \approx_{clean(g)} s$. Obviously, by Lemma 7, $Prob_A^F(s, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s, \tau^*\alpha\tau^*, \mathcal{C}) > 0$.
- Symmetry: it is sufficient to prove that, for any g, if $s_1 \approx_{clean(g)} s_2$, then $s_2 \approx_{clean(g)} s_1$. But obviously $s_1 \approx_g s_2$ and $s_2 \approx_g s_1$ and, by Lemma 7, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ implies that $Prob_A^F(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$.
- Transitivity: we must prove that, for any g, if $s_1 \approx_{clean(g)} s_2$ and $s_2 \approx_{clean(g)} s_3$, then $s_1 \approx_{clean(g)} s_3$. But obviously $s_1 \approx_g s_2$ and $s_2 \approx_g s_3$, then $s_1 \approx_g s_3$. Moreover, by Lemma 7, $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ and $Prob_A^{F'}(s_2, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F''}(s_3, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ implying that $Prob_A^F(s_1, \tau^*\alpha\tau^*, \mathcal{C}) > 0$ iff $Prob_A^{F''}(s_3, \tau^*\alpha\tau^*, \mathcal{C}) > 0$. □

Example 18. Consider the set of classes $\mathcal{G} = \{g_1, g_2\}$ of Example 16 where

$$g_1 = \{(q_2, q_7, 0 \le x < 3 \wedge 0 \le \bar{x} < 3)\}$$

and

$$g_2 = \{(q_2, q_7, x \ge 3 \wedge \bar{x} \ge 3)\}.$$

Since from q_2 and q_7 only time steps can be performed, we have that, by following Proposition 2, variable t can be deleted, and hence $\neg\left(\bigcup_{(q,q',\psi) \in g'} prec_T((q, q', \psi))\right) = \{(q_2, q_7, (x = \bar{x})), (q_2, q_7, (x \ge 3 \wedge \bar{x} \ge 3))\}$.

Therefore, we refine $\mathcal{G}$ getting the classes $\{(q_2, q_7, 0 \le x < 3 \wedge x = \bar{x})\}$ and $\{(q_2, q_7, x \ge 3 \wedge \bar{x} \ge 3)\}$ instead of g_1 and g_2, respectively. Actually, any time step does not give problems for the class g_2, but for the class g_1 one must have that $x = \bar{x}$ in order to reach bisimilar configurations. As an example, $(q_2, x = 2.5) \approx_{\mathcal{G}} (q_7, x = 2.9)$, but $(q_2, x = 2.5) \xrightarrow{0.2} (q_2, x = 2.7)$, $(q_7, x = 2.9) \xrightarrow{0.2} (q_7, x = 3.1)$ and $(q_2, x = 2.7) \not\approx_{\mathcal{G}} (q_7, x = 3.1)$.

4.3. The Cut operator

Now, following the partition/splitter technique (see [48,33]), we need a splitter operator (see also [7,15]). Hence, we define a cut operator that, given a class, splits it into a set of classes satisfying the requirements of the definition of weak bisimulation. For this purpose, we construct two PAs by using predecessor operators for computing the probabilities $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$.

For PTAs, the probabilities of each step strongly depend on the enabled transitions. The following proposition ensures that for any configuration s' reachable from a given configuration s by a discrete transition step, the values of the clocks in s' remain the same as in s or are equal to zero (through clock resets).

Proposition 7. *For each* $\sigma = (q_0, v_0) \xrightarrow{\alpha_1} (q_1, v_1) \cdots \xrightarrow{\alpha_k} (q_k, v_k)$, *with* $\alpha_i \in \Sigma \cup \{\tau\}$, *it holds that either* $v_i(x) = v_0(x)$ *or* $v_i(x) = 0$, *for any* $i \geq 0$ *and* $x \in X$.

Proof. Since there are no time steps, $v_i(x) \neq v_0(x)$ if and only if x is reset and $v_0(x) > 0$. □

Definition 20. Given a class g and a set of transitions E, with $prec(g, E)$ and $\overline{prec}(g, E)$ we denote, respectively, the classes

$$\bigcup_{(q,a,\phi,B,q') \in E} \{(q, q'', Inv(q) \wedge \phi \wedge \exists B.\psi \wedge B = 0 \wedge Inv(q')) \mid (q', q'', \psi) \in g\}$$

and

$$\bigcup_{(q,a,\phi,B,q') \in E} \{(q'', q, (Inv(q) \wedge \phi)[X := \overline{X}] \wedge (\exists \overline{B}.\psi \wedge \overline{B} = 0 \wedge (Inv(q')[X := \overline{X}]))) \mid (q'', q', \psi) \in g\}.$$

Moreover, $[g]$ is the set of triples (q, q', ψ) such that $\psi \in \Psi_{C_A}(X \cup \overline{X})$, $\psi \not\equiv false$ and there exists $(q, q', \psi') \in g$ with, for any $x \in X \cup \overline{X}$, either $\psi \Rightarrow (\psi' \wedge x = 0)$ or $\psi \Rightarrow (\psi' \wedge x > 0)$.

The formulae $prec(g, E)$ and $\overline{prec}(g, E)$ give the sets of configurations from which it is possible to reach, respectively, the first and the second component by performing a transition in E. The set $[g]$ is the set of triples enclosed in g such that each variable is either equal to 0 or greater than 0.

Since g expresses a set of configurations, for any configuration (q, v) and (q, v') such that there exists a finite path from (q, v) to (q, v') using symbols in $\Sigma \cup \{\tau\}$, we have that $Adm((q, v))$ can be different from $Adm((q, v'))$.

However, by Proposition 7, the set of transitions that can be taken from a configuration can be deterministically associated with a state, if it is known that $x > 0$ or $x = 0$, for each clock x. Actually, $Adm(q, v) \neq Adm(q, v')$ if and only if $v(x) \neq v'(x)$, for some x appearing in a condition of a transition in $Adm(q, v) \cup Adm(q, v')$. By Proposition 7, if $v(x) \neq v'(x)$, for some x appearing in a condition of a transition in $Adm(q, v) \cup Adm(q, v')$, then there exists $B \subseteq X$ such that $v(x) \neq v'(x)$ iff $x \in B$ and $v'(x) = 0$.

Hence, $Adm(q, v) \neq Adm(q, v')$ if and only if $\{x \mid v(x) = 0\} \neq \{x \mid v'(x) = 0\}$. Therefore, given $(q, q', \psi) \in [g]$ such that $Y = \{x \in X \mid \psi \Rightarrow x = 0\}$, we define the set $\mathcal{F}(\psi)$ as the set of functions f such that, for any state q and set of clocks $B \supseteq Y$, it holds that $f(q, B) \subseteq \delta(q)$ and the formula

$$\psi \wedge \bigwedge_{(q,a,\phi,B',q') \in \delta} \bigwedge_{B \subseteq X} \big((\exists B.\phi \wedge B = 0 \wedge (X \setminus B) > 0) \Leftrightarrow (q, a, \phi, B', q') \in f(q, B)\big) \tag{1}$$

is satisfiable. The set $f(q, B)$ contains the transitions that are enabled in the first component when a configuration, with q as state and with a valuation where exactly the clocks in B are equal to 0, is reached from ψ. Formula (1) ensures that, for any state q and set of clocks B, $f(q, B)$ is the set of transitions enabled when ψ is true. Actually, the formula $\phi \wedge B = 0 \wedge (X \setminus B) > 0$ means that ϕ holds and x is 0 iff $x \in B$. This formula must hold iff the transition is in $f(q, B)$. Symmetrically, we can define $\overline{\mathcal{F}}(\psi)$ for the second component where we consider $Y = \{x \in X \mid \psi \Rightarrow \overline{x} = 0\}$ instead of $Y = \{x \in X \mid \psi \Rightarrow x = 0\}$.

In the following we may write $f(q, B, a)$ with $a \in \Sigma \cup \{\tau\}$ to denote the set $f(q, B) \cap \delta(a)$, where $\delta(a)$ is the set of transitions of A with label a. The number of functions in $\mathcal{F}(\psi)$ is exponential in $|X|$ and δ.

Note that $\mathcal{F}(\psi)$ does not contain necessarily only one function. As an example, if $\psi = 0 < x < 10$ and we have two transitions, one guarded with $\phi_1 = 1 \leq x \leq 3$ and one guarded with $\phi_2 = 2 \leq x \leq 6$, then we have four cases, both ϕ_1 and ϕ_2 do not hold, only ϕ_1 holds, only ϕ_2 holds, and both ϕ_1 and ϕ_2 hold. The four cases depend on the value of x, which is a real number in $(0, 10)$.

We can calculate the probability of performing a sequence of steps labeled with $\tau^*\hat{\alpha}\tau^*$ by analyzing reachability of states in the model of PAs. In particular, we can perform this computation without using labels. Hence we consider PAs with an empty set of labels and with transitions as pairs (z, z'), where z and z' are the starting state and the target state, respectively.

More precisely, with $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \bar{g})$ and $ExecFrag_A^F(s, \tau^*\hat{\alpha}\tau^*, \bar{g})$ we denote $Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$ and $ExecFrag_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C})$, respectively, where $\mathcal{C} = \{s \mid \exists s' \in \mathcal{S}_A.s \approx_{\bar{g}} s'\}$.

In order to check whether for any scheduler F there exists a scheduler F' such that $Prob_A^F(s_1, \tau^*\hat{\alpha}\tau^*, \bar{g}) = Prob_A^{F'}(s_2, \tau^*\hat{\alpha}\tau^*, \bar{g})$ and vice versa, we calculate the probability p_i to reach $\bar{g}$ from s_i by performing $\tau^*\hat{\alpha}\tau^*$, for $i = 1, 2$. To this purpose, we construct two PAs A_1 and A_2 such that the probability to reach a special state *good* in the PA A_i is equal to p_i, for $i = 1, 2$.

In the following, when A performs a time step with $\alpha \in \mathbb{R}^{>0}$ we consider $\alpha = \lambda$, where λ denotes a generic time step.

States of A_1 are either quadruples (q, q', ψ, i) or a state in $\{wrong, good\}$. The quadruple (q, q', ψ, i) represents the configurations (q, v) and (q', v'), with $(q, v) \approx_{\{(q,q',\psi)\}} (q', v')$, that can be crossed to reach two equivalent configurations with respect to $\bar{g}$ starting from s_1 and s_2, respectively. Index i is 1 if α is not performed, i is 2 when α is performed by the first component and $i = 3$ when α is performed by both components.

The *wrong* state is reached in A_1 if a discrete transition step labeled with α', with $\alpha' \notin \{\alpha, \tau\}$, is performed by A.

If $\alpha \in \mathbb{R}^{>0}$, then state *good* is reached in A_1 when $\bar{g}$ is reached in A through a time step α. Otherwise, if $\alpha \in \Sigma \cup \{\tau\}$, then state *good* is reached only if the class $\bar{g}$ can be reached in A through a discrete transition step α performed by any configuration reachable from s_1 and s_2 via τ steps.

The probability associated with two states of the PA $z = (q_1, q_2, \psi, i)$ and $z' = (q_1', q_2, \psi', i')$ is equal to the probability p if from a configuration (q_1, v) with $v \models \psi'$ there exists a step with probability equal to p leading to the configuration (q_1', v') with $v' \models \psi'$.

The probability associated with two states of the PA $z = (q_1, q_2, \psi, i)$ and $z' = (q_1, q_2', \psi', i')$ is equal to 1 if from a configuration (q_2, v) with $v \models \psi'$ there exists a step with label in $\{\alpha, \tau\}$ to the configuration (q_2', v') with $v' \models \psi'$. Actually, the probability computed for the first configuration is not affected by the probabilities of the second one.

PA A_2 is constructed similarly.

Definition 21. Let $\alpha \in \Sigma \cup \{\tau, \lambda\}$. Let $\overline{g}, g$ be two classes and $f \in \mathcal{F}(\overline{\psi})$ and $f' \in \overline{\mathcal{F}}(\overline{\psi})$, for some $\overline{\psi}$. We define $A_1(\overline{g}, g, f, f', \alpha)$, the PA $(\emptyset, Q', q_0, \delta', \Pi')$ such that:

- $Q' = Q \times Q \times \Psi_{C_A}(X \cup \overline{X}) \times \{1, 2, 3\} \cup \{start, good, wrong\}$. Note that Q' is a finite set since the set of clock zones has finite cardinality by Proposition 3.
- $q_0 = start$.
- δ' is the set of pairs $((q_1, q_2, \psi, i), z)$, for some $q_1, q_2 \in Q$, $i \in [1, 3]$ and $\psi \in \Psi_{C_A}(X \cup \overline{X})$, one of the following requirements holds:
  1. $z = (q_1', q_2, \psi', i)$ and $(q_1, q_2, \psi) \in Ap_A(prec(z, e)) \cap \overline{g}$ for some $e \in f(q_1, \{x \mid \psi \Rightarrow x = 0\}, \tau)$, namely a τ step is performed from q_1.
  2. $z = (q_1, q_2', \psi', i)$ and $(q_1, q_2, \psi) \in Ap_A(\overline{prec}(z, e)) \cap \overline{g}$ for some $e \in f'(q_2, \{\overline{x} \mid \psi \Rightarrow \overline{x} = 0\}, \tau)$, namely a τ step is performed from q_2.
  3. $i = 1, \alpha = \lambda, z = (q_1, q_2, \psi', 3)$ and $(q_1, q_2, \psi) \in Ap_A(prec_T((q_1, q_2, \psi'))) \cap \overline{g}$. Namely, z has been reached correctly by the same time step of both components. Since both the components perform the time step the new index is 3.
  4. $i = 1, \alpha \in \Sigma \cup \{\tau\}, z = (q_1', q_2, \psi', 2)$ and $(q_1, q_2, \psi) \in Ap_A(prec(z, e)) \cap \overline{g}$ for some $e \in f(q_1, \{x \mid \psi \Rightarrow x = 0\}, \alpha)$. Namely a step labeled with α is performed from q_1.
  5. $i = 2, \alpha \in \Sigma \cup \{\tau\}, z = (q_1, q_2', \psi', 3)$ and $(q_1, q_2, \psi) \in Ap_A(prec(z, e)) \cap \overline{g}$ for some $e \in f'(q_2, \{\overline{x} \mid \psi \Rightarrow \overline{x} = 0\}, \alpha)$. Namely a step labeled with α is performed from q_2.
  6. $i = 3, z = good$ and $(q_1, q_2, \psi) \in g$. Namely, $\overline{g}$ is correctly reached.
  7. $z = wrong$. This transition simulates the possibility of reaching a configuration through a time step with label different from α (obviously this is always possible) or by a discrete transition step with label in $\Sigma \setminus \{\alpha\}$.
- Π' is the set of distributions π such that, for any $(q_1, q_2, \psi, i) \in Q'$, either there exists a transition $e = ((q_1, q_2, \psi, i), z)$ derived from points 2 or 3 or 5 or 6 or 7, such that $\pi(e) = 1$, or there exists $\pi' \in \Pi$ such that, for any transition $((q_1, q_2, \psi, i), z)$ derived from points 1 or 4 or 7, it holds that

$$\pi((q_1, q_2, \psi, i), z) = \frac{\sum_{e \in E} \pi'(e)}{\sum_{e \in f(q_1, \{x \mid \psi \Rightarrow x = 0\})} \pi'(e)}$$

  where E is one of the following sets:
  - $E = \{e \in f(q_1, \{x \mid \psi \Rightarrow x = 0\}, \tau) \mid (q_1, q_2, \psi) \in Ap_A(prec(z, e))\}$, if $((q_1, q_2, \psi, i), z)$ is a transition derived from 1;
  - $E = \{e \in f(q_1, \{x \mid \psi \Rightarrow x = 0\}, \alpha) \mid (q_1, q_2, \psi) \in Ap_A(prec(z, e))\}$, if $((q_1, q_2, \psi, i), z)$ is a transition derived from 4;
  - $E = \bigcup_{a \in \Sigma \setminus \{\alpha\}} f(q_1, \{x \mid \psi \Rightarrow x = 0\}, a)$, if $((q_1, q_2, \psi, i), z)$ is a transition derived from 7.

The PA $A_2(\overline{g}, g, f, f', \alpha)$ can be defined symmetrically.

The PAs $A_i(\overline{g}, g, f, f', \alpha)$ may have an exponential number of states, but they can be constructed by using a more efficient technique based on backward symbolic analysis thanks to the *prec* operator and the theory of regions and DBMs [12,26,36].

The following lemma states that it is possible to compute the probability of reaching with $\tau^* \hat{\alpha} \tau^*$ the class $\bar{g}$ from the configurations s and s' with $s \approx_z s'$ by computing the probabilities of reaching the state *good* of A^1 and A^2 from $z \in g$, where $A^i = A_i(g, \bar{g}, f, f', \alpha)$ for $i = 1, 2$.

Lemma 9. *Let $\mathcal{G}$ be such that $Clean^A(\mathcal{G}) = \mathcal{G}$ and $\approx_g$ is an equivalence relation for any $g \in \mathcal{G}$. Let $g, \bar{g} \in \mathcal{G}$, and (q_1, v_1) and (q_2, v_2) be two configurations such that $(q_1, v_1) \approx_{\{(q_1, q_2, \psi)\}} (q_2, v_2)$ with $(q_1, q_2, \psi) \in [g]$. Let $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$ and $\alpha' = \alpha$ if $\alpha \in \Sigma \cup \{\tau\}$, and $\alpha' = \lambda$ if $\alpha \in \mathbb{R}^{>0}$. For any $i = 1, 2$ and scheduler F of A there exists a scheduler F' of $A^i = A_i(g, \bar{g}, f, f', \alpha')$, for some $f \in \mathcal{F}(\psi)$ and $f' \in \overline{\mathcal{F}}(\psi)$, such that $Prob_A^F((q_i, v_i), \tau^* \hat{\alpha} \tau^*, \bar{g}) = Prob_{A^i}^{F'}((q_1, q_2, \psi, 1), good)$, and vice versa.*

Proof. By Propositions 4 and 7 we have that, for any $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$ and scheduler F, if $s = (q, v) \xrightarrow{\alpha} s' = (q', v')$ is a step of A, then for any state $z = (q, q'', \psi, i)$ such that $v \models \psi$, there exists a step $\sigma = (q, q'', \psi, i) \longrightarrow (q', q'', \psi', i')$ generated by the set of transitions $f(q, \{x \mid v(x) = 0\}, a)$ with $v' \models \psi$, and a scheduler F' of A^1 such that

$$P^F_{A^1}(\sigma) = P^{F'}_A(s \xrightarrow{a} s). \tag{2}$$

Similarly for the vice versa and A^2.

Moreover, we note that:

- the steps triggered by transition e of the second component do not influence the probability of paths of A^1 since, for any distribution π of A^1, we have that $\pi(e) > 0$ implies $\pi(e) = 1$;
- the steps triggered by transition e of the first component do not influence the probability of paths of A^2 since, for any distribution π of A_2, we have that $\pi(e) > 0$ implies $\pi(e) = 1$;
- the *good* state is reached only from states of A^1 and A^2 with index equal to 3 and hence the *good* state is reached only after a sequence of symbols in $\tau^*\alpha\tau^*$ of both components.

Hence, if $\alpha \in \Sigma \cup \{\tau\}$, then there exists a surjective function ζ from the executions in $ExecFrag_A$ performing words in $\tau^*\alpha\tau^*$ to $ExecFrag_{A^i}$ such that, for any scheduler F of A, there exists a scheduler F_i for A^i such that $Prob^F_A(\{\sigma' \in ExecFrag^F_A \mid \zeta(\sigma') = \sigma\}) = Prob^{F_i}_{A^i}(\sigma)$, for any $i = 1, 2$ and $\sigma \in ExecFrag^{F_i}_{A^i}$ where σ reads a string in $\tau^*\alpha\tau^*$. Therefore, for any $i = 1, 2$ and scheduler F of A there exists a scheduler F_i of A^i such that $Prob^F_A((q_i, v_i), \tau^*\alpha\tau^*, \bar{g}) = Prob^{F_i}_{A^i}((q_1, q_2, \psi, 1), good)$. Similarly we can prove the other direction.

On the other hand, if $\alpha \in \mathbb{R}^{>0}$, then $\alpha' = \lambda$. Hence, the proof is more complex with respect to the case $\alpha \in \Sigma \cup \{\tau\}$ since λ may represent, in A^1 and A^2, two different times.

Hence we consider the case in which $ExecFrag^{F'}_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g})$, for $\alpha \in \mathbb{R}^{>0}$, is empty and the case in which it is not.

1. If $ExecFrag^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g})$ is empty, then we have that there exists a scheduler F' such that $ExecFrag^{F'}_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g})$ is empty and vice versa. Actually, as proved for the case $\alpha \in \Sigma \cup \{\tau\}$, since $prec_T$ ensures that the two components take the same time step, and, since $\approx_g$ is an equivalence relation for any $g \in \mathcal{G}$, there exists a surjective function ζ from the executions in $ExecFrag_A$ performing words in $\tau^*\alpha\tau^*$ to $ExecFrag_{A^i}$ such that, for any scheduler F of A, there exists a scheduler F_i for A^i such that $Prob^F_A(\{\sigma' \in ExecFrag^F_A \mid \zeta(\sigma') = \sigma\}) = Prob^{F_i}_{A^i}(\sigma)$, for any $\sigma \in ExecFrag^{F_i}_{A^i}$ and $i = 1, 2$, if and only if it holds that $ExecFrag^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g}) \neq \emptyset$ or $ExecFrag^F_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g}) \neq \emptyset$. In fact, if either $ExecFrag^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g}) = \emptyset$ or $ExecFrag^F_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g}) = \emptyset$, then A^1 and A^2 cannot reach the state *good*, and hence $Prob^F_A((q_i, v_i), \tau^*\alpha\tau^*, \bar{g})$ could be different from the probability of reaching the state *good* of A^i w.r.t. scheduler F_i, for $i = 1, 2$.

   But, since $Clean^A(\mathcal{G}) = \mathcal{G}$, by Lemma 8, it holds that, if there exists a scheduler F of A such that for any scheduler F' of A', $Prob^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g}) > 0$ iff $Prob^{F'}_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g}) > 0$. Hence, $ExecFrag^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g})$ is not empty iff $ExecFrag^{F'}_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g})$ is not empty.

   Hence, as proved for $\alpha \in \Sigma \cup \{\tau\}$, it holds that for any $i = 1, 2$ and scheduler F of A there exists a scheduler F_i of A^i such that $Prob^F_A((q_i, v_i), \tau^*\alpha\tau^*, \bar{g}) = Prob^{F_i}_{A^i}((q_1, q_2, \psi, 1), good) = 0$, and vice versa.
2. If $ExecFrag^F_A((q_1, v_1), \tau^*\alpha\tau^*, \bar{g})$ is not empty, then, as proved for 1, we have that there exists a scheduler F' such that $ExecFrag^{F'}_A((q_2, v_2), \tau^*\alpha\tau^*, \bar{g})$ is not empty. Hence, as proved for $\alpha \in \Sigma \cup \{\tau\}$, it holds that for any $i = 1, 2$ and scheduler F of A there exists a scheduler F_i of A^i such that $Prob^F_A((q_i, v_i), \tau^*\alpha\tau^*, \bar{g}) = Prob^{F_i}_{A^i}((q_1, q_2, \psi, 1), good) > 0$, and vice versa. □

Example 19. Let us assume the state $(q_2, q_7, 0 \leq x < 3 \wedge x = \bar{x}) \in g$ and the class $\bar{g}$ of the PTA in Fig. 12 such that $\bar{g} = \{(q_3, q_8, 0 \leq x < 3 \wedge x = \bar{x}), (q_8, q_3, 0 \leq x < 3 \wedge x = \bar{x}), (q_3, q_3, x = \bar{x}), (q_8, q_8, x = \bar{x})\}$. We note that $\approx_{\bar{g}}$ is an equivalence relation. Let f be the function such that $f(q, B) = \delta(q)$.

We construct $A_1(g, \bar{g}, f, f, a)$ by considering $((q_2, q_7, 0 \leq x < 3 \wedge x = \bar{x}), 1)$ as the initial state.

We describe probabilities assigned to the transitions starting from state $z = ((q_2, q_7, 0 \leq x < 3 \wedge x = \bar{x}), 1)$. The probability distribution π satisfies one of the following requirements:

- $\pi(z, (q_2, q_7, 0 \leq x < 3 \wedge x = \bar{x}, 2)) = \pi(z, wrong) = 0.5$ (representing the steps of the first component labeled with a or b).
- $\pi(z, wrong) = 1$.
- $\pi(z, z) = 1$ (representing the τ step of the second component).

We describe now the probabilities assigned to the transitions starting from the state $z = (q_3, q_7, 0 \leq x < 3 \wedge x = \overline{x}, 2)$. The probability distribution π satisfies one of the following requirements:

- $\pi(z, (q_3, q_8, 0 \leq x < 3 \wedge x = \overline{x}, 3)) = 1$ (representing the step of the second component labeled with a).
- $\pi(z, wrong) = 1$.
- $\pi(z, z) = 1$ (representing the τ step of the second component).

Finally, for the state $(q_3, q_8, 0 \leq x < 3 \wedge x = \overline{x}, 3)$ we have only one transition to the state *good* with probability equal to 1.

With $Cut^A(g, \bar{g})$ we denote the set $\{g_1, \ldots, g_n\} \in Set(A)$ such that, $[g] \supset \bigcup_{i \in [1,n]} [g_i]$, and for any $\alpha \in \Sigma \cup \{\tau, \lambda\}$ it holds that $(q_1, q_2, \psi), (q_1', q_2', \psi') \in [g_i]$, for some $i = 1, \ldots, n$, iff for any $f \in \mathcal{F}(\psi), f' \in \overline{\mathcal{F}}(\psi')$ and scheduler F there exists a scheduler F' such that $Prob^F_{A^1}((q_1, q_2, \psi), good) = Prob^{F'}_{A^2}((q_1', q_2', \psi'), good)$, where $A^j = A_j(\bar{g}, f, f', \alpha)$ with $j = 1, 2$.

Example 20. Let us assume the classes g_1 and g_2 of Example 19. We have that $Cut^A(g_1, g_2) = \{\{(q_2, q_2, 0 \leq x < 3 \wedge x = \overline{x})\}, \{(q_7, q_7, 0 \leq x < 3 \wedge x = \overline{x})\}\}$.

Note that to compute the probabilities $Prob^F_{A^i}(z, good)$ it is sufficient to compute only once the automata $A^i = A_i(g, \bar{g}, f, f', \alpha)$, for $i = 1, 2$. Since the number of clock zones is exponential w.r.t. the size of A, we have that automata $A^i = A_i(g, \bar{g}, f, f', \alpha)$, for $i = 1, 2$, have an exponential number of states with respect to the size of A. By Proposition 1, we have that $Prob^F_{A^i}(z, good)$ is computable in exponential time on the size of A^i (in polynomial time if A^i has no τ transitions). The next corollary follows.

Corollary 1. *For any $g, \bar{g}$, $Cut^A(g, \bar{g})$ is computable in double exponential time w.r.t. the size of A. If A has no τ transitions, $Cut^A(g, \bar{g})$ is computable in exponential time w.r.t. the size of A.*

Proof. To compute $Cut^A(g, \bar{g})$ we must construct once $A^i = A_i(g, \bar{g}, f, f', \alpha)$, for $i = 1, 2$, and we must check when $Prob^F_{A^1}((q_1, q_2, \psi), good) = Prob^{F'}_{A^2}((q_1', q_2', \psi'), good)$.

Now, the numbers of (q_1, q_2, ψ) and (q_1', q_2', ψ') are at most exponential on the size of A since they are bounded on the number of clock zones. Hence, we must compute $Prob^F_{A^1}(z, good)$, for at most an exponential number on the size of A of s. Therefore, the complexity is equal to $2^{|A|} \times I$ where I is the cost to compute the value $Prob^F_{A^1}(z, good)$.

We recall that to compute the value of $Prob^F_{A^i}(z, good)$ it is sufficient to construct only once the automata $A^i = A_i(g, \bar{g}, f, f', \alpha)$, for $i = 1, 2$. Since the number of clock zones is exponential w.r.t. the size of A, we have that automata $A^i = A_i(\bar{g}, f, f', \alpha)$, for $i = 1, 2$, have an exponential number of states with respect to the size of A. By Proposition 1, we have that $Prob^F_{A^i}(z, good)$ is computable in exponential time on the size of A^i (in polynomial time if A^i has no τ transitions).

Therefore, we have that the complexity is $2^{|A|} \times 2^{2^{|A|}}$ and $2^{|A|} \times 2^{|A|}$ if A has no τ transitions. □

4.4. The algorithm

We can now define the algorithm $Classes(A)$ that returns a set in $Set(A)$ giving the configurations that are bisimilar. The algorithm refines the classes by using the triples returned by the algorithm *Clean* and the *Cut* operator until the fixpoint is reached. The algorithm starts with the class $\bigcup_{q,q' \in Q} \{(q, q', X \geq 0 \wedge \overline{X} \geq 0)\}$ (namely, the class containing all configurations). The refinement is done by splitting according to $Cut^A(g, g')$. The "for each" command enumerates the pairs g, g' in $\mathcal{G}$ at the moment in which the first cycle is processed. Hence, the command $\mathcal{G} := (\mathcal{G} \setminus \{g\}) \cup \mathcal{G}''$ does not influence the execution of the "for each" command.

```
Classes(A : PTA) : Set(A)
  G := ⋃_{q,q'∈Q} {(q, q', X ≥ 0 ∧ X̄ ≥ 0)}
  repeat
    G' := G
    G := Clean^A(G)
    for each g, g' ∈ G
      G'' := Cut^A(g, g')
      if G'' ≠ {g} then G := (G \ {g}) ∪ G''
  until (G == G')
  return G
```

We have the following theorem stating the correctness of the algorithm. This implies the decidability of weak bisimulation for PTAs.

Theorem 2. *For any configuration $s, s' \in \mathcal{S}_A$, $s \approx_A s'$ if and only if $s \approx_{Classes(A)} s'$. Moreover, $Classes(A)$ is computable in double exponential time w.r.t. the size of A. If A has no τ transitions, then $Classes(A)$ is computable in exponential time w.r.t. the size of A.*

Proof. By Lemma 6, it is sufficient to prove that $s \approx_{Class(A)} s' \Leftrightarrow \forall n \geq 0 \quad s \sim_n s'$. Let $\mathcal{G}_1, \ldots, \mathcal{G}_n, \ldots$ be the values of variable $\mathcal{G}$ at each step of the algorithm. We prove by induction that $s \approx_{\mathcal{G}_i} s' \Leftrightarrow \forall n \in \{0, \ldots, i\}. s \sim_n s'$.

The base case is obvious since $\mathcal{G}_0 := \bigcup_{q,q' \in Q} \{(q, q', X \geq 0 \wedge \overline{X} \geq 0)\}$.

By induction we have that $s \approx_{\mathcal{G}_i} s' \Leftrightarrow \forall n \in \{0, \ldots, i\}. s \sim_n s'$. Therefore $\approx_g$ is an equivalence relation, for any $g \in \mathcal{G}_i$. Hence, by Lemmas 8 and 9, we have that $s \approx_{\mathcal{G}_{i+1}} s' \Leftrightarrow \forall n \in \{0, \ldots, i+1\}. s \sim_n s'$.

The fixpoint is computable after a finite number of steps since the number of formulae is finite and, at each step of the algorithm, the set $\mathcal{G}$ is updated with a set of classes included in the previous set of classes. The number of iterations is bounded by the number of clock zones. Actually, at each step, our algorithm refines the classes and partitions some of them as the classical algorithm for weak bisimulation on finite state machines. In such a case the algorithm is polynomial on the number of states, whereas in our case, the states are the possible set of regions that are exponential on the size of A.

Hence, the cost of the algorithm is given by the formula $2^{|A|} \times I$, where I is the cost of each iteration.

The cost I of each iteration is given by $I_1 + I_2 \times I_3$ where:

- I_1 is the cost of $Clean^A(\mathcal{G})$ plus the costs of assignments and boolean check outside the command **for each**, that is exponential time on the size of A by Lemma 8 and since the size of a class is bounded by the number of regions.
- I_2 is equal to the number of iterations of the command **for each**. Since the size of $\mathcal{G}$ is bounded by the number of clock zones, I_2 is exponential w.r.t. the size of A.
- I_3 is the cost of each iteration of I_2. This cost depends on the cost of Cut^A plus the costs of assignments and boolean check inside the command **for each**. Hence I_3 is double exponential time w.r.t. the size of A because of the Cut^A operator (see Corollary 1). Note that if A has no τ transitions, then the Cut^A operator is exponential time (see Corollary 1), and hence I_3 is exponential time w.r.t. the size of A.

Therefore the complexity is of the order $2^{|A|} \times (2^{|A|} + 2^{|A|} \times 2^{2^{|A|}})$. Hence, $Class^A(\mathcal{G})$ is computable in double exponential time w.r.t. the size of A. If A has no τ transitions, then $Class(A)$ is computable in exponential time w.r.t. the size of A since the Cut^A operator costs exponential time w.r.t. A. □

5. Discussion

In this section we discuss two important topics, namely non-zeno runs and classical definition of PTA without normalization (see [35]).

5.1. Non-zeno schedulers

A scheduler F is *non-zeno* if for each execution σ in $ExecFrag_A^F$, the sum of the times occurring in σ diverges. The general notion of scheduler we gave defines both zeno and non-zeno schedulers.

Weak bisimulation implies weak bisimulation restricted to non-zeno schedulers. Let us formalise our conjecture as follows.

Consider two configurations s and s' such that $s \approx_A s'$ (recall that $\approx_A$ denotes the weak bisimilarity w.r.t. general scheduler).

For any F such that F is a non-zeno scheduler, we construct a non-zeno scheduler F' such that, for all $\mathcal{C}$ and α, it holds that

$$Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C}).$$

This obviously implies the weak bisimilarity restricted to non-zeno schedulers.

Now since $s \approx_A s'$, there exists a scheduler F'' such that, for all $\mathcal{C}$ and α, it holds that

$$Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F''}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C}).$$

We define F' in the same manner of F'' for the execution fragments performing a string that is a proper prefix of $\tau^*\hat{\alpha}\tau^*$.

Therefore F' is now defined for the execution fragments starting from s' that perform a string of the form $\tau^*\hat{\alpha}\tau^*$ and such that the last configuration is in $\mathcal{C}$, for some $\mathcal{C}$ and α. We call this set of execution fragments $E_1(\mathcal{C}, \alpha)$.

Note that this part ensures that

$$Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F''}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C}),$$

for all $\mathcal{C}$ and α.

Now, we can iterate this step. Actually, for any $\mathcal{C}$ and α, let $\bar{s}$ be a configuration in $\mathcal{C}$ reached from s by performing a string in $\tau^*\hat{\alpha}\tau^*$. For any $\sigma \in E_1(\mathcal{C}, \alpha)$ we have that $\bar{s}, last(\sigma) \in \mathcal{C}$, and this means that $\bar{s} \approx last(\sigma)$.

Hence, there exists F''' such that

$$Prob_A^F(\bar{s}, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'''}(last(\sigma), \tau^*\hat{\alpha}\tau^*, \mathcal{C}).$$

By repeating the first part of the construction, we can define F'' for the executions $\sigma_1\sigma_2$ such that $\sigma_1 \in E_1(\mathcal{C}, \alpha)$ and σ_2 performs a string of the form $\tau^*\hat{\alpha}\tau^*$ and such that the last configuration is in $\mathcal{C}$, for some $\mathcal{C}$ and α.

Hence, by iterating, we have defined F' such that

$$Prob_A^F(s, \tau^*\hat{\alpha}\tau^*, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\hat{\alpha}\tau^*, \mathcal{C}).$$

Moreover, F' assigns the same sequence of times assigned by F, hence is non-zeno.

Vice versa, weak bisimulation restricted to non-zeno schedulers does not imply weak bisimulation. Actually, if we consider a state from which the only exiting transition is a loop with condition $x \leq c$, where x is a clock not reset by the loop, a run with a suffix composed by that loop is obviously a zeno run. If two configurations are not weakly bisimilar because of that loop, the two configurations will be weakly bisimilar in a non-zeno scheduler setting. Actually, if we consider a non-zeno scheduler, that state will never be reached.

We believe that the algorithm we propose could be modified for considering non-zeno schedulers; it is sufficient to consider only clock zones from which at least a non-zeno run can be performed.

5.2. Normalization

As already mentioned, our definition of PTAs requires a run-time re-normalization of probabilities when time constraints prevent the execution of some transitions. Situations of this kind may arise whenever an automaton returns to a certain state at different times and some of the transitions available from that state are prevented.

As we will show in the next example, run-time re-normalization of probabilities allows us to depict a dynamical reallocation of probabilities when several resources are available to a system while others are prevented due to timing constraints.

We also show that re-normalization, on the one hand, allows us to relax the condition of *admissible target states* used in [35] and, on the other hand, gives a more succinct description of systems.

Example 21. Let us consider a system that can allocate n mutual exclusive resources. Execution of action a_i models the system acquiring the ith resource (r_i), while execution of action b_i represents the system releasing resource r_i.

Before the system can acquire a resource, the resource must be initialized, and we suppose that a resource initializes itself after a time amount of length c has elapsed. Hence, a resource cannot be acquired if either a time c has not elapsed from its last release or if a time c has not elapsed from the beginning of the initialization procedure.

Thus, at a certain instant of time only a subset of the n resources are available. Assuming the system chooses one of the n resources with uniform probability distribution, if k resources are available at a certain instant of time, re-normalization assures that the system chooses one of them with a probability equal to $\frac{1}{k}$.

This system can be modeled by a PTA with $n + 1$ states $\{q, q_1, \ldots, q_n\}$ and n clocks $\{x_1, \ldots, x_n\}$. At state q the system may acquire a resource, in state q_i the system is using resource r_i. Clocks x_i are used to check the initialization times for resources r_i.

We have just two kinds of transitions:

- transitions acquiring a resource have the form $(q, a_i, x_i \geq c_i, \{\}, q_i)$, for all $i \in [1, n]$;
- transitions releasing a resource have the form $(q_i, b_i, true, \{x_i\}, q)$, for all $i \in [1, n]$.

We consider invariants equal to *true*, and, finally, we have only one probability distribution π such that $\pi(q, a_i, x_i \geq c_i, \{\}, q_i) = \frac{1}{n}$ and $\pi(q_i, b_i, true, \{x_i\}, q) = 1$, for all $i \in [1, n]$.

In [35] a different definition of PTAs is given. There is no normalization of probabilities and a finite non empty discrete probability distribution on $Q \times 2^X$ is associated with each state. Moreover, each discrete probability has an enabling condition. The definition does not consider symbols, but it could be easily extended to consider actions labeling transitions. Moreover, in [35] there are some requirements on invariants. More precisely, it is required that:

- if in a configuration a certain amount of time violates the invariant, before the violation at least an enabling condition is satisfied, and
- it is never possible to perform a discrete step to a state for which the invariant is not satisfied.

These requirements could be artificial, hence we have chosen to consider infinite and finite paths.

Now, if we extend the definition of [35] by introducing actions, we do some succinctness considerations.

We can simulate (in the sense of strong bisimulation) the definition of [35]. Given a PTA defined as in [35], we can construct a PTA with our definition in polynomial size. Actually, it is sufficient to add, for each discrete probability π associated with state q and with enabling condition ϕ, a set of transitions of the form (q, a, ϕ, B, q'), for any q' and B such that $\pi(q', B) > 0$. Moreover we add a distribution function π' such that $\pi'(q, a, \phi, B, q') = \pi(q', B)$, for any q' and B such that $\pi(q', B) > 0$. Note that the requirements on the invariants of [35] mean that the model translated in our definition has only infinite paths.

It is worth noticing that, if we consider Example 21, the minimal PTA following the definition of [35] is of exponential size w.r.t. our modelization. Actually, the PTA must consider all possible subsets in $\{1, \ldots, n\}$ of possible ready resources

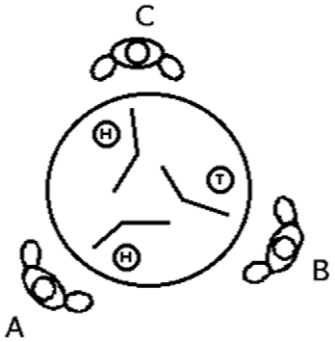

Fig. 13. The dining cryptographers.

by using at least 2^n distribution functions (showing that the minimal PTA has an exponential size). Note that invariants equal to *true* do not give any problems of simulation. This proves that our definition is exponentially more succinct than the definition of [35]. Moreover, Example 21 shows that an interesting practical problem could cause this growth up. Notice that this discussion depends neither on the uniform distribution nor on the amount of time c (that can be differentiable for each resource).

6. An application of weak bisimulation for PTAs

Security is one of the features of a system that one frequently needs to guarantee. Consider, for example, a two level system scheme in which one does not want interference between confidential data of the high level and the low level behaviour observed by an attacker who could infer confidential information (exploiting, for example, a covert channel).

Analyzing the behaviour of a system, aspects of time and/or probability could be exploited to undermine security. Intuitively, a system which in a nondeterministic setting is considered to be secure, in a richer framework, where the duration of certain actions, or the probability distribution of the possible events, is known, may reveal information leakages.

Aldini et al. [1], for example, observe that an attack could be done in the following way. Assume that a secret 1-bit value can be communicated to the unauthorized user among randomly created low-level noise, and that both secret value and random low-level noise belong to the same domain. The high behaviour does not alter the set of possible low outcomes which are always the same with or without the high-level interaction. However, for a fixed value of the high input, an attacker observing the frequency of the low results deriving from repeated executions of the system could infer (with a certain probability) which one is directly communicated by the high user.

Besides, a timing attack is a side channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms such as encryption, decryption, hashing, etc. (see, for example, [34,14]). The attack exploits the fact that every operation in a computer takes time to execute, hence information may leak from a system through measurement of the time it takes to respond to certain queries. How much such information can help an intruder depends on many variables: the cryptosystem design, the CPU running the system, the used algorithms, the timing attack countermeasures, the accuracy of the timing measurements, etc.

In this section we introduce Chaum's dining cryptographers protocol, we model the principles with PTAs, and we capture a timing attack with our notion of weak bisimulation.

6.1. The dining cryptographers protocol

The dining cryptographers protocol, introduced by Chaum in [19], is a method for anonymous communication which offers untraceability of both the sender and the recipient. We recall Chaum's introduction to the dining cryptographers problem.

Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maitre d'hotel for the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it might have been NSA (US National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol.

Each cryptographer flips an unbiased coin behind his menu, between him and the cryptographer on his right, so that only the two of them can see the outcome. Each cryptographer then states aloud whether the two coins he can see – the one he flipped and the one his left-hand neighbor flipped – fell on the same side or on different sides. If one of the cryptographers is the payer, he states the opposite of what he sees. An odd number of differences uttered at the table indicates that a cryptographer is paying; an even number indicates that NSA is paying (assuming that the dinner was paid for only once). Yet if a cryptographer is paying, neither of the other two learns anything from the utterances about which cryptographer it is.

Example 22. Consider the situation depicted in Fig. 13, and suppose NSA is paying the bill. In this case cryptographer A (who can see his own coin and C's coin, both showing heads) states that the two coins fell on the same side. Cryptographers B and C, however, see a head and a tail, hence they state the coins they can see fell on different sides. Since the number of cryptographers who state the coins are different is even, the protocol correctly reveals NSA is paying.

In case cryptographer A payed the bill he would state that the two coins he can see are different. Hence, the number of cryptographers stating the coins are different becomes odd, revealing that the bill was paid by one of the cryptographers.

It is easy to see that the protocol works correctly also if either B or C is paying.

The method can be improved in order to anonymously communicate general messages as follows. Instead of just throwing a coin, every cryptographer picks a random number in private and shows it to the cryptographer to the right. Then, each cryptographer computes the difference between his own number and the number he was shown by his left neighbor, adding a message if he wants to transmit one. Each cryptographer publicly announces the result. The published numbers are summed, and, if the sum is 0, no message was sent. If the sum is a valid message, one cryptographer transmitted a message. If the sum is invalid, more than one cryptographer tried to transmit a message; they just wait a random time and try again.

Recipient anonymity is quite simple: Everybody receives the message at the same time (when the announced values are summed up), so the message could be meant for anybody. Sender anonymity holds because no cryptographer knows the number of the person to his right. Therefore, each cryptographer (other than himself) appears to be equally likely to be the one who added the message.

Example 23. Consider again the situation depicted in Fig. 13, but, instead of flipping a coin, assume each cryptographer picks a random number. Let n_A, n_B and n_C be the random numbers extracted by cryptographers A, B and C respectively. Cryptographer C, moreover, wants to anonymously transmit the message m. We have the following situation:

- cryptographer A computes and announces: $n_A - n_C$;
- cryptographer B computes and announces: $n_B - n_A$;
- cryptographer C computes and announces: $n_C - n_B + m$;
- the announced values are summed up resulting in message m.

In a purely untimed setting this protocol correctly preserves anonymity of both the sender and the receiver, however it is subject to a very simple timing attack. Actually, even if cryptographers are known to be excellent in mathematics, it takes them some time to perform the computations needed by the protocol. Now, while cryptographers non transmitting messages just perform a single subtraction, the cryptographer who wants to transmit a message needs to perform a subtraction and a summation. Hence, by observing the time that elapses before announcing the result of the computation, the cryptographers may be able to detect the sender of the message.

6.2. The PTA model of the protocol

As a toy example, we represent the dining cryptographers protocol within the PTA model and we analyze it by using our notion of weak bisimulation. Namely, we define an anonymity property in terms of behavioural equivalence by resorting to our notion of weak bisimulation.

Actually, we are able to model the timing attack presented in Example 23. Now, since the timing attack can be detected only by observing action durations, and since the protocol we are modeling is intrinsically probabilistic, a framework is needed where both time and probability are taken into account.

We consider the simpler case introduced in Example 22, but, keeping in mind the timing attack shown in Example 23, we assume that each cryptographer takes some time to check whether the two coins are the same or not, and, in case he is paying, he also takes some time to compute the negation of what he sees. We give and compare the PTAs modeling the behaviour of a cryptographer when he is the one paying the bill or not.

In Fig. 14 we depict the PTA model of a paying cryptographer (A_{PC}). The coin tossing phase of the protocol is represented by the four initial transitions leading, each with probability $\frac{1}{4}$ and label τ, to states $q_{bb'}$, with $b, b' \in \{h, t\}$. Intuitively, b' represents the result of the coin tossed by the modeled cryptographer, while b represents the result of the coin tossed by the person to his left. Since coins are unbiased, the probability distribution over states $q_{bb'}$ can be assumed to be uniform. Notice that we are also assuming the coin tossing phase to last, at least, a fixed amount of time t_T. In order to check whether the two coins are equal or different, in state $q_{bb'}$ the cryptographer computes the XOR between the two coins, and after a time t_X can reach either state q_D, if the coins are different, or state q_E if the two coins are equal. Since the cryptographer we are modeling is the one who is paying the bill, he should announce the contrary of what he sees. Hence, he performs a NOT operation with time duration at least t_N. The only visible actions are the announcements *equal* or *different* performed from states $q_{\bar{D}}$ and $q_{\bar{E}}$, respectively, and leading to the final state q_F.

In Fig. 15 we show the simpler PTA model of a non paying cryptographer (A_{NPC}), where the transitions modeling the negations in the paying cryptographer behaviour are omitted.

Probable innocence is a notion of anonymity introduced by Reiter and Rubin in [51]. While different formal definitions can be found in the literature (see [17] for details), probable innocence intuitively states that an agent appears no more likely to be the culprit than not to be.

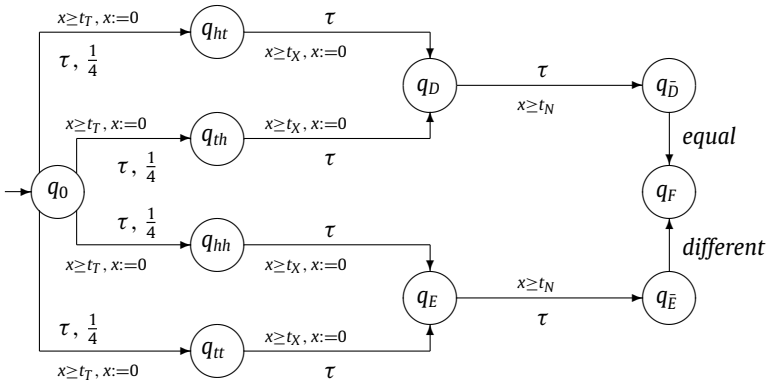

Fig. 14. A_{PC}: paying cryptographer.

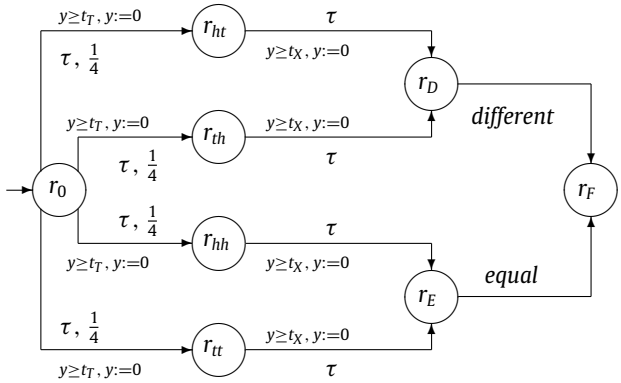

Fig. 15. A_{NPC}: non paying cryptographer.

Thus, if the protocol preserves probable innocence, the behaviour of a cryptographer when he is paying the bill should not be distinguishable from the behaviour of the cryptographer when he is not paying the bill. In terms of our notion of weak bisimulation, we should require that:

$$A_{PC} \approx A_{NPC}.$$

Now, it is easy to see that the two automata we are considering are not weakly bisimilar. Intuitively, while automaton A_{PC} performs an observable announcement *equal* or *different* only after a time $t_T + t_X + t_N$ has elapsed, automaton A_{NPC} can perform such announcements after a time $t_T + t_X$.

Namely, while there exists a scheduler F such that $(r_0, y = 0) \xrightarrow{t_T} (r_0, y = t_T) \xrightarrow{\tau} (r_{hh}, y = 0) \xrightarrow{t_X} (r_{hh}, y = t_X) \xrightarrow{\tau} (r_E, y = 0) \xrightarrow{equal} (r_F, y = 0)$ is a valid run for A_{NPC}, there is no analogous run in A_{PC} for any scheduler.

Such a timing attack can be avoided by synchronizing the announcements of all cryptographers. This, in turn, can be achieved by introducing a timeout in the last announcement. Actually, we might require the non paying cryptographer to wait in state r_D or r_E for the paying cryptographer (if any) to perform all his computation. This can be modeled in A_{NPC} by adding the condition $y \geq t_N$ in the final transitions reaching state r_F. In such a way, each cryptographer announces his results at a time greater than $t_T + t_X + t_N$ whether he is paying or not.

We denote with A'_{NPC} the automaton modeling the cryptographer modified as just explained above, and we can prove that in this case $A_{PC} \approx A'_{NPC}$, thus satisfying the anonymity property.

Namely, there exists a weak bisimulation relation $\mathcal{R}$ with classes defined as follows.

- $\mathcal{C}_1 = (q_0, r_0, x = y < t_T)$,
- $\mathcal{C}_2 = (q_0, r_0, x = y \geq t_T)$,
- $\mathcal{C}_3 = (q_{ht}, q_{th}, x = y < t_X) \cup (q_{ht}, r_{hh}, x = y < t_X) \cup (r_{hh}, r_{tt}, x = y < t_X)$,
- $\mathcal{C}_4 = (q_{hh}, q_{tt}, x = y < t_X) \cup (q_{hh}, r_{ht}, x = y < t_X) \cup (r_{ht}, r_{th}, x = y < t_X)$,
- $\mathcal{C}_5 = (q_{ht}, q_{th}, x = y \geq t_X) \cup (q_{ht}, r_{hh}, x = y \geq t_X) \cup (r_{hh}, r_{tt}, x = y \geq t_X)$,

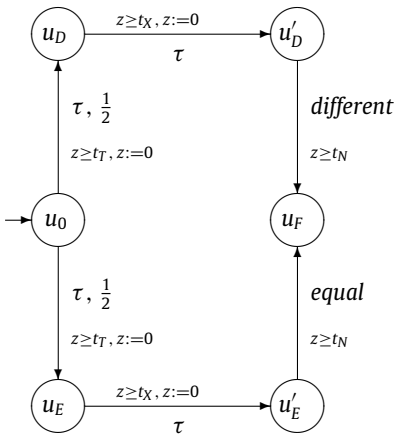

Fig. 16. A_C: the reduced cryptographer.

- $\mathcal{C}_6 = (q_{hh}, q_{tt}, x = y \geq t_X) \cup (q_{hh}, r_{ht}, x = y \geq t_X) \cup (r_{ht}, r_{th}, x = y \geq t_X)$,
- $\mathcal{C}_7 = (q_D, r_E, x = y < t_N)$,
- $\mathcal{C}_8 = (q_E, r_D, x = y < t_N)$,
- $\mathcal{C}_9 = (q_D, q_{\tilde{D}}, x = y \geq t_N) \cup (q_D, r_E, x = y \geq t_N)$,
- $\mathcal{C}_{10} = (q_E, q_{\tilde{E}}, x = y \geq t_N) \cup (q_E, r_D, x = y \geq t_N)$,
- $\mathcal{C}_{11} = (q_F, r_F, x = y \geq t_N)$.

For simplicity we omitted triples generated from commutativity and transitivity within each class $\mathcal{C}_i$. Namely, if $(q, r, \psi), (q', r', \psi) \in \mathcal{C}_i$, then also $(r, q, \psi), (q, r', \psi), \ldots \in \mathcal{C}_i$. Note that $(q_F, r_F, x = y \geq t_N) \in \mathcal{C}_{11}$. Hence, $\mathcal{C}_{11}$ contains the final states of A'_{PC} and A'_{NPC}. Actually, any configuration in $\mathcal{C}_{11}$ is terminal.

Now, since $\mathcal{R}$ is a weak bisimulation on $\mathcal{S}_{\hat{A}}$, and since the initial configurations of A'_{PC} and A'_{NPC} are in the same class we have that the two automata are weakly bisimilar.

Finally, we would like to show the use of weak bisimulation as a state space reduction technique. Looking at the classes of $\mathcal{R}$ one can notice that for any configuration in $\mathcal{C}_2$ we can reach, through a τ transition, class $\mathcal{C}_3$ or $\mathcal{C}_4$ with probability $\frac{1}{2}$. Such a consideration immediately suggests how to reduce the state space of A_{PC} and A'_{NPC}. In fact, we have that the two automata are both bisimilar to the reduced automaton A_C in Fig. 16.

7. Related works

A preliminary version of this paper has appeared in [38]. The notion of bisimulation has already been introduced and studied for real-time models (e.g. in [57,20,47,5,4]) and probabilistic models (see [7,8,15,22,1]).

On the one hand, simulation and bisimulation were shown to be decidable for finite timed transition systems by Asarin et al. [6] and Cerans [20], respectively. In the latter paper, bisimulations are defined in terms of the uncountable unfolded version of the given timed transition systems, and the decision mechanisms produce relations over nontrivial regional constructions.

On the other hand, Baier and Hermanns [7] introduced a notion of weak bisimulation for fully probabilistic systems, and gave an algorithm to decide it with a time complexity cubic in the number of states of the fully probabilistic system.

In [49] algorithms for deciding weak bisimulation for Labeled Concurrent Markov chains are proposed. In [15] the problem of deciding weak bisimulation in the context of probabilistic nondeterministic automata is studied. Our work is close to this last one for the context and the algorithm used, but time features introduced more complex problems to be solved (see Section 4).

There is also some work on Stochastic Process Algebras where probability and timing are combined. In [28], minimization algorithms based on behavioural equivalences for stochastic process algebras are discussed. Some examples of such equivalence relations are strong equivalence [29], strong (and weak) Markovian bisimilarity [27], and extended Markovian bisimilarity [10].

Among the papers on bisimulation (or simulation) relations based on automata models, we still could cite some works.

In [45], Lynch and Vaandrager present a variety of simulation proof techniques for a general automaton based model for timed systems. They also show how some results for untimed automata can be carried over to the setting of Timed Automata.

As regards Probabilistic Automata we should cite Segala's thesis [53], in which the common semantics for labeled transition systems is extended to the probabilistic framework. A compositional trace semantics is defined where a trace is replaced by a probability distribution over traces. The classical bisimulation and simulation relations are extended according

to such a trace semantics both in their strong and weak versions. Furthermore, probabilistic forward simulations are defined, where a state is related to a probability distribution over states.

In Stoelinga's thesis [56], a new notion of bisimulation relation, called *delay bisimulation*, is introduced for Probabilistic Automata. While this relation abstracts away from internal moves only in a limited way, there is a gain in the efficiency of the algorithms that automatically compute the bisimilar states. Namely, delay bisimulation is shown to be decidable in polynomial time and space.

Jensen and Gregersen [31,32] present a model which is similar to Probabilistic Timed Automata but obtained as a generalization of reactive systems [43]. They also present a specification formalism in terms of a real timed probabilistic logic and a model checking method for verification with respect to the logic. However, differently from Segala's probabilistic automata, their model cannot have nondeterministic choice between transitions labeled with the same action.

Finally, in [55] probabilistic timed simulation and bisimulation relations for Probabilistic Timed Automata are studied. An EXPTIME algorithm for deciding whether two Probabilistic Timed Automata are probabilistically timed similar or bisimilar is presented together with a a logical characterization of probabilistic timed bisimulation.

On the interpretation of timed steps. In [57,44], Wang and Larsen introduce strong and weak bisimulation equivalences for a real time process calculus obtained by extending Milner's CCS [46] with delays. Our notion of weak bisimulation for PTAs differs from the mentioned one. In our framework, a weak transition $\stackrel{t}{\Longrightarrow}$ is given by a sequence of internal moves followed by a single time transition of duration t followed by a sequence of internal moves. In [57] both time delays and internal moves are interchangeable, the observable resulting after a series of time delays and internal moves is just the time delay obtained summing up the different timed steps: a weak timed transition consists of a sequence of steps (either internal or timed steps) such that the sum of the different time steps is t. Thus, the main difference is that our notion of weak bisimulation is *weak* only with respect to internal moves, while the notion of weak bisimulation in [57] is *weak* with respect to both internal and timed steps.

In [57] also a notion of strong bisimulation is given (which is similar to our notion when abstracting from internal moves). In a sense, we relax this equivalence by introducing internal invisible moves, which then lead to the definition of our *intermediate* weak bisimulation.

As a consequence, we are treating as *visible* consequent timed steps. This becomes actually relevant when consequent timed steps are interleaved with invisible τ actions which might change a time invariant by leading to a new state with different time constraints, or the probability of observing the passage of time (see Example 13). Such an assumption makes finer the classes of our weak bisimulation relation but allows us to take into account the fact that the scheduler is invoked again after a timed step. While no assumption about the scheduler is needed in [57] (since their model is purely possibilistic, nondeterminism could be treated in the classical way), in this paper we used schedulers to combine probabilities and nondeterminism. Such an invocation would actually make distinguishable, to the eyes of a malicious scheduler, a system which can perform a single timed step of length t from a system which can perform two timed steps (maybe interleaved by some internal move) with lengths t_1 and t_2 such that $t_1 = t_2$, and so alter its observable behaviour (see [16,18] for examples on how malicious schedulers may collaborate with an attacker allowing him to distinguish two bisimilar processes). Moreover, notice that in our framework the scheduler also decides the amount of time to pass. We believe that, for example in the context of security analysis, our stricter notion of bisimulation is more suitable than the one presented in [57].

8. Conclusions

We have considered a model of Probabilistic Timed Automata and we have presented a notion of weak bisimulation in order to compare automata behaviour. We propose an algorithm based on symbolic representation that allows to decide weak bisimulation with a complexity congruent with the algorithm given for the untimed version. We have applied such a notion in the context of security analysis by modeling a timing attack on a protocol where both time and probability play a role. Therefore, our framework can be used, for example, to prevent possible attacks based on statistical or timing analysis.

While the notion of weak bisimulation we introduced in this paper is quite strict, a notion of approximate weak bisimulation could be extremely useful when analyzing performance or security aspects of probabilistic systems.

In order to introduce a quantitative measure for insecure behavior (hence, to estimate the probability that a certain insecure behavior arises), one may resort to an approximate notion of weak bisimulation for deciding if two systems behave almost in the same way, namely, to assess bisimilarity of automata for which the difference between their probabilistic behaviour is within a small distance.

In [13,22,23], for example, metrics are introduced in order to quantify the similarity of the behaviour of probabilistic transition systems that are not strictly bisimilar. In [1] the authors introduce an enriched notion of weak probabilistic bisimulation, which is able to tolerate fluctuations making the security conditions less restrictive and relating systems that may have largely different possible behaviour under the condition that such behaviour is observable with a negligible probability. It would be interesting to extend also our notion of weak bisimulation with an approximate one.

As another possible improvement of this work, one may consider reasoning about the compositionality of PTAs, thus allowing for the analysis of more complex systems. A lot of work has been done on probabilistic parallel composition

operators (see [53,21,54]). Hence, it would be interesting to study the application of parallel composition operators within the model of PTAs, and verify, for example, whether weak bisimulation is a congruence with respect to these constructs.

A more practical direction may consist in the development of tools for the automatic verification of weak bisimulation for PTAs or for the state space reduction of PTAs. This latter point could improve the verification of PTAs through model checkers for PTAs (see, for example, PRISM [50,36]).

References

[1] A. Aldini, M. Bravetti, R. Gorrieri, A process-algebraic approach for the analysis of probabilistic non-interference, Journal of Computer Security 12 (2004) 191–245.
[2] R. Alur, C. Courcoubetis, D.L. Dill, Verifying automata specifications of probabilistic real-time systems, in: Real-Time: Theory in Practice, in: Springer LNCS, vol. 600, 1992, pp. 28–44.
[3] R. Alur, D.L. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.
[4] J.H. Andersen, K.J. Kristoffersen, K.G. Larsen, J. Niedermann, Automatic synthesis of real time systems, in: Proc. of ICALP'95, in: Springer LNCS, vol. 944, 1995, pp. 535–546.
[5] H.H. Andersen, M. Mendler, An asynchronous process algebra with multiple clocks, in: Proc. of ESOP'94, in: Springer LNCS, vol. 788, 1994.
[6] E. Asarin, O. Maler, A. Pnueli, On discretization of delays in timed automata and digital circuits, in: Proc. of CONCUR'98, in: Springer LNCS, vol. 1466, 1998, pp. 470–484.
[7] C. Baier, H. Hermanns, Weak bisimulation for fully probabilistic processes, in: Proc. of CAV'97, in: Springer LNCS, vol. 1254, 1997, pp. 119–130.
[8] C. Baier, On algorithmic verification methods for probabilistic systems, Habilitation Thesis, Univ. Mannheim, 1998.
[9] R.E. Bellman, Dynamic Programming, Princeton University Press, 1957.
[10] M. Bernardo, R. Gorrieri, A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time, Theoretical Computer Science 202 (1998) 1–54.
[11] D. Beauquier, On probabilistic timed automata, Theoretical Computer Science 292 (2003) 65–84.
[12] P. Bouyer, Forward analysis of updatable timed automata, Formal Methods in System Design 24 (2004) 281–320.
[13] F. van Breugel, J. Worrel, Towards quantitative verification of probabilistic systems (extended abstract), in: Proc. of 28th Int. Colloquim on Automata, Languages and Programming, in: Springer LNCS, vol. 2076, 2001, pp. 421–432.
[14] D. Brumley, D. Boneh, Remote timing attacks are practical, The International Journal of Computer and Telecommunications Networking 48 (2005) 701–716.
[15] S. Cattani, R. Segala, Decision algorithm for probabilistic bisimulation, in: Proc. of CONCUR'02, in: Springer LNCS, vol. 2421, 2002, pp. 371–385.
[16] K. Chatzikokolakis, G. Norman, D. Parker, Bisimulation for demonic schedulers, in: Proc. of FOSSACS'09, in: Springer LNCS, vol. 5504, 2009, pp. 318–332.
[17] K. Chatzikokolakis, C. Palamidessi, Probable innocence revisited, Theoretical Computer Science 367 (2006) 123–138.
[18] K. Chatzikokolakis, C. Palamidessi, Making random choices invisible to the scheduler, in: Proc. of CONCUR'07, in: Springer LNCS, vol. 4703, 2007, pp. 42–58.
[19] D. Chaum, The dining cryptographers problem: unconditional sender and recipient untraceability, Journal of Cryptology 1 (1988) 65–75.
[20] K. Cerans, Decidability of bisimulation equivalences for parallel timer processes, in: Proc. of CAV'92, in: Springer LNCS, vol. 663, 1992, pp. 302–315.
[21] P.R. D'Argenio, H. Hermanns, J.-P. Katoen, On generative parallel composition, in: Proc. of PROBMIV'98, in: Elsevier ENTCS, vol. 22, 1998, pp. 30–54.
[22] J. Desharnais, V. Gupta, R. Jagadeesan, P. Panangaden, The metric analogue of weak bisimulation for probabilistic processes, in: Proc. of 17th Symposium on Logic in Computer Science, IEEE CS Press, 2002.
[23] J. Desharnis, V. Gupta, R. Jagadeesan, P. Panangaden, Metrics for labelled Markov processes, Theoretical Computer Science 318 (2004) 323–354.
[24] R. Focardi, R. Gorrieri, A classification of security properties, Journal of Computer Security 3 (1995) 5–33.
[25] P.R. Halmos, Measure Theory, Springer-Verlag, 1950.
[26] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, Symbolic model checking for real-time systems, Information and Computation 111 (1994) 193–244.
[27] H. Hermanns, U. Herzog, V. Mertsiotakis, Stochastic process algebras — between Lotos and Markov chains, Computer Networks and ISDN Systems 30 (1998) 901–924.
[28] H. Hermanns, M. Stegle, Bisimulation algorithms for stochastic process algebras and their BDD-based implementation, in: Proc. of ARTS'99, in: Springer LNCS, vol. 1601, 1999, pp. 244–264.
[29] J. Hillston, A Compositional Approach to Performance Modelling, Cambridge University Press, 1996.
[30] H. Howard, Dynamic Programming and Markov Processes, MIT Press, 1960.
[31] H.E. Jensen, H. Gregersen, Formal design of reliable real time systems, Master's Thesis, Aalborg University, 1995.
[32] H.E. Jensen, Model checking probabilistic real time systems, in: Proc. of the 7th Nordic Work. on Progr. Theory, Institute of Technology, 1996, pp. 247–261.
[33] P.C. Kanellakis, S.A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, Information and Computation 86 (1990) 43–68.
[34] P.C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: Proc. of CRYPTO 1996, in: Springer LNCS, vol. 1109, 1996, pp. 104–113.
[35] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston, Automatic verification of real-time systems with discrete probability distribution, Theoretical Computer Science 282 (2002) 101–150.
[36] M. Kwiatkowska, G. Norman, J. Sproston, Symbolic model checking for probabilistic timed automata, in: Proc. of FORMATS/FTRTFT2004, in: Springer LNCS, vol. 3253, 2004, pp. 293–308.
[37] R. Lanotte, D. Beauquier, A decidable probability logic for timed probabilistic systems, CoRR, cs.LO/0411100, 2004.
[38] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Weak bisimulation for probabilistic timed automata and applications to security, in: Proc. of SEFM'03, IEEE CS Press, 2003, pp. 34–43.
[39] R. Lanotte, A. Maggiolo-Schettini, A. Troina, A classification of time and/or probability dependent security properties, in: Proc. of QAPL'05, in: ENTCS, vol. 153(2), Elsevier, 2005, pp. 177–193.
[40] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Reachability results for timed automata with unbounded data structures, Acta Informatica 47 (2010) 279–311.
[41] R. Lanotte, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Design and verification of long-running transactions in a timed framework, Science of Computer Programming 73 (2008) 76–94.
[42] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Time and probability-based information flow analysis, IEEE Transactions on Software Engineering 36 (2010) 719–734.
[43] K.G. Larsen, A. Skou, Bisimulation through probabilistic testing, Information and Computation 94 (1991) 1–28.
[44] K.G. Larsen, Y. Wang, Time abstracted bisimulation: implicit specifications and decidability, Information and Computation 134 (1997) 75–101.
[45] N.A. Lynch, F.W. Vaandrager, Forward and backward simulations, II: timing-based systems, Information and Computation 128 (1996) 1–25.
[46] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[47] X. Nicollin, J. Sifakis, S. Yovine, From ATP to timed graphs and hybrid systems, Acta Informatica 30 (1993) 181–202.
[48] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM Journal on Computing 16 (1987) 973–989.
[49] A. Philippou, I. Lee, O. Sokolsky, Weak bisimulation for probabilistic systems, in: Proc. CONCUR'00, in: Springer LNCS, vol. 1877, 2000, pp. 334–349.
[50] PRISM Model Checker. Web site: http://www.cs.bham.ac.uk/dxp/prism.

[51] M.K. Reiter, A.D. Rubin, Crowds: anonimity for web transactions, ACM Transactions on Information and System Security 1 (1998) 66–92.
[52] P. Ryan, S. Schneider, Process algebra and non-interference, in: Proc. of CSFW'99, IEEE CS Press, 1999, pp. 214–227.
[53] R. Segala, Modeling and verification of randomized distributed real-time systems, Ph.D. Thesis, MIT, Laboratory for Computer Science, 1995.
[54] A. Sokolova, E.P. de Vink, Probabilistic automata: system types, parallel composition and comparison, in: Validation of Stochastic Systems, in: Springer LNCS, vol. 2925, 2004, pp. 1–43.
[55] J. Sproston, A. Troina, Simulation and bisimulation for probabilistic timed automata, in: Proc. of FORMATS'10, in: Springer LNCS, vol. 6246, 2010, pp. 213–227.
[56] M. Stoelinga, Alea jacta est: verification of probabilistic, real-time and parametric systems, Ph.D. Thesis, University of Nijmegen, the Netherlands, 2002.
[57] Y. Wang, Real-time behaviour of asynchronous agents, in: Proc. of CONCUR'90, in: Springer LNCS, vol. 458, 1990.