Weak Bisimulation for Probabilistic Timed Automata and Applications to Security[*]

Ruggero Lanotte, Andrea Maggiolo-Schettini and Angelo Troina
Dipartimento di Informatica, Università di Pisa

Abstract

We are interested in describing timed systems that exhibit probabilistic behaviors. To this purpose, we define a model of probabilistic timed automata and give a concept of weak bisimulation together with an algorithm to decide it. We use this model for describing and analyzing a probabilistic non-repudiation protocol in a timed setting.

1 Introduction

Timed Automata have been introduced by Alur and Dill [2] 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. Extensions with probability have been proposed (e.g. in [5], [11] and [12]).

In this paper we study weak bisimulation for probabilistic timed automata. As the definition of weak bisimulation requires that a time step is simulated with a sequence of untimed τ steps followed by a time step followed by a sequence of τ steps, we assume a model of timed automata where the elapsing of time is associated with transitions and not with states. With this assumption, a probability is associated with transitions and, therefore, the choice among steps is done probabilistically. It is easy to see that these Probabilistic Timed Automata recognize the same class of languages recognized by Timed Automata.

We prove the decidability of weak bisimulation for Probabilistic Timed Automata. We use the model and the mentioned result to describe and analyze a security problem for a non-repudiation protocol in a timed setting.

[*]This research has been supported by Progetto MURST Metodi Formali per la Sicurezza e il Tempo (MEFISTO).

2 Probabilistic Timed Automata

We assume a set X of variables, called *clocks*. A *clock valuation* over X is a mapping $v : X \to \mathbb{R}^{\geq 0}$ assigning time values to clocks. For a clock valuation v and a time value t, $v + t$ denotes the clock valuation such that $(v + t)(x) = v(x) + t$. Moreover, given a set of clocks $Y \subseteq X$, with $v[Y := 0]$ we denote the valuation that sets each clock in Y to 0, while leaving unchanged the valuations of the other clocks.

Let v_1 and v_2 be two valuations on two disjoint sets of clocks X_1 and X_2; with $v_1 \cup v_2$ we denote the valuation on clocks $X_1 \cup X_2$ such that $v_1 \cup v_2(X) = v_1(x)$ if $x \in X_1$ and $v_1 \cup v_2(X) = v_2(x)$ otherwise.

Given a set of clocks X, the most general set of *clock constraints* over X, denoted $\Phi(X)$, is defined by the following grammar, where ϕ ranges over $\Phi(X)$, $x, y \in X$, $c \in \mathbb{Q}$ and $\sim \in \{<, \leq, =, \neq, >, \geq\}$:

$$\phi ::= x \sim c \,|\, \phi \wedge \phi \,|\, \neg\phi \,|\, \phi \vee \phi \,|\, true$$

We write $v \models \phi$ when *the clock valuation* v *satisfies the clock constraint* ϕ. Formally, $v \models x \sim q$ iff $v(x) \sim q$, $v \models \phi_1 \wedge \phi_2$ iff $v \models \phi_1$ and $v \models \phi_2$, $v \models \neg\phi$ iff $v \not\models \phi$, $v \models \phi_1 \vee \phi_2$ iff $v \models \phi_1$ or $v \models \phi_2$, and $v \models true$.

A *Probabilistic Timed Automaton* is a 6-tuple $A = (\Sigma, X, Q, q_0, \delta, \pi)$, where:

- Σ is a finite alphabet of actions.
- X is a finite subset of clocks.
- Q is a finite set of *states* and $q_0 \in Q$ is the initial state.
- δ is a finite set of *transitions*. $\delta \subseteq Q \times \Phi(X) \times \Sigma \cup \{\tau, \lambda\} \times 2^X \times Q$. The symbol τ represents the silent or internal move, and the symbol λ describes time elapsing. For a state q, we denote with $start(q)$ the set of transitions with q as a source state, i.e. the set $\{(q_1, \phi, a, Y, q_2) \in \delta \,|\, q_1 = q\}$.

- $\pi : \delta \to [0,1]$ is a *probability function*. If $e \in \delta$, then $\pi(e)$ is the probability of performing the transition e. We require that for each state s it holds that $\sum_{e \in start(s)} \pi(e) \in \{0,1\}$.

A *configuration* of A is a pair (q,v) where q is a state of A, and v is a valuation. Given the probabilistic timed automaton A, we call $\mathcal{S}_A$ the set of configurations of A.
There is a *discrete step* from a configuration $s_1 = (q_1, v_1)$ to a configuration $s_2 = (q_2, v_2)$ through action $a \in \Sigma \cup \{\tau\}$, written $s_1 \xrightarrow{a} s_2$, if there is a transition $e = (q_1, \phi, a, Y, q_2) \in \delta$ such that $v_1 \models \phi$, $\pi(e) > 0$ and $v_2 = v_1[Y := 0]$.
There is a *continuous step* from a configuration $s_1 = (q_1, v_1)$ to a configuration $s_2 = (q_2, v_2)$ through time $t \in \mathbb{R}^{\geq 0}$, written $s_1 \xrightarrow{t} s_2$, if there is a transition $e = (q_1, \phi, \lambda, Y, q_2) \in \delta$ such that $v_1 + t \models \phi$, $\pi(e) > 0$ and $v_2 = (v_1 + t)[Y := 0]$.

For configurations $s_1 = (q_1, v_1)$, $s_2 = (q_2, v_2)$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{\geq 0}$, we define with $P(s_1, \alpha, s_2)$ the probability of reaching configuration s_2 from configuration s_1 through a transition labeled with α. Formally we have:

$$P(s_1, \alpha, s_2) = \frac{\sum_{e \in Adm(s_1, \alpha, s_2)} \pi(e)}{\sum_{e \in Adm(s_1)} \pi(e)},$$

where $Adm(s_1, \alpha, s_2)$ is the set
$\{(q_1, \phi, \lambda, Y, q_2) \in \delta \mid v_1 + \alpha \models \phi \ \wedge \ v_2 = (v_1 + \alpha)[Y := 0]\}$ if $\alpha \in \mathbb{R}^{\geq 0}$, the set
$\{(q_1, \phi, \alpha, Y, q_2) \in \delta \mid v_1 \models \phi \ \wedge \ v_2 = v_1[Y := 0]\}$, otherwise, and $Adm(s_1) = \cup_\alpha \cup_{s_2} Adm(s_1, \alpha, s_2)$.

A configuration $s = (q_i, v_i)$ is called *terminal* iff $\sum_{e \in Adm(s)} \pi(e) = 0$; we denote with S_t the set of terminal configurations.

An *execution fragment* starting from s_0 is a finite sequence of steps $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \xrightarrow{\alpha_k} s_k$ such that $s_0, s_1, \ldots, s_k \in \mathcal{S}_A$, $\alpha_1, \alpha_2, \ldots, \alpha_k \in \Sigma \cup \{\tau, \lambda\}$ and $\forall i \in \{1, \ldots, k\}$ $P(s_{i-1}, \alpha_i, s_i) > 0$. We define $last(\sigma) = s_k$ and $|\sigma| = k$. If $|\sigma| = 0$ we put $P(\sigma) = 1$, else, if $|\sigma| = k \geq 1$, we define $P(\sigma) = P(s_0, \alpha_1, s_1) \cdot \ldots \cdot P(s_{k-1}, \alpha_k, s_k)$. The execution fragment σ is called *maximal* iff $last(\sigma) \in S_t$. We denote with $ExecFrag(s)$ the set of execution fragments starting from s.

An *execution* is either a maximal execution fragment or an infinite sequence $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$, where $s_0, s_1 \ldots \in \mathcal{S}_A$, $\alpha_1, \alpha_2, \ldots \in \Sigma \cup \{\tau, \lambda\}$ and $\forall i \geq 1$ $P(s_{i-1}, \alpha_i, s_i) > 0$. We denote with $Exec(s)$ the set of executions starting from s. Finally, let $\sigma \uparrow$ denote the set of executions σ' such that $\sigma \leq_{prefix} \sigma'$, where *prefix* is the usual prefix relation over sequences.

Assuming the basic notions of probability theory (see e.g. [9]) we define the probability space on the executions starting from a given configuration $s \in \mathcal{S}_A$ as follows. Let $\Sigma_F(s)$ be the smallest sigma field on $Exec(S)$ that contains the basic cylinders $\sigma \uparrow$, where $\sigma \in ExecFrag(s)$. The probability measure $Prob$ is the unique measure on $\Sigma_F(s)$ such that $Prob(\sigma \uparrow) = P(\sigma)$.

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

Consider now $Exec(\tau^* \hat{\alpha}, \mathcal{C})$, the set of executions that lead to a configuration in $\mathcal{C}$ via a sequence belonging to the set of sequences $\tau^* \hat{\alpha}$. We define $Exec(s, \tau^* \hat{\alpha}, \mathcal{C}) = Exec(\tau^* \hat{\alpha}, \mathcal{C}) \cap Exec(s)$, where $Exec(s)$ is the set of executions starting from s. Finally, we define the probability $Prob(s, \tau^* \hat{\alpha}, \mathcal{C}) = Prob(Exec(s, \tau^* \hat{\alpha}, \mathcal{C}))$ as in figure 1.

3 Regions

We recall the definition of clock equivalence. Configurations reachable by performing a transition starting from a given state do not depend on probability, and therefore we may use concepts and properties given for Timed Automata.

Let A be a Probabilistic Timed Automaton; with C_A we denote the greatest constant that appears 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, with C_A the largest integer appearing in clock constraints over x;

- 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 $fract(\_)$ is the fractional part);

- for each clock x with $v(x) \leq C_A$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.

As proved in [2], $v \approx v'$ implies that, for any $\phi \in \Phi(X)$ with constants less or equal than 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 [6] and [10].

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

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

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

With $\Psi_C(X)$ we denote the set of clock zones in $\Psi(X)$ that use integer constants in $[-C, C]$.

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

$$Prob(s,\tau^*\hat{\alpha},\mathcal{C}) = \begin{cases} 1 & if\ \alpha=\tau \wedge s\in\mathcal{C} \\ \sum_{q\in S} Prob(s,\tau,q)\cdot Prob(q,\tau^*,\mathcal{C}) & if\ \alpha=\tau\wedge s\notin\mathcal{C} \\ \sum_{q\in S} Prob(s,\tau,q)\cdot Prob(q,\tau^*\alpha,\mathcal{C}) + Prob(s,\alpha,\mathcal{C}) & if\ \alpha\neq\tau \end{cases}$$

Figure 1. Definition of $Prob(s,\tau^*\hat{\alpha},\mathcal{C})$

The following proposition states that the set of configurations reachable by performing transitions starting from a set of configurations expressed by a region is a region.

Proposition 3.1 *If (q_1,ψ_1) is a region of A and e is a transition, then the set of reachable configurations by performing e and starting from (q_1,ψ_1), is a region.*

Now, it is obvious that the set of reachable configurations can be calculated by using the proposition above, but it is also obvious that the set of regions is not finite. Therefore we need an approximation.

If (q,ψ) is a region of A, we denote with $Ap_A(q,\psi)$ the set $\{(q,v) \mid [v]\cap\psi \neq false\}$. The following proposition, proved in [6], states that Ap_A returns a region.

Proposition 3.2 (Approximation) *$Ap_A(q,\psi)$ is a region of A with constants belonging to the interval $[-C_A, C_A]$.*

Let (q,ψ) be a region of A and e be a transition with q as source state. With $post((q,\psi),e)$ we denote the region calculated as in propositions 3.1 by approximating (q,ψ) by $Ap_A(q,\psi)$. Now, it is obvious that $post((q,\psi),e)$ is a region of A with constants in the interval $[-C_A,C_A]$. Therefore, these regions are finitely many.

The following theorem, proved in [6], states the correctness of the operator $post$.

Theorem 3.3 *Let R' be the set of regions reachable with a transition e starting in the set of regions R. Then $R' \subseteq post(Ap(R),e) \subseteq Ap_A(R')$.*

4 Weak bisimulation

In this section we define weak bisimulation for probabilistic timed automata. In order to abstract from τ (or internal) moves, Milner [14] introduces the notion of observable step, which consists of a single *visible* action α preceded and followed by an arbitrary number (including zero) of internal moves. Such moves are described by a *weak* transition relation $\Longrightarrow$, defined as $\overset{\alpha}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^*$, where $\longrightarrow$ is the classical strong relation, and $\overset{\tau}{\Longrightarrow} = (\xrightarrow{\tau})^*$. It is worth noting that with such a definition a weak internal transition $\overset{\tau}{\Longrightarrow}$ is possible even without performing any internal action. For the definition of weak bisimulation in the fully probabilistic setting, Bayer and Hermann [3] replace Milner's weak internal transitions $s \overset{\tau}{\Longrightarrow} t$ by the probability $Prob(s,\tau^*,t)$ of reaching configuration t from s via internal actions. Similarly, for visible actions α, they define $\overset{\alpha}{\Longrightarrow}$ by the probability $Prob(s,\tau^*\alpha,t)$.

The probabilistic model we have chosen for probabilistic timed automata is that of fully probabilistic systems. In such a model, as demonstrated by Bayer and Hermanss in [3], the two relations of weak bisimulation equivalence and branching bisimulation equivalence do coincide. Relying on this result, we use branching bisimulation in order to decide weak bisimulation.

Definition 4.1 *Let $A=(\Sigma,X,Q,q_0,\delta,\pi)$ be a Probabilistic Timed Automaton. A branching bisimulation on A is an equivalence relation $\mathcal{R}$ on $\mathcal{S}_A$ such that for all $(s,s')\in\mathcal{R}$, $\mathcal{C}\in\mathcal{S}_A/\mathcal{R}$:*

$$Prob(s,\tau^*\alpha,\mathcal{C}) = Prob(s',\tau^*\alpha,\mathcal{C}) \qquad \forall\alpha\in\Sigma\cup\tau\cup\mathbb{R}^{\geq 0}.$$

Two configurations s, s' are called branching bisimilar on A (denoted $s\approx s'$) iff $(s,s')\in\mathcal{R}$ for some branching bisimulation $\mathcal{R}$.
Two Probabilistic Timed Automata $A=(\Sigma,X,Q,q_0,\delta,\pi)$ and $A'=(\Sigma,X',Q',q_0',\delta',\pi')$ such that $Q\cap Q'=\emptyset$ and $X\cap X'=\emptyset$ are called branching bisimilar (denoted by $A\approx A'$) if, given the Probabilistic Timed Automaton $\hat{A}=(\Sigma,X\cup X',Q\cup Q',q_0,\delta\cup\delta',\hat{\pi})$, with

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

it holds $(q_0,v_0)\approx(q_0',v_0)$, where for each $x\in X\cup X'$ it holds that $v_0(x)=0$.

Note that the function $\hat{\pi}$ is well defined since $Q\cap Q'=\emptyset$ implies $\delta\cap\delta'=\emptyset$. We shall see that the choice of the initial state of $\hat{A}$ is indifferent for the computation of the branching bisimulation equivalence classes. We have chosen q_0, but we could choose q_0' as well.

5 Decidability of bisimulation

In this section we develop an algorithm that computes the classes of the branching bisimulation equivalence and decides if two configurations are branching bisimilar by checking that they are in the same class. To do this we have to check the condition of definition 4.1.

We extend the algorithm given for untimed systems.

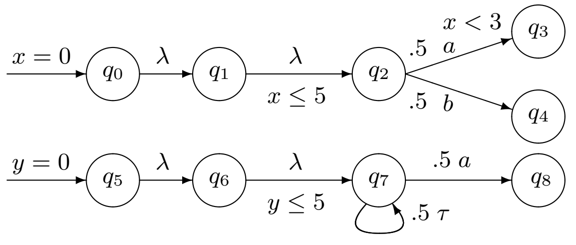

Figure 2. An example

Example 5.1 *Consider the automata of figure 2. In the untimed version the probability to reach q_4 from the state q_2 is 0.5. Now, in the timed version, we observe that in state q_2, when clock x has value smaller then 3, the automaton may execute both transitions with probability $0,5$. Otherwise, if clock x has value greater than 3, the transition labeled with a cannot be executed, and so 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, one may consider to use the algorithm for the untimed version on the region graphs of the two automata, i.e. the graph of regions resulting by applying the successor operator. 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 $y \geq 0$. 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 so they are equal for every time if and only if $x = y$. This means that we cannot consider the clocks separately, but we must have formulae on all the pairs of states.

Since we have to check also the bisimilarity for states of the same automaton, as an example q_0 and q_1, we have to consider formulae that express conditions on the value of clocks at state q_0 together with the value of clocks at state q_1. As an example $x^1 = x^2$ means that the values of clocks at state q_0 are the same of those at state q_1.

The algorithm splits a class if in the class there exist two configurations starting from which there are two different probabilities of reaching a certain class by performing $\tau^*\alpha$.

The algorithm takes a class and calculates these probabilities by solving a set of pairs of systems of equalities (rather than a pair of systems of equalities, as in the untimed case). The algorithm terminates when no class can be splitted further.

5.1 The Algorithm

We define the labeling of the clocks in X by setting $X^l = \{x^l \mid x \in X\}$, for any $l \in \{1,2\}$. When we consider two configurations (q,v) and (q',v') of A the set of clocks X^1 represents the clocks of q in X, and X^2 those of q'.

Let $A = (\Sigma, X, Q, q_0, \delta, \pi)$ be a fixed Probabilistic Timed Automaton; with A^l, for $l \in \{1,2\}$, we represent the fixed Probabilistic Timed Automaton A where we rename X with X^l. We use δ^l, $start^l(q)$ and π^l to denote the set of transitions, the set of transitions starting from q and the probability function of A^l, respectively.

If $Q = \{q_1, \ldots, q_n\}$, then a class is a function $g : [1,n]^2 \rightarrow \Psi_{C_A}(X^1 \cup X^2)$ Now, classes are finitely many since the number of clock zones are finitely many. A class g represents the set of pairs of configurations $((q_h, v_h), (q_k, v_k))$ of A such that $v_h \cup v_k \models g(h,k)$. Hence, with $[g]$ we denote the set of triples (q_h, q_k, ψ) such that $\psi \Rightarrow g(h,k)$ representing the configuration (q_h, v) and (q_k, v') with $v \cup v' \models \psi$.

Example 5.2 *The function g such that $g(0,2) \equiv x^1 \leq x^2$ and $false$ otherwise, is a class of example 5.1. This represents the set of pairs of configurations $((q_0, v), (q_2, v'))$, such that $v(x) \leq v'(x)$.*

Two classes g_1 and g_2 are disjoint if and only if for each h,k it holds that $g_1(h,k) \wedge g_2(h,k) \equiv false$. A set of disjoint classes $\mathcal{G}$ represents the relation $(q_h, v) \approx_{\mathcal{G}} (q_k, v')$ such that $v \cup v' \models g(h,k)$, for some $g \in \mathcal{G}$. With $\bar{g}_{\mathcal{G}}$ and g_{true} we represent the class such that $\bar{g}_{\mathcal{G}}(h,k) = \bigwedge_{g \in \mathcal{G}} \neg g(h,k)$ and $g_{true}(h,k) = true$, respectively. The class $\bar{g}_{\mathcal{G}}$ represents the configurations that are not bisimilar, i.e. if $v \cup v' \models \bar{g}_{\mathcal{G}}(h,k)$, then $(q_h, v) \not\approx_{\mathcal{G}} (q_k, v')$. The class g_{true} we represent the biggest class. Moreover, if ψ is a clock zone in $\Psi(X^1 \cup X^2)$ with $g \cap \psi$ we denote the class such that for each h,k it holds that $(g \cap \psi)(h,k) = g(h,k) \wedge \psi$.

We can extend the definition of $post$ over triples. If e_1 and e_2 are two transitions, then with $post(q_h, q_k, \psi, e_1, e_2, g)$ we denote the set of triples in $[g]$ reachable from (q_h, q_k, ψ) by using the transition e_1 and e_2 synchronously. More precisely, if $a_1 \neq a_2$ or $e_1 \notin start^1(q_h)$ or $e_2 \notin start^2(q_k)$, then $post(q_h, q_k, \psi, e_1, e_2, g)$ is the triple $(q_h, q_k, false)$, and, otherwise, it is the triple $(q_r, q_w, \psi' \wedge g(r,w))$, where $post((q,\psi), e') = (q', \psi')$, for some states q, q', and $e' = (q, \phi_1 \wedge \phi_2, a_1, Y_1 \cup Y_2, q'))$, if $e_1 = (q_h, \phi_1, a_1, Y_1, q_r)$ and $e_2 = (q_k, \phi_2, a_2, Y_2, q_w)$. Moreover, with e_T we denote the set of *useless* transitions $\cup_{q \in Q}(q, true, \tau, \emptyset, q)$. We will use this set to describe a step of only one of the two components.

A set of triples R is empty if and only if each $(q_h, q_k, \psi) \in R$ is such that $\psi \equiv false$. A set of transitions E is enabled in R if and only if for each $e \in E$ there exists $(q_l, q_k, \psi) \in R$, such that if $e \in \delta^1$, then $post((q_l, q_k, \psi), e, e_T, g_{true})$ is not empty, and, if $e \in \delta^2$, then $post((q_l, q_k, \psi), e_T, e, g_{true})$ is not empty.

With $\mathcal{F}$ we denote the set of functions f such that for each $h \in [1, n]$, $l \in [1, 2]$ it holds that $f(h, l) \subseteq 2^{(2^{start^l(q_h)} \times 2^{start^l(q_h)})}$ and if $(E, E') \in f(h, l)$, then $E \cap E' = \emptyset$. The pair $(E, E') \in f(h, l)$ represents that the transitions $E \cup E'$ are the only ones enabled, and, if a transition in E is executed, then it takes configurations with state h of component l to configurations in the same class, and those in E' take configurations with state h of component l to configurations in a different class. Since a state can be crossed several times we use $2^{2^{start^l(q_h)}}$ instead of $2^{start^l(q_h)}$. We will write $f(R, l)$ to denote the set $\cup_{(q_h, q_k, \psi) \in R} \{E \cup E' \mid (E, E') \in f(h, l)\}$.

Let $f \in \mathcal{F}$ and $E_{h,k} = start(q_h) \cup start(q_k)$. We define the formula $\psi_f =$

$$\exists t \geq 0. \bigwedge_{h,k \in [1,n]} g(h,k) \Rightarrow \bigwedge_{l=1,2} \bigvee_{(E,E') \in f(h,l)} \left(\bigwedge_{e \in E \cup E'} \phi_e\right) \wedge \left(\bigwedge_{e \in E_{h,k} \setminus E \cup E'} \neg\phi_e\right)$$

where if $e = (q_r, \phi, \alpha, Y, q_w)$, and, $\psi' = \bigvee_{j \in [1,n]} g(w, j)$ if $e \in \delta^1$ and $\bigvee_{j \in [1,n]} g(j, w)$ otherwise, then

- if $e \in E$ and $\psi'' = \phi \wedge (\exists Y. Y = 0 \wedge \psi')$, then $\phi_e \equiv \psi''[X^l := X^l + t]$, if $\alpha = \lambda$, and $\phi_e \equiv \psi''$ otherwise. Namely, the transition e is enabled and the reachable configurations are in g. If e is λ-labeled, then we must consider the time elapsing represented by the variable t.
- if $e \in E'$ and $\psi'' = \phi \wedge (\exists Y. Y = 0 \wedge \neg\psi')$, then $\phi_e \equiv \psi'[X^l := X^l + t]$, if $\alpha = \lambda$, and $\phi_e \equiv \psi'$ otherwise. Namely, the transition e is enabled and the reachable configurations are not in g.

The formula ψ_f gives the weakest precondition such that in a configuration represented by g one can choose to perform, at a certain instant of time, only transitions expressed by f. This is necessary to normalize probabilities when calculating the probability of performing a sequence expressed by $\tau^*\alpha$. It is obvious that, by means of the quantifier elimination showed in [10], the formula ψ_f is in $\Psi(X^1 \cup X^2)$. We note also that $[\![g \cap \psi_f]\!]$ contains the triple refined with the condition ψ_f that ensures that the respective subregions surely perform only transitions in f.

Example 5.3 *Let g be a class such that $g(2, 8) \equiv x^1 = y^2$ and false otherwise, $f(2, 1) = \{(\emptyset, E)\}$ and $f(8, 2) = \{(E', E'')\}$ where E are the transitions from q_2 to $\{q_3, q_4\}$, E' is the τ-labeled transition and E'' is the transition from q_7 to q_8. We have that $\psi_f \equiv \exists t \geq 0.(x^1 = y^2) \Rightarrow (x^1 \geq 3 \wedge y^2 + t \leq 5 \wedge x^1 = y^2)$.*

If g is a class, $f \in \mathcal{F}$ and $R \subseteq [g_{true}]$, then with $R_1(f, g)$ we denote the fixpoint of the computation such that $R_0 = R$ and R_k is the set R_{k-1} joint with the set

$$\bigcup_{(E,E') \in f(R_{k-1},1)} \{post(R_{k-1}, e, e_T, g) \mid e \in E \ is \ \tau-labeled\}.$$

It is obvious that the fixpoint is reached in a finite sequence of steps since the set of clock zones are finitely many. The set $R_1(f, g)$ is the set of triples reachable by performing a sequence of τ transitions of A^1 while remaining in g. Analogously we can define $R_2(f, g)$.

If $\mathcal{G}$ is a set of disjoint classes, $g_1 \in \mathcal{G}$, $g_2 \in \mathcal{G} \cup \{\bar{g}_{\mathcal{G}}\}$, q_i, q_j are two states and $\alpha \in \Sigma \cup \{\tau, \lambda\}$, then we define the function $split(q_i, q_j, \alpha, g_1, g_2)$ that returns a function $f \in \mathcal{F}$ such that the probability of reaching the class g_2 from configurations (q_i, v) and (q_j, v'), with $v \cup v' \models \psi_f$ and taking transitions in f, are different, if $g_2 \neq \bar{g}_{\mathcal{G}}$, and are different from 0 otherwise. Now we show how this function computes f.

We can calculate the probability of performing $\tau^*\alpha$ by solving a set of systems of equalities. For the untimed case, the probability of performing a sequence in $\tau^*\alpha$ can be calculated by solving a system of equalities by following the definition of figure 1. Now, in the framework of timed automata, the transitions enabled at a certain instant depend on the values of clocks. Therefore we must consider all the possible behaviors, which means to consider the possible set of transitions performable at a certain instant, namely each $f \in \mathcal{F}$. We describe now how is the system of equalities that depends on f.

Let $f \in \mathcal{F}$; we use the variable y and the set of variables y_R where $\emptyset \subset R \subseteq [g_1 \cap \psi_f]$. The variable y_R represents the probability of the triple R in $[g_1]$ of reaching $[g_2]$ by sequences in $\tau^*\alpha$. The variable y represents a generic configuration of g_2.

We consider the system of equalities $I_{i,f}$ composed by the equality $y = 1$ and the set of equalities $y_R = p_1 \cdot y_1 + \cdots + p_n \cdot y_n$ where there exists $(E, E') \in f(R, 1)$ enabled in R such that, if $E \cup E' = \{e_1, \ldots, e_m\}$, then for each $l \in \{1, \ldots, m\}$ it holds that:

- if $e_l \in E$ is τ-labeled and $R' = post(R_2(f, g_1), e_l, \{e_T\} \cup f(R_2(f, g_1)), g_1)$ is not empty, then $y_l = y_{R'}$ and $p_l = \frac{\pi^1(e_l)}{\sum_{e(\in E \cup E') \cap \delta^1} \pi^1(e)}$.

  Namely, we have a sequence of τ steps while remaining in the class $g \cap \psi_f$ of A^2, followed by one of A^1 (possibly synchronized with one of A^2).

- if $e_l \in E'$ is α-labeled and $post(R, e_l, f(R,2), g_2)$ is not empty, then $y_l = y$ and $p_l = \frac{\pi^1(e_l)}{\sum_{e \in (E \cup E') \cap \delta^1} \pi^1(e)}$.

  Namely the two configurations, by synchronizing on α, fall into g_2.

- Otherwise $p_l = 0$ and $y_l = y$.

Analogously we can define $I_{j,f}$.

The following proposition states the relationship of the set of equalities defined above with the probabilities of performing $\tau^*\alpha$.

Proposition 5.4 *Let (q_i, v), (q_j, v') such that $v \cup v' \models g_1(i,j) \cap \psi_f$. $Prob((q_i, v), \tau^*\alpha, g_2)$ and $Prob((q_j, v'), \tau^*\alpha, g_2)$ are equal to the solutions of the variable y_{R_0} of the systems of equalities $I_{i,f}$ and $I_{j,f}$, respectively, where $R_0 = \{(q_i, q_j, g(i,j) \wedge \psi_f)\}$.*

If p_1 and p_2 are the solutions of the variable y_{R_0} of the systems of equalities $I_{i,f}$ and $I_{j,f}$, respectively, where $R_0 = \{(q_i, q_j, g(i,j) \wedge \psi_f)\}$, then, by proposition 5.4, we have that f is a *cut* if and only if $p_1 \neq p_2$ if $g_2 \neq \bar{g}_{\mathcal{G}}$, and, $p_1 \neq 0 \wedge p_2 \neq 0$ otherwise.

Example 5.5 *Let us consider the class g and the function f of example 5.3. We want to calculate the probability of reaching the class g' such that $g'(3,4) \equiv g'(3,8) \equiv true$ with the symbol a. We must solve the systems:*

$$\begin{cases} y_R = (0.5) \cdot y \\ y = 1 \end{cases} \qquad \begin{cases} y_R = (0.5) \cdot y_R + (0.5) \cdot y \\ y = 1 \end{cases}$$

where $R = \{(q_2, q_7, x^2 = y^7 \wedge x^2 < 3)\}$. The first system of equalities expresses the fact that from q_2 we can perform a a with probability equal to 0.5, and the second one expresses the fact that from q_7 we can perform a a with probability equal to 1. Therefore we have a cut.

If there exists no cut, then the function $split$ returns a special symbol $\bar{f}$ representing no function, otherwise it returns a cut f. Now, if the function f returned is not $\bar{f}$, then it determines the following *refinement* on the class g_1: if $\{S^1, \ldots, S^m\}$ is a partition of $\{1, \ldots, n\}$ such that for each $h \in S^r$ and $k \in S^t$ it holds that f is not a cut for h and k if and only if $r = t$, then the class g_1 is refined in the union of classes $\{g^0, g^1, \ldots, g^m\}$ such that $g^0 = g_1 \cap \neg\psi_f$ and for each $l \in [1, m]$ and $h, k \in [1, n]$

$$g^l(h,k) = \begin{cases} false & \text{if } \{h,k\} \not\subseteq S^l \\ g_1(h,k) \wedge \psi_f & \text{otherwise.} \end{cases}$$

Example 5.6 *The cut f of example 5.5 generates the refinement of the class g in the classes g^0 such that $g^0(2,7) \equiv x^1 = y^2 \wedge x^1 \geq 3$ and false otherwise.*

We call $\bar{\mathcal{G}}$ the set of classes resulting by refining the classes by using the function returned by $split$ until we reach the fixpoint starting from the class g_{true}. Therefore we have the following theorem that implies the decidability of weak bisimulation for probabilistic timed automata.

Theorem 5.7 *The set $\bar{\mathcal{G}}$ can be computed with a finite number of applications of the refinement induced by the function $split$ and $(q_i, v_i) \approx (q_j, v_j)$ if and only $(q_i, v_i) \approx_{\bar{\mathcal{G}}} (q_j, v_j)$.*

6 A Case Study: Probabilistic Non Repudiation in a Timed Setting

In this section, as a case study, we model and analyze a non-repudiation protocol that employs a probabilistic algorithm to achieve a fairness property. In particular, we extend the case study presented in [1] to a timed setting.

6.1 A Probabilistic Non-Repudiation Protocol

We consider a protocol that guarantees a non-repudiation service with a certain probability without resorting to a trusted third party [13]. In particular, the probabilistic protocol is fair up to a given tolerance ε decided by the originator. Assume that an authentication phase precedes the protocol. We denote by $Sign_E(M)$ the encryption of message M under the private key of the entity E and with $\{M\}_K$ the encryption of M under the key K. Finally, we use t to denote a time stamp. The protocol can be described as follows (with the notation $R \rightarrow O : Msg$ we denote a message Msg sent by R and received by O):

1.	$R \rightarrow O:$	$Sign_R(request, R, O, t)$	
2.	$O \rightarrow R:$	$Sign_O(\{M\}_K, O, R, t)$	$(= M_1)$
3.	$R \rightarrow O:$	$Sign_R(ack_1)$	
4.			
$a._{1-p}$	$O \rightarrow R:$	$Sign_R(M_r, R, O, t)$	$(= M_i)$
	$R \rightarrow O:$	$Sign_R(ack_i)$	
		goto step 4	
$b._p$	$O \rightarrow R:$	$Sign_R(K, R, O, t)$	$(= M_n)$
5.	$R \rightarrow O:$	$Sign_R(ack_n)$	

The recipient R starts the protocol by sending a signed, timestamped request to the originator O. O sends to R the requested message M ciphered under the key K, and waits for the ack from R (ack_i represents the acknowledgment related to message M_i). At step 4 the originator makes a probabilistic choice according to $p = \varepsilon$. At step $4a$ (taken with probability $1 - p$) O sends to R a random message M_r, receives the ack and returns at step 4, while at step $4b$ (taken with probability p) O sends to R the key K necessary to decrypt the message $\{M\}_K$. Upon reception of the last ack (ack_n), related to the message containing the key K, the originator terminates the protocol correctly.

Intuitively, the non-repudiation of origin is guaranteed by the messages M_1 and M_n (signed with the private key of O), while the non repudiation of receipt is given by the last message $Sign_R(ack_n)$. If the protocol terminates after the delivery of the last ack, both parties obtain their expected information, and the protocol is fair. If the protocol terminates before sending the message containing the key K, then neither the originator nor the recipient obtains any valuable information, thus preserving fairness. A strategy for a dishonest recipient consists in guessing the last message containing the key K, verifying if a received message contains the needed key and then blocking the transmission of the last ack. Therefore, the key to success of the protocol is the immediacy in sending back the ack messages. The originator decides a deadline for the reception of each ack, after which, if the ack is not received, the protocol is stopped. Obviously, the cryptosystem must be adequately chosen, in such a way that the time needed to verify a key, by deciphering the message, has to be too long with respect to the transmission time of an ack message. Anyway, a malicious recipient can try to randomly guess the message containing the key K, and in this case the probability for the recipient of guessing the last message depends on the parameter p chosen by the originator.

6.2 Security Analysis of the Protocol

In this section we describe the parties of the protocol through the model of probabilistic timed automata. Since we have to manage communication between agents, we distinguish input and output actions. Given an alphabet of action types Σ_{Type} we define the set of input actions as $\Sigma_I = \{a \mid a \in \Sigma_{Type}\}$ and the set of output actions as $\Sigma_O = \{\overline{a} \mid a \in \Sigma_{Type}\}$. Finally, we define the alphabet of actions as $\Sigma = \Sigma_O \cup \Sigma_I$. Moreover, we assume that output actions behave as generative actions [8], while input actions behave as reactive actions. Generative action executable in a state will be executed according to their probability distribution. In a state, the choice between executing a generative action or a reactive one depends on the external environment. If this provides a generative action with which a reactive action may synchronize, the reactive action will be executed with probability 1. There is still the possibility that more actions may react to the environment, and the choice among them will be due according to a probability distribution (in our case study this does not happen).

We consider τ and λ as generative actions.

We start with introducing the probabilistic timed automata modeling an originator and a recipient behaving correctly. The originator (Fig. 3) is always ready to start a communication by accepting a request, sending the first message containing M encrypted with K (action $\overline{firstmessage}$) and receiving the first ack. Then, in state

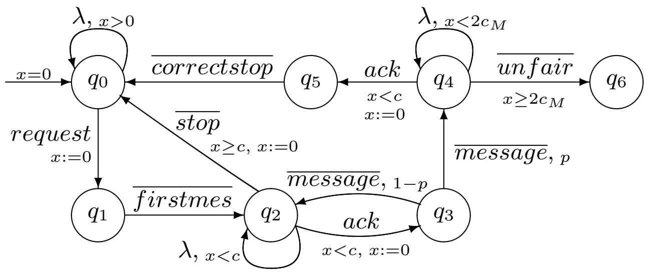

Figure 3. Representation of $Orig$

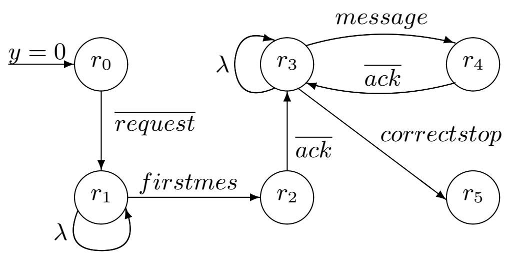

Figure 4. Representation of $HRecip$

q_3, with probability $1-p$, it sends a random message reaching state q_2 and, with probability p, sends the last message containing K and reaches state q_4. We do not model value passing, so we simply call all these actions $\overline{message}$. In state q_2 the reception of the ack message is modeled by the input action ack, while the expiration of the deadline (represented by the constant c) is modeled by the action $\overline{stop}$ executed when the clock x assumes a value greater than c. The fair termination of the protocol is reached when the originator receives the last ack and performs the action $\overline{correctstop}$. The protocol terminates in an unfair way if and only if the originator does not receive the ack related to the message containing K, and in such a case it executes the action $\overline{unfair}$. The constant c_M used in state q_4 represents an estimation of the maximum transmission delay of a message. In particular, it is reasonable to assume that a message sent will always arrive at destination in time c_M.

In Fig. 4, we show the automaton representing a recipient that behaves correctly. The recipient starts the protocol by sending a request, receives the first message, sends the first ack and reaches state r_3, from where, whenever it receives a message, it sends an ack back. The protocol terminates when the input action $correctstop$ is executed. As in the model for the originator, we represent the elapsing of time for transmission delays through λ transitions. For simplicity we do not put conditions on such kind of transitions in the states r_1 and r_2.

In order to verify a *Non-Interference* security property [7] we have to single out the high level actions and the low level ones. It is natural to consider the action $\overline{unfair}$ (executed when the protocol terminates in an incorrect way) as the unique low level action of the system. Intuitively, a low user that observes the protocol run should never see the execution of the action $\overline{unfair}$.

Given two probabilistic timed automata R and Q, we define the parallel composition of R and Q, denoted $R||Q$. The set of states of $R||Q$ is given by the cartesian product of the states of the two automata R and Q. Given a state (r, q) of $R||Q$, the set of transitions starting from (r, q) is obtained by the following rules:

- If from state r the automaton R can perform a generative action $\overline{\alpha}$ leading to r' with probability p, and Q cannot perform either a reactive action α or any generative action in state q, then $R||Q$ performs a generative action $\overline{\alpha}$ with probability p and reaches state (r', q).

- If from state r the automaton R can perform a generative action $\overline{\alpha}$ leading to r' with probability p, and Q can perform a reactive action α leading to state q' and it cannot execute any generative action, then R and Q synchronize and $R||Q$ performs a generative action $\overline{\alpha}$ with probability p and reaches state (r', q').

- If R can perform a generative action $\overline{\alpha}$ with probability p and Q can perform a generative action $\overline{\alpha'}$ with probability p', then $R||Q$ executes either the action $\overline{\alpha}$ with probability $1/2 \cdot p$ or the action $\overline{\alpha'}$ with probability $1/2 \cdot p'$ which synchronizes, in both cases, with a reactive action of the same type, if the other automaton can perform it.

Any rule has a symmetric one.

The automata modeling originator and recipient of our case study can synchronize through actions with type in $\{request, firstmes, ack, message, correctstop\}$.

Now we have to formalize the non-repudiation property to be checked. The hostile environment is represented by the recipient that tries to obtain its expected information without sending the last ack. According to this we check bisimulation equivalence between the model where both parties behave correctly and the model involving a malicious recipient. Now we can consider the protocol to be secure if the system $Orig||Recip$ is bisimilar to $Orig||HRecip$ for each possible malicious recipient $Recip$. Formally, the protocol satisfies the non-repudiation property if and only if:

$$Orig||Recip \approx Orig||HRecip \qquad \forall Recip.$$

We observe that the protocol does not satisfy the security condition. In particular, if both participants behave correctly, the unfair behavior cannot be executed; instead, it is possible to find a malicious recipient that receives the expected information and denies sending the final ack.

In Fig. 5 we show the automaton representing a malicious recipient that maximizes the probability of guessing

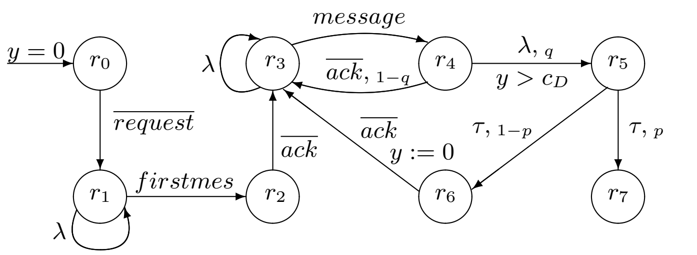

Figure 5. Representation of $MRecip$

the last message of the protocol (we assume that it knows the probability distribution chosen by the originator). It follows a Bernoulli distribution with parameter q to decide either to send the ack message or to try to compute M by employing the last received message (see state r_4). We assume that the time necessary to decipher the message is greater than the constant c_D. Note that if $c > c_D$ the recipient can send an ack even after failing to decipher the message (see state r_6). So the originator should take care of the deadline c chosen. State r_7 represents, instead, the state reached by the malicious recipient when correctly guessing the last message. Since the probability that a message received from $Orig$ is the last one, containing the needed key, is equal to p, we assume that in state r_5 the malicious recipient goes to the winning state r_7 with probability p and to state r_6, from which it tries to send the ack back, with probability $1 - p$. Such a probabilistic choice represents the frequency that models the behavior of the malicious recipient in state r_5, even if it already knows, when reaching state r_5 trying to decipher the message, if the received key is the right one.

The probability of executing the action $\overline{unfair}$ for the system $Orig||HRecip$ is equal to 0, while the probability of executing it for the system $Orig||MRecip$ is (as $c < c_D$):

$$z = p \cdot q \cdot \sum_{i=0}^{\infty} ((1-p) \cdot (1-q))^i = \frac{p \cdot q}{1 - (1-p) \cdot (1-q)}.$$

Given $0 < p < 1$ chosen by the originator and $0 < q < 1$, the maximum value for z is p, obtained by taking $q = 1$. The recipient model, which optimizes the probability of violating the fairness condition, is obtained by removing the transition labeled with $\overline{ack}$ from state r_4 to state r_3.

The consideration above gives as a result that the systems $Orig||MRecip$ and $Orig||HRecip$ are not bisimilar according to definition 4.1. Moreover, since the probability of unfairness depends on a parameter chosen by the originator, we may resort to the definition of a bisimilarity up to a given tolerance ε. Such a definition is easy to give. We only need to relax the equality of definition 4.1 with the formula:

$$|Prob(s, \tau^*\alpha, C) - Prob(s', \tau^*\alpha, C)| < \varepsilon$$

and then make some adjustment to the algorithm. Given such a definition, the two systems will result to be bisimilar up to ε in the case $\varepsilon > p$.

7 Conclusion

We have considered a model of Timed Probabilistic Automata. We have presented a notion of weak bisimulation in order to compare them and an algorithm that permits to decide it. Finally, we have applied such a notion in the context of security analysis by modeling an interesting protocol where both time and probability play a role.

References

[1] Aldini, A., Gorrieri, R.: *Security Analysis of a Probabilistic Non-repudiation Protocol*. In Proc. of PAPM-PROBMIV '02, Springer LNCS 2399, 17–36, 2002.

[2] Alur, R., Dill, D. L.: *A theory of timed automata*. Theoretical Computer Science **126**, 183–235, 1994.

[3] Baier, C., Hermanns, H.: *Weak Bisimulation for Fully Probabilistic Processes*. Theoretical Computer Science **126**, 183–235, 1994.

[4] Baier, C: *On Algorithmic Verification methods for Probabilistic Systems*. Habilitation thesis, Univ. Mannheim, 1998.

[5] Beauquier, D.: *On Probabilistic Timed Automata* Theoretical Computer Science **292**, 2003, 65-84.

[6] Bouyer, P.: *Timed Automata May Cause Some Troubles*. BRICS RS-02-35.

[7] Goguen, J. A., Meseguer, J.: *Security Policy and Security Models*. In Proc. of IEEE Symp. on Security and Privacy, IEEE CS Press, 11–20, 1982.

[8] van Glabbeek, R.J, Smolka, S.A., Steffen, R.: *Reactive, Generative and Stratified Models of Probabilistic Processes*. Information and Computation, **121**, 1995, 59–80.

[9] Halmos, P. R.: *Measure Theory*. Springer-Verlag, 1950.

[10] Henzinger, T. A., Nicollin, X., Sifakis, J., Yovine, S.: *Symbolic Model Checking for Real-time Systems*. Information and Computation **111** (1994), 193-244.

[11] Kwiatkowska, M, Norman, G, Segala, R, Sproston, J.: *Automatic Verification of Real-time Systems with Discrete Probability Distribution*. ARTS'99, LNCS **1601**, 1999, 75–95.

[12] Kwiatkowska, M, Norman, R, Sproston, J.: *Symbolic Model Checking of Probabilistic Timed Automata Using Backwards Reachability*. Tech. rep. CSR-00-01, University of Birmingham.

[13] Markowitch, O., Roggeman, Y.: *Probabilistic Non-Repudiation without Trusted Third Party*. 2nd Conference on Security in Communication Network, 1999.

[14] Milner, R.: *Communication and Concurrency*. Prentice Hall, 1989.

Appendix

Definition 7.1 *A set of triples $R \subseteq [g_{true}]$ is* 1-defined by h *(resp.* 2-defined by h*) if and only if for each $(q_l, q_r, \psi) \in R$ it holds that $l = h$ (resp. $r = h$).*

Proof of Proposition 5.4
It is sufficient to prove that from every configuration reachable from (q_i, v) and (q_j, v') while remaining in $g_1 \cap \psi_f$, only the transition expressed by f can be taken. If this holds, then the thesis is a consequence of the proof of the untimed case (see [4]).

First of all we note that, by definition of 1-definedness in h, if R is 1-defined by h, then $R(f_2, g_1)$ is 1-defined by h. Moreover, also $post(R_2(f, g_1), e_l, \{e_T\} \cup f(R_2(f, g_1)), g_1)$ and $post(R, e_l, f(R, 2), g_2)$ are 1-defined by k, for some k.

Since we consider as starting variable y_{R_0} with $R_0 = (q_i, q_j, g_1(i,j) \wedge \psi_f)$, which is obviously 1-defined, then, by induction, each variable y_R used in $I_{i,f}$ is 1-defined. Moreover, by definition of post and by theorem 3.3, for configuration (q_h, v_h) and (q_k, v_k) reached while remaining in $g_1 \cap \psi_f$, there exists a variable y_R such that $v_h \cup v_k \models \psi$, for some (q_h, q_k, ψ) is in R.

Therefore, since R is 1-defined and by definition of ψ_f, the valuation $v_h \cup v_k$ satisfies

$$\bigwedge_{l=1,2} \bigvee_{(E,E') \in f(h,l)} \left(\bigwedge_{e \in E \cup E'} \phi_e \right) \wedge \left(\bigwedge_{e \in E_{h,k} \setminus E \cup E'} \neg \phi_e \right).$$

So there exists one and only one pair (E, E') in $f(h, 1)$ enabled in the configuration (q_h, v_h). This implies that the probability to take a transition $e \in E \cup E'$ is equal to $\frac{\pi^1(e)}{\sum_{e \in (E \cup E') \cap \delta^i} \pi^1(e)}$.

Therefore, by induction, we have the thesis. Similarly we can prove the thesis for the second component.

□

Definition 7.2 *Let $A = (\Sigma, X, Q, q_0, \delta, \pi)$ be a probabilistic timed automaton. We define inductively equivalence relations $\sim_n$ on $\mathcal{S}_A$. We set $\sim_0 = \mathcal{S}_A \times \mathcal{S}_A$ and, for $n = 0, 1, \ldots$, $s \sim_{n+1} s'$ iff $\forall \alpha \in \Sigma \cup \{\tau, \lambda\}$ $\forall \mathcal{C} \in \mathcal{S}_A/\sim_n$ it holds that $Prob(s, \tau^*\alpha, \mathcal{C}) = Prob(s', \tau^*\alpha, \mathcal{C})$.*

Lemma 7.3 *Let $A = (\Sigma, X, Q, q_0, \delta, \pi)$ be a probabilistic timed automaton and $s, s' \in \mathcal{S}_A$. Then,*

$$s \approx s' \Leftrightarrow \forall n \geq 0 \quad s \sim_n s'.$$

Proof: Let $\sim' = \bigcap_{n \geq 0} \sim_n$. We have to show that $\approx = \sim'$. It easy to see that $\sim'$ is an equivalence relation. By induction on n we can show that $\sim_0 \supseteq \sim_1 \supseteq \ldots \supseteq \approx$. Hence, $\sim' \supseteq \approx$. In order to show that $\sim' \subseteq \approx$ we prove that $\sim'$ is a branching bisimulation.

For each $n \geq 0$ and each $B \in \mathcal{S}_A/\sim'$, there exists a unique element $B_n \in \mathcal{S}_A/\sim_n$ with $B \subseteq B_n$. Then, $B_0 = \mathcal{S}_A \supseteq B_1 \supseteq B_2 \supseteq \ldots$ and $B = \bigcap_{n \geq 0} B_n$.

Claim 1: We want to prove that if $Prob(s, \tau^*\alpha, B) > 0$ and $B \in \mathcal{S}_A/\sim'$, then $Prob(s, \tau^*\alpha, B) = inf_{n \geq 0} Prob(s, \tau^*\alpha, B_n)$. In the following, we call $P[B_n]$ the probability $Prob(s, \tau^*\alpha, B_n)$. Since $B = \bigcap_{n \geq 0} B_n$ and $B_n \supseteq B_{n+1}$, we have $1 = P[B_0] \geq P[B_1] \geq \ldots \geq P[B_n]$. 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 $\Delta > 0$. There exists a subset X of $\mathcal{S}_A \setminus B$ such that $P[Y] < \Delta$ where $Y = \mathcal{S}_A \setminus (B \cup X)$. For all $n \geq 0$, $B_n = B \cup (Y \cap B_n) \cup (X \cap B_n)$. The sets B, $Y \cap B_n$ and $X \cap B_n$ are pairwise disjoint. Hence, $P[B_n] = P[B] + P[Y \cap B_n] + P[X \cap B_n] < P[B] + \Delta + P[X \cap B_n] = r + P[X \cap B_n]$. Since $r \leq P[B_n]$ we get $X \cap B_n \neq \emptyset$. As a consequence $X \cap B \neq \emptyset$, giving a contradiction.

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

□

Proof of Theorem 5.7
The algorithm terminates since the classes resulting from a split have the conditions enclosed in those of the original one. Actually, for each ψ the class $g \cap \psi$ is enclosed in g since $\psi \wedge g(h,k) \Rightarrow g(h,k)$, for any h, k.

Now, if the function split returns a cut, then if $g_2 \neq \bar{g}\mathcal{G}$, the refinement is correct by proposition 5.4. On the other hand, if $g_2 = \bar{g}_{\mathcal{G}}$, then we delete the relations $(q, v) \approx (q', v')$ such that (q, v) and (q', v'), by performing $\tau^*\alpha$, can reach (q'', v'') and (q''', v''') with $(q'', v'') \not\approx (q''', v''')$. This is correct by definition of branching bisimulation.

Therefore, by induction and by using lemma 7.3, it follows that the algorithm is correct.

□