Time-abstracting Bisimulation for Probabilistic Timed Automata

Taolue Chen
CWI
PO Box 94079, 1090 GB Amsterdam, The Netherlands
chen@cwi.nl

Tingting Han Joost-Pieter Katoen
MOVES, RWTH Aachen University, Germany
FMT, University of Twente, The Netherlands
{tingting.han, katoen}@cs.rwth-aachen.de

Abstract

*This paper focuses on probabilistic timed automata (*PTA*), an extension of timed automata with discrete probabilistic branchings. As the regions of these automata often lead to an exponential blowup, reduction techniques are of utmost importance. In this paper, we investigate probabilistic time-abstracting bisimulation (*PTAB*), an equivalence notion that abstracts from exact time delays.* PTAB *is proven to preserve probabilistic computational tree logic (*PCTL*). The region equivalence is a (very refined)* PTAB*. Furthermore, we provide a non-trivial adaptation of the traditional partition-refinement algorithm to compute the quotient under* PTAB*. This algorithm is symbolic in the sense that equivalence classes are represented as polyhedra.*

1 Introduction

Digital technology has been widely deployed in safety-critical situations and real-life environments, which leads to increased interests in computer systems expressed in terms of quantitative timing constraints. Timed automata (TAs, [2]) are a prominent and well-established formalism for modeling, analysis and verification of such *real-time* systems, which have received much attention both in terms of theoretical and practical developments.

Traditional approaches to the formal description of real-time systems usually express the system model purely in terms of nondeterminism. However, many real-life systems, such as multimedia equipment, communication protocols and networks, exhibit *random* behaviors, thus it may be desirable to refer to the *likelihood* of certain properties satisfied by the real-time system. This notion is particularly important when the fault-tolerance aspect of systems is concerned. This suggests the study of *probabilistic* models.

In this paper, we investigate *probabilistic timed automata* (PTAs) [8], which are a probabilistic extension of timed automata. As in TAs, in the research of PTAs, the notion of *region graphs* plays an essential role, see e.g. [8]. However, it is well recognized that albeit being a very useful tool for theoretical purposes, the region graph is too large to be of any practical interest: its size is exponential in the number of clocks of the system as well as the size of the constants in the time constraints. To overcome this explosion, inspired by [12], we propose *probabilistic time-abstracting bisimulations* (PTAB) for PTAs, where the passage of arbitrary time is abstracted by a τ transition. This equivalence is usually much coarser than the region equivalence, therefore, in practice, it induces a much smaller state space partition. In particular, the region equivalence constitutes a (very fine) probabilistic time-abstracting bisimulation. The *bisimulation quotient* is a finite-state Markov decision process (MDP), where the states are equivalence classes over *symbolic states* (a set of states) with either τ or discrete probabilistic transitions.

PTAB is particularly useful when the desired properties do not involve time constraints, which are prevalent in practice, i.e., safety, reachability, etc. Those properties can be well captured by the *probabilistic computation tree logic* (PCTL) [6], which is proven to be preserved under PTABs. In this case, the existing tools and algorithms for MDPs w.r.t. PCTL [7] can thus be applied for PTAs.

To obtain a minimal PTAB quotient, our algorithm works in the *partition-refinement* fashion [11]. We start from an initial partition that respects state labeling, and proceed by refining each block till it contains only bisimilar states. Due to the fact that PTAs involve an interweaving of time, nondeterminism and probability distributions, the minimization has thus to deal with the following difficulties: When taking a τ transition, it must guarantee that time traverses continuously, e.g., time cannot jump from 0 to 2 without traversing 1. Thus, we introduce the *timed predecessor* set as a splitter, as in [12]. Moreover, since a discrete transition results in one or more probability distributions, the splitter of only one block in [12] is, however, not applicable. Our algorithm instead, adopts the idea of *mutual-refine* technique in [3], which maintains a state partition and a distribution partition. In each refinement iteration, the distribution partition is used to refine a state partition and vice versa. The algorithm in [3], unfortunately, cannot be applied in our setting in a straightforward way, as the number of symbolic states in a block may grow in each iteration

when time comes into play. As a result, the symbolic state space and the distribution set vary in each iteration, let alone their partitions. To solve this problem, an *Expand* operator is introduced, recalculating the symbolic states in a block as well as the distribution set before the mutual-refine technique is applied. This algorithm is symbolic, namely, equivalence classes are symbolic states and set-theoretic operators are used to compute the set of (time) predecessor states of a symbolic state.

Related works. [3, 5] present algorithms for the probabilistic bisimulation and simulation for discrete probabilistic systems. [10] investigates weak probabilistic bisimulation for PTAs with a decision procedure, however, the algorithm is region based, which is tried to be avoided in the current paper. [8] presents a comprehensive exposition for PTAs and model checking algorithms for PTAs. [9] gives a symbolic algorithm for model checking, however, the problem of deciding time-abstracting bisimulations is not considered.

Structure of the paper. Section 2 presents basic definitions regarding PTAs. Section 3 defines probabilistic time-abstracting bisimulation. Section 4 presents the bisimulation minimization algorithm and constitutes the core of this paper. Section 5 shows that bisimulation preserves PCTL formulae. This paper is concluded in Section 6.

2 Preliminaries

Definition 1 (Probability distribution) *For a finite set S, a* distribution *is a function $\mu : S \to [0,1]$ such that $\sum_{s\in S}\mu(s) = 1$. μ_s^1 denotes the unique distribution on S with $\mu(s) = 1$. $Supp(\mu)$ denotes the* support *of μ, i.e., the set of states $s \in S$ with $\mu(s) > 0$. With $Distr(S)$ we denote the set of all probability distributions on S.*

Clocks and valuations. Let $\mathbb{R}$ denote the set of *non-negative* reals and let $\mathcal{X} = \{x_1, \cdots, x_n\}$ be a set of variables in $\mathbb{R}$, called *clocks*. An $\mathcal{X}$-valuation is a function $\nu : \mathcal{X} \mapsto \mathbb{R}$ assigning to each clock x a value $\nu(x)$. The set of all valuations over $\mathcal{X}$ is denoted by $\mathbb{R}^{\mathcal{X}}$. We write $\mathbf{0}$ for the valuation that assigns zero to all clocks. For a subset $X \subseteq \mathcal{X}$, $\nu[X := 0]$ is the valuation ν' such that $\forall x \in X.\ \nu'(x) = 0$ and $\forall x \notin X.\ \nu'(x) = \nu(x)$. For $d \in \mathbb{R}$, $\nu + d$ is the valuation ν'' such that $\forall x \in \mathcal{X}.\ \nu''(x) = \nu(x) + d$, by which it implies that all clocks proceed at the same speed.

Hyperplanes and polyhedra. An *clock constraint* on $\mathcal{X}$ is an expression of the form $x \bowtie c$ or $x - y \bowtie c$ or the conjunction of any clock constraints, where $x, y \in \mathcal{X}$, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and $c \in \mathbb{N}$. An *atomic constraint* does not contain any conjunctions. Let $CC(\mathcal{X})$ and $ACC(\mathcal{X})$ denote the set of clock constraints and atomic clock constraints over $\mathcal{X}$, respectively.

An $\mathcal{X}$-valuation ν *satisfies* a clock constraint g, denoted as $\nu \models g$, and is defined as follows: $\nu \models x \bowtie c$ iff $\nu(x) \bowtie c$, $\nu \models x - y \bowtie c$ iff $\nu(x) - \nu(y) \bowtie c$ and $\nu \models g_1 \wedge g_2$ if $\nu \models g_1$ and $\nu \models g_2$.

An *$\mathcal{X}$-hyperplane* is a set of valuations satisfying an atomic constraint. The class of $\mathcal{H}_{\mathcal{X}}$-polyhedra is defined as the smallest subset of $2^{\mathbb{R}^{\mathcal{X}}}$ which contains all $\mathcal{X}$-hyperplanes and is closed under set union, intersection, and complement.

Intersection ($\cap$), union ($\cup$) and complement() are well-defined operations on polyhedra. Given a polyhedron Z and a subset of clocks $X \subseteq \mathcal{X}$, the operation $Z[X := 0]$ is defined as $\{\nu \mid \nu[X := 0] \in Z\}$.

Probabilistic timed automata. Let AP denote a fixed, finite set of atomic propositions ranged over by $a, b, c, \ldots$.

Definition 2 (Probabilistic timed automata [8]) *A* probabilistic timed automaton *(*PTA*) is a tuple $\mathcal{G} = (Loc, \mathcal{X}, \ell_0, L, inv, \rightsquigarrow)$ where:*

- *Loc is a finite set of* locations*;*
- *$\mathcal{X}$ is a set of clocks;*
- *$\ell_0 \in Loc$ is the initial location;*
- *$L : Loc \to 2^{AP}$ is a labeling function for the locations;*
- *$\rightsquigarrow \subseteq Loc \times CC(\mathcal{X}) \times Distr(2^{\mathcal{X}} \times Loc)$ is a transition relation;*
- *inv:$Loc \to CC(\mathcal{X})$ is an invariant-assignment function.*

All invariants are downward-closed *in the sense that for any $d \in \mathbb{R}, \nu + d \models inv(\ell)$ implies that $\nu \models inv(\ell)$.*

The system starts in location ℓ_0 with all its clocks initialized to 0. The values of all the clocks increase uniformly with time. We refer to $\ell \overset{g}{\rightsquigarrow} \eta$ as a *transition*, where the *guard* g is a clock constraint on the clocks of $\mathcal{G}$ and η is a distribution over the (X, ℓ) pairs with $X \subseteq \mathcal{X}$ a set of clocks to be reset and ℓ the successor location. The intuition is that the PTA $\mathcal{G}$ can move from location ℓ to location ℓ' via two phases. In the first phase, a distribution η is nondeterministically chosen when g holds. In the second phase, a successor location ℓ' is probabilistically chosen according to $\eta(X, \ell')$, where the clocks in X should be reset when entering ℓ'. The function *inv* assigns to each ℓ a location invariant that constrains the amount of time that may be spent in ℓ. In other words, location ℓ should be left before the invariant $inv(\ell)$ becomes invalid. If there is no outgoing transition enabled and no further progress is possible, it is a *timelock*. The labeling function L associates to each location ℓ a set of atomic propositions that are valid in ℓ.

Example 1 Fig. 1 is an example PTA, where from ℓ_1 there are two distributions (or transitions) and thus is nondeterministic. The transitions to ℓ_3 and ℓ_4 share the same guard $x > 1$, since they belong to the same distribution. The transition to ℓ_3 resets the clock $\{x\}$. The labeling on ℓ_1 and ℓ_3 is $\{a\}, \emptyset$ otherwise.

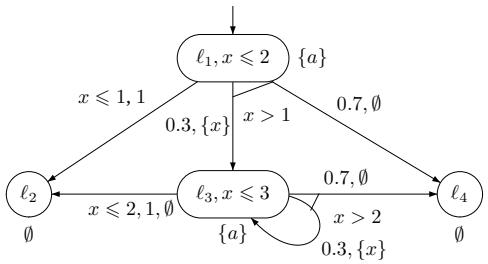

Figure 1. An example PTA

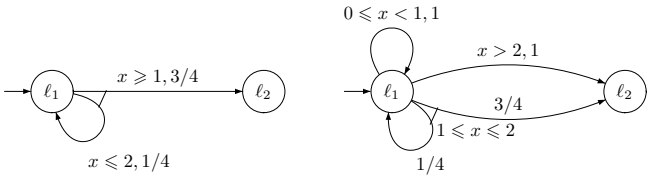

Figure 2. The encoding to a one-clock-constraint model

Remark 1 *Due to the syntax, one transition is associated with a single clock constraint. This requirement is intuitive and reasonable since the more-than-one clock constraint case, see e.g., [10], can be encoded by adding more distributions. To give an example, the left in Fig. 2 is a* PTA *with two clock constraints in one distribution. This can be encoded by the* PTA *on the right in Fig. 2. It goes as follows: When $0 \leqslant x < 1$, the transition from ℓ_1 to ℓ_2 is not enabled. Thus the only possible transition is the self-loop on ℓ_1, which is normalized to probability 1. When $1 \leqslant x \leqslant 2$, both transitions are enabled, and their probabilities remain the same. The $x > 2$ case is similar as $0 \leqslant x < 1$.*

Probabilistic timed structures. The semantics of a timed automaton is an infinite timed transition system. The semantics of a PTA is provided by a probabilistic timed structure, in fact an infinite MDP.

Definition 3 (Probabilistic timed structures) *A* probabilistic timed structure *(*PTS*) $\mathcal{M}$ is a* labeled Markov decision process $(S, Steps, L, s_0)$ *where S is a set of states, $Steps : S \to 2^{\mathbb{R} \times Distr(S)}$ is a function that assigns to each state $s \in S$ a set of pairs (t, μ) where $t \in \mathbb{R}$ and $\mu \in Distr(S)$ and $L : S \to 2^{AP}$ is a state labeling function. $s_0 \in S$ is the initial state.*

$Steps(s)$ is the set of transitions that can be nondeterministically chosen in state s. The transition labels are of the form (t, μ) where t is the duration of the transition and μ is the probability distribution over the successor states. $s \overset{t,\mu}{\to} s'$ means that after t time units have elapsed, a transition is fired from s to s' with probability $\mu(s')$.

Paths. Paths in a PTS arise by resolving both the nondeterministic and probabilistic choices. A *path* of the PTS $\mathcal{M} = (S, Steps, L, s_0)$ is a finite or infinite sequence:

$$\omega = s_0 \overset{t_0,\mu_0}{\to} s_1 \overset{t_1,\mu_1}{\to} s_2 \overset{t_2,\mu_2}{\to} \ldots$$

where $s_i \in S$, $(t_i, \mu_i) \in Steps(s_i)$ and $\mu_i(s_{i+1}) > 0$ for all $0 \leqslant i \leqslant |\omega|$, where $|\omega|$ is the number of transitions in ω. A finite path ω ends in a state, denoted $last(\omega)$.

We use $Path_{fin}$ to denote the set of finite paths and $Path_{fin}(s)$ the set of finite paths that start in s. $Path_{inf}$ and $Path_{inf}(s)$ are the counterpart for infinite paths. Consider a path $\omega \in Path_{inf}$ of $\mathcal{M}$. A *position* of ω is a pair (i, t') where $i \in \mathbb{N}$ and $t' \in \mathbb{R}$ such that $0 \leqslant t' \leqslant t_i$. The *state at position* (i, t') on ω is denoted by $\omega(i, t') = s_i + t'$.

Definition 4 (Scheduler of a PTS) *A* scheduler *of a* PTS $\mathcal{M} = (S, Steps, L, s_0)$ *is a function $\mathfrak{G}$ mapping every finite path ω of $\mathcal{M}$ to a pair (t, μ) such that $\mathfrak{G}(\omega) \in Steps(last(\omega))$. Let $\mathfrak{W}$ be the set of all schedulers of $\mathcal{M}$.*

A scheduler resolves the nondeterminism by choosing a probability distribution based on the process executed so far. Formally, if a PTS is guided by scheduler $\mathfrak{G}$ and has the finite path ω as its *history*, then it will be in state s in the next step with probability $\mu(s)$, where $\mathfrak{G}(\omega) = (t, \mu)$.

We denote the set of infinite paths induced by a given scheduler $\mathfrak{G}$ to be $Paths^{\mathfrak{G}}$ with $Paths^{\mathfrak{G}} = \{\omega \in Paths \mid \mathfrak{G}(\omega{\downarrow_i}) = \mu_i \text{ for } i \geqslant 0\}$, where $\omega{\downarrow_i}$ returns the prefix of ω up to length i. $Paths^{\mathfrak{G}}(s)$ is defined as $Paths^{\mathfrak{G}} \cap Paths(s)$.

Scheduler $\mathfrak{G}$ on PTS $\mathcal{M}$ induces a discrete-time Markov chain (DTMC) $\mathcal{M}^{\mathfrak{G}}$, where the nondeterminism has been resolved. Each state in $\mathcal{M}^{\mathfrak{G}}$ is a finite path fragment ω in $\mathcal{M}$. The transition probability is determined by $\mathfrak{G}$ and the chosen probability distribution. We omit the formal definition of $\mathcal{M}^{\mathfrak{G}}$, the probability space of $\mathcal{M}^{\mathfrak{G}}$, and a basic cylinder as they are standard and can be found in e.g., [4].

Semantics. Any PTA can be interpreted as a PTS. Due to the continuous nature of clocks, these underlying PTSs have infinitely many states (even uncountably many), and are infinitely branching. PTA can thus be considered as a *finite* description of infinite PTSs.

Given a PTA $\mathcal{G} = (Loc, \mathcal{X}, \ell_0, L, inv, \rightsquigarrow)$, a *state* of $\mathcal{G}$ is a pair (ℓ, ν), where $\ell \in Loc$ is a location and $\nu \in inv(\ell)$ is a valuation satisfying the invariant of ℓ.

Definition 5 (PTS semantics of a PTA) *Let $\mathcal{G} = (Loc, \mathcal{X}, \ell_0, L, inv, \rightsquigarrow)$ be a* PTA. *The* PTS *of $\mathcal{G}$ is $\mathcal{M}_{\mathcal{G}} = (S, Steps, L', s_0)$ with:*

- $S = \{(\ell, \nu) \mid \nu \models inv(\ell), \ell \in Loc\}$;
- $L'((\ell, \nu)) = L(\ell) \cup \{g \in ACC(\mathcal{X}) \mid \nu \models g\}$;
- $s_0 = (\ell_0, \mathbf{0})$;
- *Given $(t, \mu) \in Steps((\ell, \nu))$, transition $\to$ is defined by the following rules:*

– discrete transition: $(\ell,\nu) \overset{0,\mu}{\to} (\ell',\nu')$, *if the following conditions hold:*

1. $\exists$ *transition* $\ell \overset{g}{\rightsquigarrow} \eta$ *in* $\mathcal{G}$ *with* $\eta(\ell',X) > 0$;
2. $\nu \models g$;
3. $\nu' = \nu[X := 0]$;
4. $\mu(\ell',\nu') = \sum_{X\subseteq\mathcal{X},\nu'=\nu[X:=0]} \eta(X,\ell')$.

Usually, we simply write $(\ell,\nu) \to \mu$.

– delay transition: $(\ell,\nu) \overset{d,\mathbf{1}}{\to} (\ell,\nu+d)$ *for all* $0 \leqslant d \leqslant t$, *if* $\nu + d \models inv(\ell)$.
Note that $\mathbf{1}$ *indicates that the probability distribution is* $\mu^1_{(\ell,\nu+d)}$. *Usually, we simply write* $(\ell,\nu) \overset{d}{\to} (\ell,\nu+d)$.

Symbolic states. We define symbolic states which are used for the effective representation and manipulation of the infinite state space of PTS. Generally, a symbolic state is a set of states of $\mathcal{M}_\mathcal{G}$.

In a nutshell, a *zone* $Z \in \mathbb{R}^\mathcal{X}$ of $\mathcal{X}$ is a set of valuations which satisfy a conjunction of constraints. Formally, the zone for the constraint g is $Z = \{\nu \mid \nu(x) \models g, x \in \mathcal{X}\}$. Geometrically, a zone is a polyhedron (note that we do not require a zone to be convex). A *symbolic state* $\mathbf{S}$ is a set of states whose clock evaluations form a zone. Strictly, $\mathbf{S}$ is a set of pairs of location and zone, namely, of the form (ℓ, Z). The union of all symbolic states is the state space S.

3 Time-abstracting Bisimulation for PTS

In order to refine the dense state space as much as possible, we adopt the time-abstracting bisimulation [12] for state space minimization, which abstracts from the quantitative aspect of time: we know that *some* time passes, but not how much. We first introduce a technical definition:

Definition 6 $\mu,\mu' \in Distr(S)$ *are equivalent w.r.t. equivalence* $\mathcal{R}$ *on* S, *written* $\mu \equiv_\mathcal{R} \mu'$, *if* $\forall U \in S/\mathcal{R}.\ \mu(U) = \mu'(U)$.

Definition 7 (Probabilistic time-abstracting bisimulation) *Let* $\mathcal{G}$ *be a* PTA, $\mathcal{M}_\mathcal{G} = (S, Steps, L', s_0)$ *be the* PTS *of* $\mathcal{G}$. *A* probabilistic time-abstracting bisimulation *(*PTAB*) for* $\mathcal{G}$ *is an equivalence relation* $\mathcal{R}$ *on* S *such that for all states* $(s_1,s_2) \in \mathcal{R}$, *the following conditions hold:*

- $L'(s_1) = L'(s_2)$;
- *If* $s_1 \overset{t_1}{\to} s_1'$, *for some* $t_1 \in \mathbb{R}$, *then there exists* $t_2 \in \mathbb{R}$ *and* $s_2' \in S$ *such that* $s_2 \overset{t_2}{\to} s_2'$ *and* $(s_1', s_2') \in \mathcal{R}$;
- *If* $s_1 \to \mu_1$, *for some* $\mu_1 \in Distr(S)$, *then there exists some* $\mu_2 \in Distr(S)$ *such that* $s_2 \to \mu_2$ *and* $\mu_1 \equiv_\mathcal{R} \mu_2$.

s_1 *and* s_2 *are* probabilistic time-abstracting bisimilar, *denoted* $s_1 \sim s_2$, *if* $(s_1,s_2) \in \mathcal{R}$ *for some* PTAB $\mathcal{R}$.

We use τ-transitions to abstract away the exact time passage, formally, $s \overset{\tau}{\to} s'$ iff $\exists t \in \mathbb{R}.s \overset{t}{\to} s'$.

Region equivalence. In the following, we first recall the definition and properties of region equivalence (RE) [1] which is essential in turning the infinite state space of a PTS into a *finite* quotient. We will then show a similar result as in [12] that the RE for PTS is in fact a PTAB.

Consider a set of clocks $\mathcal{X}$ and let $c = c_{\max}(\mathcal{G})$ the largest integer constant among all the clock constraints and invariants in $\mathcal{G}$. Two clock evaluations ν and ν' are *region equivalent*, denoted $\nu \cong \nu'$, iff they satisfy:

- $\forall x \in \mathcal{X}$, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or both $\nu(x) > c$ and $\nu'(x) > c$.
- $\forall x,y \in \mathcal{X}$, either $\lfloor \nu(x) - \nu(y) \rfloor = \lfloor \nu'(x) - \nu'(y) \rfloor$ or both $\lfloor \nu(x) - \nu(y) \rfloor > c$ and $\lfloor \nu'(x) - \nu'(y) \rfloor > c$.

Note that $\lfloor r \rfloor$ is the maximal integer that is at most r. The equivalence classes induced by $\cong$ are *regions*. The region equivalence can be lifted to states such that $(\ell,\nu) \cong (\ell',\nu')$ if $\ell = \ell'$ and $\nu \cong \nu'$.

The region equivalence has following properties:

Lemma 8 *For valuations* $\nu,\nu' \in \mathbb{R}^\mathcal{X}$ *with* $\nu \cong \nu'$:

1. *for any zone* Z, $\nu \in Z$ *iff* $\nu' \in Z$;
2. *for any set of clocks* $X \subseteq \mathcal{X}$, $\nu[X := 0] \cong \nu'[X := 0]$;
3. $\forall d \geqslant 0\ \exists d' \geqslant 0.\ \nu + d \cong \nu' + d'$.

Theorem 9 *The region equivalence is a* PTAB, *i.e.,* $\cong \subseteq \sim$.

Proof: Let $(\ell,\nu),(\ell,\nu')$ be two states in PTS $\mathcal{M} = (S, Steps, L', s_0)$ such that $(\ell,\nu) \cong (\ell,\nu')$.

- (Labels) Due to the fact that if $(\ell,\nu) \cong (\ell,\nu')$, $\nu \in Z$ iff $\nu' \in Z$ (Lemma 8(1)), it holds that $\{g \in ACC(\mathcal{X}) \mid \nu \models g\} = \{g' \in ACC(\mathcal{X}) \mid \nu' \models g'\}$. Since $L'((\ell,\nu)) = L(\ell) \cup \{g \in ACC(\mathcal{X}) \mid \nu \models g\}$, where L is the labeling function in the corresponding PTA $\mathcal{G}$, we have $L'((\ell,\nu)) = L'((\ell,\nu'))$.
- (Timed transition) Let $(\ell,\nu) \overset{d}{\to} (\ell,\nu+d)$. Due to Lemma 8(3), there exists a $d' \geqslant 0$ such that $\nu + d \cong \nu' + d'$. $\nu', \nu' + d' \models inv(\ell)$ since $\nu, \nu + d \models inv(\ell)$. For any $d'' < d'$, $\nu + d'' \models inv(\ell)$, by the downward-closedness of $inv(\ell)$. Thus $(\ell,\nu') \overset{d'}{\to} (\ell,\nu'+d')$.
- (Prob. transition) Let $(\ell,\nu) \to \mu$. μ is chosen by some scheduler $\mathfrak{G}$. As $\mathfrak{G}$ can only select enabled transitions, $\mu \models g$. Let $Z = \{\nu \mid \nu \models g\}$, $\nu \in Z$. Since $(\ell,\nu) \cong (\ell,\nu')$, due to Lemma 8(1), $\nu' \in Z$, which means that $\nu' \models g$, thus μ is also enabled in (ℓ,ν'). Therefore, we can construct a scheduler $\mathfrak{G}'$ which chooses the same distribution as $\mathfrak{G}$. Since $\mu \equiv_\cong \mu$, $(\ell,\nu') \to \mu$.

RE satisfies all conditions of being a PTAB, thus $\cong \subseteq \sim$. □

The above theorem asserts that the region equivalence is a (probably very refined) PTAB. Note that the converse does not hold in general. It can be the case that $(\ell,\nu) \sim (\ell',\nu')$ where $\ell \neq \ell'$ (see Example 2), however, $(\ell,\nu) \ncong (\ell',\nu')$.

The next result shows that timelocks are preserved by $\sim$.

Proposition 1 *If* $(\ell,\nu) \sim (\ell',\nu')$, *then* (ℓ,ν) *has a timelock iff* (ℓ',ν') *has a timelock.*

Evidently, the converse does not hold.

4 Minimization of PTA

Having defined the PTAB, an immediate question is: how to compute it, since one of the crucial steps of exploiting PTAB for verification is to generate the quotient of the given PTA. A simple answer might be, taking the region graph, since the region equivalence is a PTAB! However, as pointed in [1], the number of regions grows exponentially with the number of clocks in the TA, the finite region equivalence quotient is too large to be of any practical interest, and the same applies to PTAs. Therefore, for the sake of efficiency, we are interested in the *minimal* quotient, namely, the one corresponding to the *coarsest* bisimulation. In what follows, we will propose an algorithm to compute the quotient of a PTS w.r.t. the coarsest PTAB, which combines the algorithm in [12] for timed automata and the algorithm in [3] for MDPs.

Partition refinement. Prior to presenting our algorithm, let us first recall how the minimization algorithm works for finite (non-probabilistic, without time) labeled transition systems (LTSs). The algorithm relies on the *partition-refinement* technique [11]. Roughly speaking, the state space S is partitioned in *blocks*, i.e., pairwise disjoint sets of states. Starting from an initial partition Π_0 where, e.g., all equally-labeled states form a block, the algorithm successively refines these blocks such that ultimately each block contains only bisimilar states. The refinement is based on the fact that a bisimulation induces a pre-stable partition. Formally, given a partition Π of states and blocks $C_1, C_2 \in \Pi$, C_1 is *pre-stable* w.r.t. C_2 if $C_1 \subseteq \mathsf{pred}(C_2)$ or $C_1 \cap \mathsf{pred}(C_2) = \emptyset$, where $\mathsf{pred}(C)$ is the set of direct predecessors of all the states in C. If C_1 is not stable w.r.t. C_2, then C_1 can further be partitioned into two sub-blocks $C_1 \cap \mathsf{pred}(C_2)$ and $C_1 \setminus \mathsf{pred}(C_2)$. In this case, C_2 is a *splitter* of C_1. Π is *pre-stable* if all its blocks are pairwise pre-stable. The main sketch of the algorithm below, albeit simple, is the essence of partition refinement.

Algorithm 1 The general partition-refinement algorithm

Require: The LTS, the initial partition Π_0
Ensure: The partition Π under the coarsest bisimulation

```
1: Π := Π_0;
2: while (∃C_1, C_2 ∈ Π, C_1 is not stable w.r.t. C_2) do
3:     Π_{C_1} := {C_1 ∩ pred(C_2), C_1 \ pred(C_2)};
4:     Π := (Π \ {C_1}) ∪ Π_{C_1};
5: end while
6: return Π;
```

The scheme can be adapted to infinite state spaces, assuming that they admit effective representations of blocks and decision procedures for computing intersection, set-difference and predecessors of blocks, and testing whether a block is empty. For termination, it must be ensured that a pre-stable partition always exists. In [12], such an adaptation is given for TA to compute time-abstracting bisimulation since the state space of TA falls in this category.

4.1 Bisimulation quotienting algorithm

In this section, we shall move further, taking the probabilistic transitions into account. This is not trivial since the infinite states (caused by time) and probabilistic transitions are closely interweaved, thus the set pred should be replaced by the *discrete predecessors* discpred and the *timed predecessors* timepred in a proper way.

The set of timed predecessors splits a block where a discontinuity on time occurs when taking a timed transition. This is captured by the *time-refinement operator* (see Def. 10). Besides, due to Proposition 1, a state having a timelock must be in a different block than a state that does not suffer from a timelock. This suggests a *first discrete-refinement operator* (see Def. 11).

For discrete predecessors, since a probability distribution rather than a state is associated with a transition, successively dividing a block by a single-block splitter does not suffice. Instead, we adapt the *mutual-refine* algorithm in [3]. The algorithm maintains a distribution partition in addition to a state partition, and in each iteration refines one partition by the other and vice versa, till both partitions stabilize. However, this algorithm cannot be directly applied in our case, since a block might expand in a new partition as the number of symbolic states in it may grow. Consequently, in a new partition, it is possible that the distribution set differs from the one in the last iteration and obviously the old distribution partition is obsolete. The *Expand operator* (see Def. 12) thus recalculates the symbolic states, the distribution set, as well as the distribution partition and as a final step in one iteration, a state block is refined by the *second discrete-refinement operator* (see Def. 13) using the newest distribution partition.

The algorithm is presented in Algo. 2. A detailed explanation follows.

Determining the initial partition. The initial partition of states $\Pi_{AP} = S/\mathcal{R}_{AP}$ is the AP-partition of S, where $\mathcal{R}_{AP} = \{(s_1, s_2) \in S \times S \mid L(s_1) = L(s_2)\}$. Initially, the zone of symbolic state (ℓ, ν) is $\mathit{inv}(\ell)$, thus on the symbolic state level, $\Pi_{AP} = \big\{\{([\ell]_{\mathcal{R}_{AP}}, \mathit{inv}(\ell))\} \mid \ell \in \mathit{Loc}\big\}$.

Refining partitions. In the rest of this section, we will concentrate on how to refine an existing partition. For reference convenience, given a PTA, we designate each transition (leading to a distribution) a unique action name, and for each location ℓ, we denote $\nabla(\ell)$ as the set of outgoing transitions from ℓ, which is ranged over by $\alpha, \beta \ldots$. Let $\nabla = \bigcup_{\ell \in Loc} \nabla(\ell)$. Moreover, for each transition α, g_α and μ_α are the guard and the resulting distribution, respectively.

Algorithm 2 The partition-refinement algorithm for PTA

Require: The PTA $\mathcal{G}$ and PTS $\mathcal{M}_{\mathcal{G}} = (S, Steps, L', s_0)$
Ensure: The partition Π under the coarsest PTAB
1: Initialization: Get the initial partition, $\Pi := \Pi_{AP}$;
2: Partition Π according to $Refine_d^1(\Pi, \nabla)$.
3: **Repeat**
4: PHASE I – **Refine Π by discrete transitions:**
5: Choose some block $C \in \Pi$,
6: $C' = Expand(C, \Pi)$;
7: Update the distribution set $Distr'$;
8: Compute the equivalence class $Distr'/\Pi$;
9: Choose some $M \in Distr'/\Pi$;
10: $\Pi := Refine_d^2(C', M)$;
11: PHASE II – **Refine Π by time delays:**
12: Choose some block $C \in \Pi$;
13: $\Pi := Refine_t(\Pi, C)$;
14: **until** Π does not change.
15: **return** Π;

For instance, there are 4 uniquely labeled transitions in the PTA in Fig. 1.

As we have two types of transitions, there are two types of refinements as well. For timed transitions, the time-refinement operator is as follows:

Definition 10 (The time-refinement operator) *Let Π be a partition of S and $C_1, C_2 \in \Pi$. Then $Refine_t(C_1, C_2)$ equals:*

$$\{C_1 \cap \mathsf{timepred}(C_2), C_1 \setminus \mathsf{timepred}(C_2)\} \setminus \{\emptyset\},$$

where $\mathsf{timepred}(S) = \{s \mid \exists s' \in S, t \in \mathbb{R}, s \xrightarrow{t} s'\}$.

We define $Refine_t(\Pi, C_2) = \bigcup_{C_1 \in \Pi} Refine_t(C_1, C_2)$.

This corresponds to PHASE II (line 11-13) in Algo. 2.

For discrete transitions, the split consists of two steps. The first step is to differentiate the symbolic states that can fire a discrete transition from those that cannot. In this step, a splitter is the action set ∇, which refines a block as:

Definition 11 (The 1^{st} discrete-refinement operator) *Let Π be a partition of S, ∇ be the action set and $C \in \Pi$. Then:*

$$Refine_d^1(C, \nabla) = \{C^+, C^-\} \setminus \{\emptyset\},$$

where $C^+ = \{(\ell, Z) \mid \exists \alpha \in \nabla(\ell), Z \subseteq g_\alpha\}$ and $C^- = \{(\ell, Z) \mid \forall \alpha \in \nabla(\ell), Z \cap g_\alpha = \emptyset\}$.

We define $Refine_d^1(\Pi, \nabla) = \bigcup_{C \in \Pi} Refine_d^1(C, \nabla)$.

All symbolic states in C^+ have an enabled discrete transition whereas none of them in C^- does. Actually, C^- is the set of states that have a timelock. This is used in line 2 of the algorithm. This operator has only to be performed once, because the further refinement won't change the fact of having a discrete transition.

As the second step, we can further partition C^+ according to the distributions. Suppose the current partition $\Pi = \{C_1, ..., C_n\}$, $n \in \mathbb{N}$. For any block C_i, we can write $C_i = \{(\ell_i^1, Y_i^1), \ldots, (\ell_i^q, Y_i^q)\}$ with $C_i = \bigcup_{1 \leqslant j \leqslant q} (\ell_i^j, Y_i^j)$,

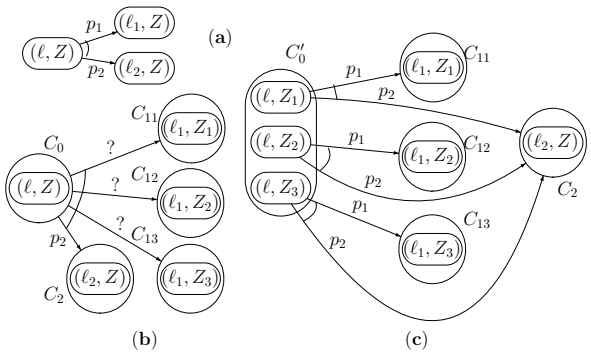

Figure 3. The motivation of *Expand*

and for any $1 \leqslant h \neq k \leqslant q$, $\ell_i^h \neq \ell_i^k$. For index $1 \leqslant h \leqslant m$, $\alpha \in \nabla(\ell_i^h)$ such that $Y_i^h \subseteq g_\alpha$, we want to derive the distributions induced by α.

However, it is possible that the resulting symbolic state of a transition bestrides different blocks, where the probability $\mu(C)$ to a block C may not be well defined. For instance, Fig. 3(a) illustrates a distribution from $\mathbf{S}_0 = (\ell, Z)$ to $\mathbf{S}_1 = (\ell_1, Z)$ and $\mathbf{S}_2 = (\ell_2, Z)$, where $\mathbf{S}_0 \in C_0$, $\mathbf{S}_2 \in C_2$ but $\mathbf{S}_1$ scatters in C_{11}, C_{12} and C_{13}, as in Fig. 3(b). Note that $\{Z_1, Z_2, Z_3\}$ is a partition of Z. The problem is that $\mu(C_{1i})$ can *not* be defined for $1 \leqslant i \leqslant 3$.

To solve this problem, we have to split a *symbolic state* in such a way that each sub-symbolic state has well-defined probabilistic transitions over the partition Π, as in Fig. 3(c). As a result of this split, the number of blocks stays the same, but the symbolic state space expands in terms of transitions. In the following, we define the *Expand* operator formally.

For symbolic state $\mathbf{S} = (\ell, Z)$ and action α, let $Supp(\mu_\alpha) = \{(\ell_1, X_1), \ldots, (\ell_m, X_m)\}$ with probabilities $p_1, ..., p_m$, respectively, where $X_i \subseteq \mathcal{X}$ is the reset clock set and p_i is the associated probability with $\sum_{1 \leqslant i \leqslant m} p_i = 1$. For successor (ℓ_j, X_j), the resulting symbolic state is $\mathbf{S}_j = (\ell_j, Z[X_j := 0])$. In the following, we will split Z into a partition $\mathcal{Z} = \{Z_1, \ldots, Z_f\}$ such that for any (sub) symbolic state (ℓ, Z') of $\mathbf{S}$, i.e., $Z' \in \mathcal{Z}$, each of its successor states is located only in one block. For $C_k \in \{C_1, ..., C_n\}$, define

$$Z_j^k = \big\{(\ell, \nu) \mid \nu \in Z,\ (\ell_j, \nu[X_j := 0]) \in C_k\big\}.$$

It is possible that $Z_j^k = \emptyset$. For each successor $1 \leqslant j \leqslant m$, $\{Z_j^1, Z_j^2, \ldots, Z_j^n\}$ is a partition of Z. We have the following partitions:

$$\begin{array}{ll} \text{For 1-st successor} : & \{Z_1^1, \ldots, Z_1^k, \ldots, Z_1^n\}, \\ & \vdots \\ \text{For } j\text{-th successor} : & \{Z_j^1, \ldots, Z_j^k, \ldots, Z_j^n\}, \\ & \vdots \\ \text{For } m\text{-th successor} : & \{Z_m^1, \ldots, Z_m^k, \ldots, Z_m^n\} \end{array}$$

We define

$$Z_{\vec{k}} = \bigcap_{1 \leqslant j \leqslant m} Z_j^{\vec{k}[j]},$$

where for each j, $1 \leqslant \vec{k}[j] \leqslant n$. $Z_j^{\vec{k}[j]}$ denotes choosing the $\vec{k}[j]$-th element in the j-th row, where $\vec{k}$ is a vector of indices. Stated in words, $Z_{\vec{k}}$ is obtained by taking the intersection of one arbitrary element from each row in the above "matrix". It is not difficult to see that

$$\{Z_{\vec{k}} \mid 1 \leqslant \vec{k}[j] \leqslant n, 1 \leqslant j \leqslant m\} \setminus \{\emptyset\}$$

is a partition of Z and in the worst case, this partition may contain n^m blocks.

For each $Z_{\vec{k}}$, since $Z_{\vec{k}} \subseteq Z_j^{\vec{k}[j]}$ for $1 \leqslant j \leqslant m$, it must be the case that $(\ell_j, Z_{\vec{k}}[X_j := 0]) \subseteq C_{\vec{k}[j]}$ for each $1 \leqslant j \leqslant m$. Hence, the probability from the symbolic state $(\ell, Z_{\vec{k}})$ to C_i for $1 \leqslant i \leqslant n$ is obtained by adding the nonzero probabilities in the i-th column:

$$\Pr\big((\ell, Z_{\vec{k}}), \alpha, C_i\big) = \sum_{1 \leqslant j \leqslant m, \vec{k}[j]=i} p_j.$$

In the following, we merge those $Z_{\vec{k}}$ and $Z_{\vec{k}'}$ such that for each $C_i \in \Pi$, $\Pr\big((\ell, Z_{\vec{k}}), \alpha, C_i\big) = \Pr\big((\ell, Z_{\vec{k}'}), \alpha, C_i\big)$. The partition $\mathcal{Z} = \{Z_1, ..., Z_f\}$ is then obtained. And the expansion operator expands a block with (possibly) more refined symbolic states as follows:

Definition 12 (The Expand operator) *Let Π be a partition of S, $\alpha \in \nabla$, $C \in \Pi$ and $\mathbf{S} = (\ell, Z) \in C$. Then:*

$$\mathit{Expand}(\mathbf{S}, \alpha, \Pi) = \{(\ell, Z_i) \mid 1 \leqslant i \leqslant f\},$$

where Z_i is defined as described above.

$\mathit{Expand}(C, \Pi) = \bigcup_{\mathbf{S} \in C, \alpha \in \nabla} \mathit{Expand}(\mathbf{S}, \alpha, \Pi)$.

Note that for each sub-symbolic state $\mathbf{T}$ of $\mathbf{S}$ in $\mathit{Expand}(\mathbf{S}, \alpha, \Pi)$, $\Pr(\mathbf{T}, \alpha, C_k)$ is well-defined. Let us denote $\mu_{\mathbf{T},\alpha}$ as the distribution over Π from $\mathbf{T}$ via action α. Now the distribution set is updated as $\mathit{Distr}' = \{\mu_{\mathbf{T},\alpha} \mid \mathbf{T} \in \mathit{Expand}(C, \Pi)$ with $\mathbf{T} = (\ell, Y)$ for some ℓ, Y and $\alpha \in \nabla(\ell)\}$. The distribution partition on Distr' over Π, denoted by Distr'/Π, can thus be updated accordingly, based on the following fact: Let $M \in \mathit{Distr}'/\Pi$, then $\forall \mu, \mu' \in M$, $\mu(C) = \mu'(C)$ for any $C \in \Pi$. As the mutual-refine technique, the state partition can in turn be refined by the distribution partition as follows:

Definition 13 (The 2^{nd} discrete-refinement operator) *Let Π be a partition of S with $C \in \Pi$, $C' = \mathit{Expand}(C, \Pi)$ and $M \in \mathit{Distr}'/\Pi$. Then:*

$$\mathit{Refine}_d^2(C, M) = \{C_M, C' \setminus C_M\} \setminus \{\emptyset\},$$

where $C_M = \{\mathbf{T} \mid \mu_{\mathbf{T},\alpha} \in M$ and $\mathbf{T} = (\ell, Y), \alpha \in \nabla(\ell)\}$.

The above steps correspond to PHASE I, line 4-10 in Algo. 2.

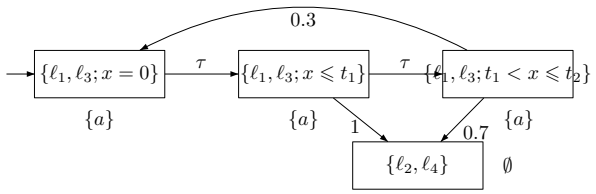

Figure 4. The bisimulation quotient

Example 2 The bisimulation quotient of the PTA in Fig. 1 is shown in Fig. 4. There are four equivalence classes. The label τ denotes that some time passes during the transition. The intuition is that from ℓ_1 or ℓ_3 it is possible to go to a state within a given period of time (the first τ) where either it takes a discrete transition to the sinking state or it stays (taking the second τ) for some time till it can take a transition back resetting its clock with probability 0.3 or it goes to the sinking state with probability 0.7. t_1 and t_2 are arbitrary time points which have been abstracted from the original model.

Correctness and termination. It is not difficult to see that by any of the *Refine* operators, we may obtain some new blocks, where for any two states in different blocks, they are not bisimilar and these blocks are disjoint. The correctness of the algorithm follows from standard correctness arguments of the partition-refinement algorithm. Termination is ensured by Theorem 9. Namely, in the worst case, the algorithm will generate the partition induced by the region equivalence.

Complexity. We analyze the complexity of the algorithm briefly. Since in the worst case, the region equivalence will be obtained, our algorithm needs to refine exponentially many times to reach the fixpoint, and thus it is an EXPTIME algorithm. On the other hand, it is not hard to see that generally for PTAs the EXPTIME lower bound can be obtained. However, we note that (1) for PTAs with only one clock, we can show that the algorithm only needs polynomial many time to reach the fixpoint. Thus in this case, we can get a polynomial time algorithm; (2) In practice, usually, PTAs have a much coarser partition than the one induced by the region equivalence, and thus our algorithm is expected to perform pretty well in this case.

5 Verification of branching-time properties

The logic PCTL. In this section we prove that PTAB preserves branching-time properties specified in probabilistic CTL [6]. The syntax and semantics of PCTL is:

$$\Phi ::= \mathrm{tt} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq p}(\phi),$$

where $p \in [0, 1]$ is a probability, $a \in AP$, $\trianglelefteq \in \{<, \leqslant, >, \geqslant\}$ and ϕ is a path formula defined as:

$$\phi ::= \Phi \,\mathsf{U}\, \Phi \mid \Phi \,\mathsf{W}\, \Phi.$$

The path formula $\Phi \,\mathsf{U}\, \Psi$ asserts that Ψ is eventually satisfied and that all preceding states satisfy Φ. W is the weak counterpart of U which does not require Ψ to eventually become true. Most of the operators are standard, with the exception that $s \models \mathcal{P}_{\trianglelefteq p}(\phi)$ iff for any scheduler $\mathfrak{G} \in \mathfrak{W}$, $Prob(s,\phi) \trianglelefteq p$ in the DTMC $\mathcal{D}^{\mathfrak{G}}$, where $Prob(s,\phi) = \Pr\{\sigma \in Paths(s) \mid \sigma \models \phi\}$.

Bisimulation can be lifted to paths in the following way:

Lemma 14 (Bisimulation on paths) *Let $s \sim s'$. Then: for each (finite or infinite) path $\omega = s_0 \overset{t_0,\mu_0}{\rightarrow} s_1 \overset{t_1,\mu_1}{\rightarrow} s_2 \cdots \in Paths(s)$, there exists a path $\omega' = s'_0 \overset{t'_0,\mu'_0}{\rightarrow} s_1 \overset{t'_1,\mu'_1}{\rightarrow} s'_2 \cdots \in Paths(s')$ of the same length such that $s_i \sim s'_i$, for all $i \geqslant 0$.*

Theorem 15 *Let $\mathcal{G}$ be a* PTA *and $\sim$ be a* PTAB *on $\mathcal{G}$. For any* PCTL *formula Φ: $s \sim s'$ implies $s \models \Phi$ iff $s' \models \Phi$.*

Proof: The proof is by induction on the structure of Φ. Basis: If $s \sim s'$, then $L(s) = L(s')$. The interesting induction steps are for $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$, where $\phi = \Psi_1 \,\mathsf{U}\, \Psi_2$. Assume that $s \models \Phi$, then there exists a scheduler $\mathfrak{G} : Paths^* \to \mathbb{R} \times Distr(S)$ such that $Paths^{\mathfrak{G}}(s, \Psi_1 \mathsf{U} \Psi_2) = \{\omega \in Paths^{\mathfrak{G}}(s) \mid \exists i \geqslant 0.\ \omega(i,t_i) \models \Psi_2 \wedge \forall 0 \leqslant j < i,\ t < t_j.\ \omega(j,t) \models \Psi_1\}$ and $\Pr(Paths^{\mathfrak{G}}(s, \Psi_1 \mathsf{U} \Psi_2)) \trianglelefteq p$.

Assume $\omega \in Paths^{\mathfrak{G}}(s, \Psi_1 \mathsf{U} \Psi_2)$, according to Lemma 14, there exists a probabilistic time-abstracting bisimilar path $\omega' \in Paths^{\mathfrak{G}'}(s', \Psi_1 \mathsf{U} \Psi_2)$, and vice versa. We can thus construct a scheduler $\mathfrak{G}' : Paths^* \to \mathbb{R} \times Distr(S)$ as follows: for $\omega \in Paths^*(s)$ and its bisimilar path $\omega' \in Paths^*(s')$, if $\mathfrak{G}(\omega) = (\mu, t)$, then $\mathfrak{G}'(\omega') = (\mu', t')$ and $\mu \equiv_\sim \mu'$.

It remains to show that $\Pr(Paths^{\mathfrak{G}}(s, \Psi_1 \mathsf{U} \Psi_2)) = \Pr(Paths^{\mathfrak{G}'}(s', \Psi_1 \mathsf{U} \Psi_2))$. Due to the fact that for each $\Psi_1 \mathsf{U} \Psi_2$ path, the probability distribution determined by $\mathfrak{G}$ and $\mathfrak{G}'$ is equivalent, the probability measure of the two sets of paths coincides. □

The above theorem states that a PCTL formula is preserved by a PTAB, which indicates that all PCTL properties can be checked on the quotient PTSs, thus all the existing techniques, algorithms, and tools for finite MDPs can be applied.

6 Conclusion

We have investigated probabilistic time-abstracting bisimulation for probabilistic timed automata. This equivalence usually provides a much coarser partition than tradition region equivalence and preserves PCTL. We provided a non-trivial adaptation of the traditional partition-refinement algorithm to compute the quotient under PTAB. This algorithm is symbolic in the sense that equivalence classes are represented as polyhedra.

In future works, we would like to investigate *weak* probabilistic time-abstracting bisimulations, including its definition and decision procedures. Experimental research of the proposed algorithms is to be carried out. Furthermore, it is also interesting to study the abstract-refinement and counterexample generation for PTAs.

Acknowledgement. This research has been financially supported by the Dutch Bsik project BRICKS, the Dutch NWO project QUPES, the EU FP7 project QUASIMODO, and partially supported by the Chinese national 863 program (2007AA01Z178), NSFC (60736015) and JSNSF (BK2006712).

References

[1] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] C. Baier, B. Engelen, and M. E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.*, 60(1):187–231, 2000.

[4] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FSTTCS, LNCS 1026*, pages 499–513, 1995.

[5] S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.

[6] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[7] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *STTT*, 6(2):128–142, 2004.

[8] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, 2002.

[9] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. Comput.*, 205(7):1027–1077, 2007.

[10] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Weak bisimulation for probabilistic timed automata and applications to security. In *SEFM*, pages 34–43, 2003.

[11] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[12] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.