Formal Methods in System Design manuscript No.
(will be inserted by the editor)

An Extension of the Inverse Method to Probabilistic Timed Automata

Étienne André · Laurent Fribourg ·
Jeremy Sproston

Received: date / Accepted: date

Abstract Probabilistic timed automata can be used to model systems in which probabilistic and timing behaviour coexist. Verification of probabilistic timed automata models is generally performed with regard to a single reference valuation π_0 of the timing parameters. Given such a parameter valuation, we present a method for obtaining automatically a constraint K_0 on timing parameters for which the reachability probabilities (1) remain invariant and (2) are equal to the reachability probabilities for the reference valuation. The method relies on parametric analysis of a non-probabilistic version of the probabilistic timed automata model using the "inverse method". The method presents the following advantages. First, since K_0 corresponds to a dense domain around π_0 on which the system behaves uniformly, it gives us a measure of *robustness* of the system. Second, it allows us to obtain a valuation satisfying K_0 which is *as small as possible* while preserving reachability probabilities, thus making the probabilistic analysis of the system easier and faster in practice. We provide examples of the application of our technique to models of randomized protocols, and introduce an extension of the method allowing the generation of a "probabilistic cartography" of a system.

Keywords Probabilistic Model Checking · Parametric Timed Automata

Étienne André
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, F-93430, Villetaneuse, France
Tel.: +33 1 49 40 28 61
Fax: +33 1 48 26 07 12
E-mail: Etienne.Andre@lipn.univ-paris13.fr

Laurent Fribourg
LSV – ENS de Cachan & CNRS, France
Tel.: +33 1 47 40 75 36
Fax: +33 1 47 40 75 21
E-mail: Laurent.Fribourg@lsv.ens-cachan.fr

Jeremy Sproston
Dipartimento di Informatica, Università di Torino, Italy
Tel.: +39 011 6706772
Fax: +39 011 751603
E-mail: sproston@di.unito.it

1 Introduction

Timed automata are finite control automata equipped with *clocks*, which are real-valued variables which increase uniformly at the same rate [1]. This modelling formalism is useful for reasoning about real-time systems, because one can specify quantitatively the interval of time during which the transitions can occur, using the bounds involved in invariants and guards labelling the nodes and arcs of the automaton. An extension of timed automata to the probabilistic framework, where discrete actions are replaced by discrete probability *distributions* over discrete actions, has been defined in [10,17]. This formalism has been applied to a number of case studies [16]. Model-checking analysis of probabilistic timed automata normally proceeds by reducing the model to a finite-state probabilistic system [17,16, 20,15].

The constants used in some timing constraints of a real-time system may not be known, or may be known with some uncertainty. Therefore methods for automatically generating values on parameters in timing constraints for which the system behaves correctly are desirable. Methods for synthesising such parameters in timed automata have first been presented in [2]. In [4], the following *inverse problem* has been considered: given a parametric timed automaton and a *reference valuation* π_0, which is a particular valuation of the parameters of the model, find a constraint K_0 on the parameters which is satisfied by the reference valuation and in which the model behaves in the same manner as in the case of the reference valuation. For example, if π_0 is known to exhibit good behaviour, such as the impossibility of reaching an error state, then our aim is to find a constraint on the parameters within which such good behaviour is guaranteed. In particular, this allows the system designer to optimize some parameters of the system.

In this paper, we introduce *parametric probabilistic timed automata*, which combine probabilistic timed automata and parametric timed automata. We then consider the application of the inverse method to parametric probabilistic timed automata models. In this context, the computed constraint K_0 defines parameter valuations for which, in particular, minimum (resp., maximum) probabilities of satisfying a given property (e.g. reachability of an error state) are all equal. The method presents the following advantages. First, since K_0 corresponds to a dense domain around π_0 on which the system behaves uniformly, it gives us a measure of *robustness* of the system. Second, it usually allows us to change the timing constants of the system to a valuation in K_0 *smaller than* π_0, while maintaining the behaviour of the system, thus making the probabilistic analysis of the system easier and faster in practice. Indeed, probabilistic analyses of timed systems are often performed using an integer-time semantics, in which the performance of the analysis depends critically on the size of the timing constants used in the system. Therefore obtaining small values of such constants is of interest for improving the performance of integer-time analysis. Our approach is also useful for avoiding repeated executions of probabilistic model checking analyses for the same model with different parameter valuations, not only in the case in which we aim to test the robustness of the system, but also in the case in which we aim to understand the way in which different parameter valuations have an impact on the probabilistic behaviour of the system. This is an advantage both for verification based on the integer-time semantics and that based on a continuous-time approach, such as [15].

In order to compute such a constraint K_0, we transform the system into a *non-probabilistic* timed automaton, and apply the original inverse method of [4] to this timed automaton. The justification for this transformation is a pragmatic one: in this way a constraint K_0 can be obtained quickly (usually in a few seconds), without computations concerning probability values, and then can subsequently be used to alleviate the computation of reachability probabilities, which is the bottleneck in the probabilistic model-checking process.

We provide examples of the application of our technique to models of randomized protocols, and show that the computation times of reachability probabilities drastically decrease when applied to much smaller values than π_0 within K_0, when using the integer-time semantics verification approach. Furthermore, by using small parameter values within K_0, the use of the integer-time semantics approach becomes competitive, and in some cases improves on, the continuous-time, game-based verification engine of [15].

We also show that the behavioural cartography introduced in [5] can be applied to probabilistic systems as well, thus allowing us to synthesise a "probabilistic cartography" of a system. As a consequence, the value of the reachability probabilities is uniform in each "tile" or part of the parametric space.

An illustrative example. Consider the CSMA/CD protocol, as studied in the context of probabilistic timed automata in [20]. We consider the case in which there are two stations, 1 and 2, trying to send data at the same time. The overall model is given by the parallel composition of three probabilistic timed automata representing the medium and two stations trying to send data.

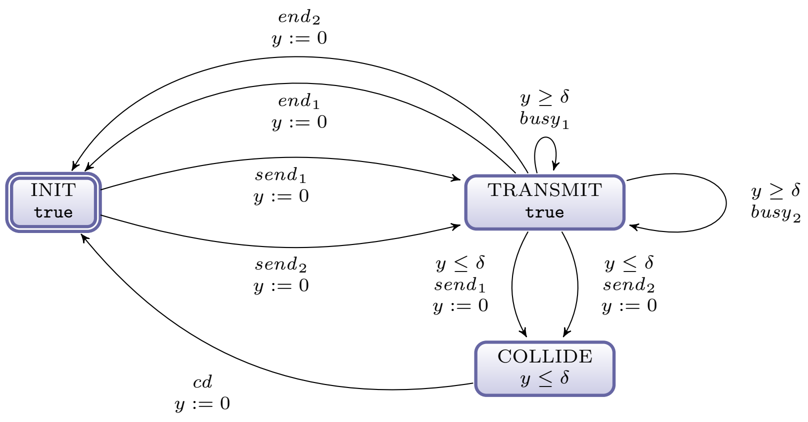

Fig. 1 CSMA/CD Medium

The probabilistic timed automaton representing the medium is given in Figure 1. We use the standard conventions for the graphical representation of timed automata. The medium is initially ready to accept data from any station (event $send_1$ or $send_2$). Once a station, say 1, starts sending its data there is an interval of time (at most δ), representing the time it takes for a signal to propagate between the stations. In this interval the medium can accept data from station 2 (resulting

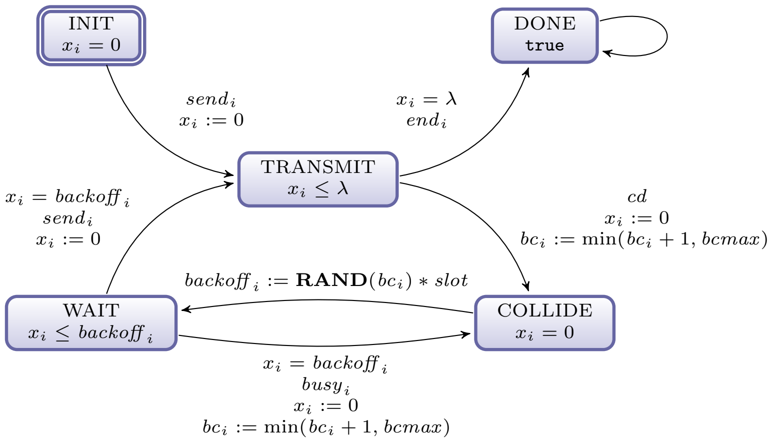

Fig. 2 CSMA/CD Station i

in a collision). After this interval, if station 2 tries to send data it will get the busy signal ($busy_2$). When a collision occurs, there is a delay (again at most δ) before the stations realize there has been a collision, after which the medium will become free (event cd). If the stations do not collide, then when station 1 finishes sending its data (event end_1) the medium becomes idle.

We model the situation in which initially the stations collide. The probabilistic timed automaton representing a station i ($i = 1, 2$) is given in Figure 2. Station i starts by sending its data. If there is no collision, then, after λ time units, the station finishes sending its data (event end_i). On the other hand, if there is a collision (event cd), the station attempts to retransmit the packet, where the scheduling of the retransmission is determined by a truncated binary exponential backoff process. The number of slots (each of length $slot$) that station i waits after the nth transmission failure is chosen as a uniformly distributed *random* integer in the range $0, 1, 2, \dots, 2^{bc_i+1} - 1$, where $bc_i = \min(n, bcmax)$, and $bcmax$ is a fixed upper bound for bc_i (initially $bc_i = n = 0$). This random choice is depicted in Figure 2 by the assignment $backoff_i := \mathbf{RAND}(bc_i) * slot$. Once this time has elapsed, if the medium appears free the station resends the data (event $send_i$), while if the medium is sensed busy (event $busy_i$) the station repeats this process.

We consider in the following that $bcmax$ is a constant equal to 1, and that δ, λ and $slot$ are *parameters*. The reference valuation for these parameters, taken from the IEEE standard 802.3 for 10 Mbps Ethernet, is $\delta = 26$ microseconds, $\lambda = 808$ microseconds, and $slot = 2\delta = 52$ microseconds. The method for inferring a constraint K_0 on the parameters, which is satisfied by the reference valuation and in which the behaviour of the model remains the same, consists in transforming the system into a non-probabilistic parametric timed automaton. We replace the random choice of the number of slots $backoff_i := \mathbf{RAND}(bc_i) * slot$ with a *non-deterministic* choice. The application of the inverse method to the non-probabilistic parametric timed automaton (see Section 5.1) infers for K_0 the following constraint: $(0 < \delta < slot) \wedge (15 slot < \lambda < 16 slot)$. In particular, the minimum and maximum probabilities for a message sent by a station to be transmitted after having collided exactly k times with another message are the same under the reference valuation and any other parameter valuation satisfying K_0. This has two practical implications. Firstly, in order to compute the aforementioned minimum and maximum probabilities for $\delta = 26, \lambda = 808, \mathit{slot} = 52$, it suffices to compute the minimum and maximum probabilities for $\delta = 1, \lambda = 31, \mathit{slot} = 2$ (because both valuations satisfy the constraint $(0 < \delta < \mathit{slot}) \wedge (15\mathit{slot} < \lambda < 16\mathit{slot})$ synthesised by the inverse method). Note that, with the valuation $\delta = 26, \lambda = 808, \mathit{slot} = 52$, and considering the property of transmitting after exactly 10 collisions, our discrete-time model has 101337 states, whereas the valuation $\delta = 1, \lambda = 31, \mathit{slot} = 2$ results in a discrete-time model with 4861 states. The second practical implication concerns the case in which the system designer wishes to understand the behaviour of the system, in terms of minimum and maximum probabilities, for a number of parameter valuations. The approach of obtaining such information by changing manually the timing parameters and repeating model-checking analysis is potentially time consuming. Instead, the application of the inverse method shows that the minimum and maximum probabilities remain invariant for all parameter valuations satisfying the constraint K_0.

This paper is an extended and improved version of [7]. Firstly, Proposition 1 of [7] has been modified, in order to take into account a situation not considered in [7]. We define criteria so that our results hold, and give syntactic conditions for the model to satisfy these criteria. Secondly, the section on experiments has been improved, and the use of the latest version of IMITATOR [6] dramatically reduces the computation time of one of the case studies. Finally, we also extend the behavioural cartography [5] to the probabilistic case.

Comparison to related work. As noted in [11], parameter synthesis of probabilistic models has received scant attention. Lanotte *et al.* [21] consider parametric discrete-time Markov chains (DTMCs), and establish minimal (and maximal) parameter values for the probabilities associated with transitions in order to ensure reachability properties. Daws [9] also consider DTMCs in which the transition probabilities are parameters. Han *et al.* [11] study continuous-time Markov chains in which the average speed (rate) of state changes are parameters. In contrast to [21,9,11], we consider here the model of probabilistic timed automata. The parameters correspond to the timings that appear in guards of transitions and invariants of locations. Such a parametric framework of probabilistic timed automata appears in [8], but the model there did not feature non-deterministic choice. In contrast, our model here features *both* nondeterministic and probabilistic choice.

Plan of the paper. In Section 2, we present the definition of probabilistic timed automata. In Section 3, we present the definition of parametric probabilistic timed automata. In Section 4, we apply the inverse method to probabilistic timed automata in the following way: we construct a non-probabilistic version of the model, which is then subject to the inverse method for parametric timed automata. In Section 5, we apply the method to two probabilistic protocols with timing parameters (CSMA/CD and IEEE 802.11 WLAN). In Section 6, we iteratively apply this extension of the inverse method to compute a probabilistic cartography of the system. We conclude in Section 7.

2 Probabilistic Timed Automata

2.1 Preliminaries

Let $\mathbb{R}_{\geq 0}$ be the set of non-negative real numbers. A (discrete) probability *distribution* over a countable set Z is a function $\mu : Z \to [0,1]$ such that $\sum_{z \in Z} \mu(z) = 1$. We define $\mathsf{support}(\mu) = \{z \in Z \mid \mu(z) > 0\}$. Then for an uncountable set Z we define $\mathsf{Dist}(Z)$ to be the set of functions $\mu : Z \to [0,1]$, such that $\mathsf{support}(\mu)$ is a countable set and μ restricted to $\mathsf{support}(\mu)$ is a (discrete) probability distribution. A *point distribution* is a distribution $\mu \in \mathsf{Dist}(Z)$ such that $\mu(z) = 1$ for some (unique) $z \in Z$. Often we write μ_z for the point distribution such that $\mu(z) = 1$.

Let V be a set of variables of the form $V = \{v_1, \ldots, v_N\}$. An *inequality on the variables of* V is $e \prec e'$, where $\prec \in \{<, \leq\}$, and e, e' are two linear terms of the form $\sum_{1 \leq i \leq N} \alpha_i v_i + d$ where $v_i \in V$, $\alpha_i \in \mathbb{N}$, for $1 \leq i \leq N$, and $d \in \mathbb{N}$. A *constraint on the variables of* V is a conjunction of inequalities on the variables of V.

2.2 Timed Probabilistic Systems

We review the definition of timed probabilistic systems, as defined in [17], which are variants of Segala's probabilistic timed automata [22]. A *timed probabilistic system (TPS)* is a tuple $\mathsf{T} = (S, S_0, \mathit{Act}, \Rightarrow)$ where S is a set of *states*, including a set S_0 of *initial states*, Act is a finite set of *actions* (disjoint from $\mathbb{R}_{\geq 0}$), and $\Rightarrow \subseteq S \times \mathbb{R}_{\geq 0} \times \mathit{Act} \times \mathsf{Dist}(S)$ is a *probabilistic transition relation.* We assume that the probabilistic transition relation is *total*; that is, for every state $s \in S$, there exists $(s, d, a, \mu) \in \Rightarrow$ for some $d \in \mathbb{R}_{\geq 0}$, $a \in \mathit{Act}$ and $\mu \in \mathsf{Dist}(S)$.

A transition $s \xrightarrow{d,a,\mu} s'$ is made from a state $s \in S$ by first nondeterministically selecting a duration-action-distribution triple (d, a, μ) such that $(s, d, a, \mu) \in \Rightarrow$, and second by making a probabilistic choice of target state s' according to distribution μ, such that $\mu(s') > 0$. A *path* of a TPS is a non-empty finite sequence of transitions $\omega = s_0 \xrightarrow{d_0,a_0,\mu_0} s_1 \xrightarrow{d_1,a_1,\mu_1} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} s_n$. Given a path $\omega = s_0 \xrightarrow{d_0,a_0,\mu_0} s_1 \xrightarrow{d_1,a_1,\mu_1} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} s_n$, we let $\mathit{last}(\omega) = s_n$. The *length* of ω, denoted by $|\omega|$, is the number of transitions in ω. The set of paths of a TPS T is denoted by $\mathit{Path}^{\mathsf{T}}$. When clear from the context we omit the superscript T and write Path. We let $\mathit{Path}(s)$ denote the set of paths commencing in the state $s \in S$.

A *scheduler* is a function which resolves nondeterminism by choosing, based on the path executed so far, an outgoing transition in the last state of the path. Formally, a scheduler of a TPS is a function σ such that, for each path ω of the TPS, if $\sigma(\omega) = (d, a, \mu)$ then $(\mathit{last}(\omega), d, a, \mu) \in \Rightarrow$. Intuitively, given a scheduler σ of a TPS, after taking a finite path ω, the TPS will be in state s in the next step with probability $\mu(s)$, where $\sigma(\omega) = (d, a, \mu)$. We denote the set of paths induced by a given scheduler σ to be $\mathit{Path}^{\sigma} = \{\omega = s_0 \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} s_n \mid \sigma(\mathsf{pref}(\omega, i)) = (d_i, a_i, \mu_i) \textit{ for all } i < n\}$, where $\mathsf{pref}(\omega, i)$ returns the prefix of ω up to length i. Then we define $\mathit{Path}^{\sigma}(s) = \mathit{Path}^{\sigma} \cap \mathit{Path}(s)$. For each $s \in S$ and scheduler σ, we can define the probability measure $\mathit{Prob}^{\sigma}_{s}$ over measurable sets of paths in the standard way [14].

2.3 Probabilistic Timed Automata

Probabilistic timed automata [10,17] are an extension of classical timed automata [1] with discrete probability distributions, and can be used to model probabilistic real-time systems, such as timed randomized protocols or fault-tolerant systems. This probabilistic extension adds discrete probability distributions over edges, so that the choice of the next location of the automaton is not only nondeterministic, but now also probabilistic.

2.3.1 Syntax

A *clock* is a variable x_i which takes values in $\mathbb{R}_{\geq 0}$. All clocks evolve linearly at the same rate. We denote a set of clocks by $X = \{x_1, \dots, x_H\}$. We define a *clock valuation* as a function $w : X \to \mathbb{R}_{\geq 0}$ assigning a non-negative real value to each clock. We will often identify a valuation w with the point $(w(x_1), \dots, w(x_H)) \in \mathbb{R}^H_{\geq 0}$. For $d \in \mathbb{R}_{\geq 0}$, we write $w + d$ to denote the valuation such that $(w + d)(x) = w(x) + d$ for all $x \in X$. Given a clock valuation w and a set $\rho \subseteq X$ of clocks, we denote by $\rho(w)$ the clock valuation such that $\rho(w)(x) = 0$ if $x \in \rho$ and $\rho(w)(x) = w(x)$ otherwise.

A *constraint on clocks* X is a constraint on the set of variables X. Given a constraint D on the clocks and a clock valuation w, we obtain $D[w]$ by replacing each clock x in D with $w(x)$. A clock valuation w *satisfies* constraint D (denoted by $w \models D$) if $D[w]$ evaluates to `true`.

A *probabilistic timed automaton (PTA)* A is a tuple $\mathsf{A} = (\Sigma, Q, \overline{q}, X, I, prob)$, where:

- Σ is a finite set of *actions*,
- Q is a finite set of *locations* with an *initial location* $\overline{q} \in Q$,
- X is a set of *clocks*,
- I is the *invariant* function, assigning to every $q \in Q$ a constraint $I(q)$ on the clocks X, and
- *prob* is the *probabilistic edge relation* consisting of elements of the form (q, g, a, η), where $q \in Q$, g is a constraint on the clocks X, $a \in \Sigma$, and $\eta \in \mathsf{Dist}(2^X \times Q)$.

We use the following conventions for the graphical representation of probabilistic timed automata: locations are represented by nodes, within which name and the invariant of the location is written; probabilistic edges are represented by arcs from locations, labelled by the associated guard and event, and which split into multiple arcs, each of which leads to a location and which is labelled by a set of clocks to be reset to 0 and a probability (probabilistic edges which correspond to probability 1 are illustrated by a single arc from location to location).

Example 1 Consider the PTA shown in Figure 3 (left). From location q_0, one can choose nondeterministically between actions a and b. When choosing action b, one can reach location q_1 (with probability 1) if the guard $x \geq 1$ is satisfied. When choosing action a, if the guard $x \geq 2 \wedge x \leq 3$ is satisfied, then one can reach location q_1 and reset x with probability 1/3, reach location q_2 and reset x to 0 with probability 1/3, or reach location q_3 and reset x and y to 0 with probability 1/3. The rest of this probabilistic timed automaton does not feature probabilistic edges, and can be explained in a similar way as for timed automata.

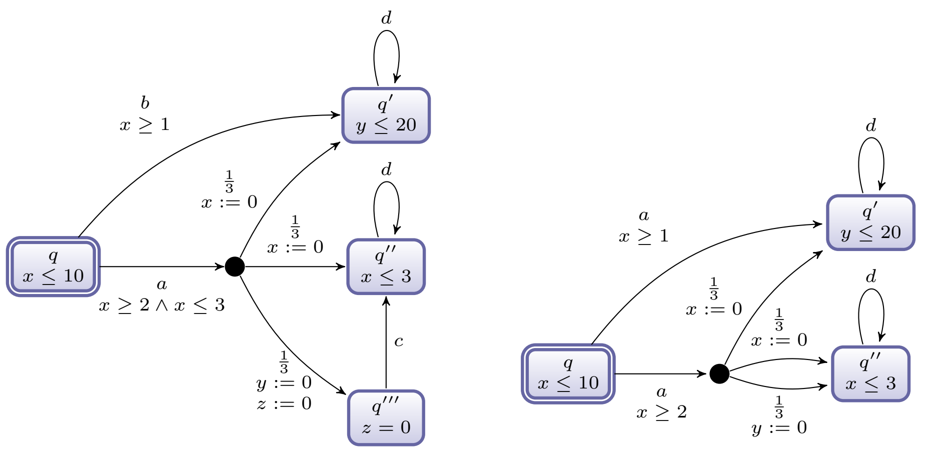

Fig. 3 Examples of PTAs satisfying the assumptions of determinism on actions and reset unicity (left) and satisfying neither assumption (right)

Networks of PTAs can be defined by using parallel composition based on the synchronization of discrete transitions of different components sharing the same action [19].

Syntactic assumptions on PTAs. We make the following syntactic assumptions on probabilistic timed automata in order to simplify both the presentation and the proofs of later results.

Determinism on actions: Given a location $q \in Q$ and action $a \in \Sigma$, there is at most one probabilistic edge of the form $(q, \_, a, \_) \in prob$.
Reset unicity: For any probabilistic edge $(q, g, a, \eta) \in prob$ and location $q' \in Q$, there exists at most one $\rho \in 2^X$ such that $\eta(\rho, q') > 0$.

Example 2 In Figure 3 we give an example of a PTA which satisfies neither determinism on actions nor reset unicity (right), and an example of a PTA which satisfies both assumptions (left). The PTA on the right does not satisfy determinism on actions, because there are two probabilistic edges labelled by a exiting the location q; it also does not satisfy the assumption of reset unicity, because the lower probabilistic edge has two distinct probabilistic alternatives which lead to location q''.

The assumptions of determinism on actions and reset unicity are commonly met in practice, and they simplify the proofs of our subsequent results. For the remainder of this paper, we assume that all of the PTAs we consider satisfy the assumptions of determinism on actions and reset unicity.

2.3.2 Semantics

In this section, we will consider the PTA $\mathsf{A} = (\Sigma, Q, \overline{q}, X, I, prob)$. A *state* of A is a pair $(q, w) \in Q \times (X \to \mathbb{R}_{\geq 0})$ such that $w \models I'(q)$. Informally, the behaviour of A can be understood as follows. The model starts in the initial location $\overline{q}$ with all clocks set to 0. In this, and any other state (q, w), there is a nondeterministic choice of (1) the amount of time which then passes, and (2) which discrete transition is subsequently taken. Note that, for point (1), time can pass only if invariant $I'(q)$ is satisfied while time elapses. Furthermore, for point (2), a discrete transition can be made according to any probabilistic edge $(q, g, a, \eta) \in \mathit{prob}'$ with source location q which is *enabled*; that is the constraint g is satisfied by the current clock valuation w. Then the probability of moving to the location q' and resetting all of the clocks in ρ to 0 is given by $\eta(\rho, q')$.

A PTA can be interpreted as an infinite TPS. Due to the continuous nature of clocks, the underlying TPS has uncountably many states, and is uncountably branching. A PTA can thus be considered as a *finite* description of infinite TPS. Let $\mathsf{A} = (\Sigma, Q, q_0, X, I, \mathit{prob})$ be a PTA. The *semantics* of A is the TPS $\mathsf{T}_\mathsf{A} = (S, S_0, \Sigma, \Rightarrow)$ with $S = \{(q, w) \in Q \times (X \to \mathbb{R}_{\geq 0}) \mid w \models I(q)\}$, $S_0 = \{(\overline{q}, \mathbf{0})\}$ where $\mathbf{0}(x) = 0$ for all $x \in X$, and where $((q, w), d, a, \mu) \in \Rightarrow$ if both of the following conditions hold :

Time elapse: $w + d \models I(q)$;
Edge traversal: there exists a probabilistic edge $(q, g, a, \eta) \in \mathit{prob}$ such that $w + d \models g$ and, for each $(\rho, q') \in \mathsf{support}(\eta)$, we have $\mu(q', \rho(w + d)) = \eta(\rho, q')$.

We write Path^A for $\mathit{Path}^{\mathsf{T}_\mathsf{A}}$. Observe that the rule for discrete transitions is a simplified version of the standard rule [17], and which is permitted by the assumption of reset unicity. The definition of T_A also relies on the fact that A satisfies the well-formedness assumption explained below.

Well-formedness assumptions on PTAs. In order to define the notion of well-formedness for PTAs, we introduce below the assumptions of admissible targets and no deadlock.

A PTA has *admissible targets* if whenever a probabilistic edge is enabled, all of the probabilistic alternatives (pairs of target location and clock reset) result in valid states; that is, they do not result in pairs (q, w) in which w does not satisfy $I(q)$. More formally, a PTA is said to have admissible targets if, for each probabilistic edge $(q, g, a, \eta) \in \mathit{prob}$ and state $(q, w) \in S$ such that $w \models g$, we require that $(q', \rho(w)) \in S$ for each $(\rho, q') \in \mathsf{support}(\eta)$ (equivalently, that $\rho(w) \models I(q')$ for each $(\rho, q') \in \mathsf{support}(\eta)$).

Example 3 An example of a PTA which does not have admissible targets is illustrated in Figure 3 (right). It is possible that the value of the clock x exceeds 3 when the lower probabilistic edge from q_0 is taken, in which case, on taking the probabilistic alternative labelled by $y := 0$, the invariant of q_2 is not satisfied. Instead, the PTA on the left-hand side of Figure 3 has admissible targets.

A PTA can be transformed into a PTA with admissible targets by incorporating the invariant associated with the target location into the guard of each probabilistic edge (along the lines of the transformation in [20]).

To guarantee the existence of at least one transition from each state, we assume that the PTA A has *no deadlock*: in all states of A reachable from $(\overline{q}, \mathbf{0})$ (i.e. final states of paths in $\mathit{Path}^{\mathsf{T}_\mathsf{A}}$), it is always possible to take some probabilistic edge, possibly after letting time elapse. This assumption guarantees that the probabilistic transition relation of the associated probabilistic system is total (see Section 2.2).

For the remainder of this paper, we assume that all of the PTAs we consider satisfy the well-formedness assumptions of admissible targets, no deadlock, in addition to the previously introduced assumptions of determinism on actions and reset unicity.

2.4 Time-abstract Trace Distributions

Let $\mathsf{A} = (\Sigma, Q, \overline{q}, X, I, prob)$ be a PTA. Given a path $\omega = (q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} (q_1, w_1) \xrightarrow{d_1,a_1,\mu_1} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$ of A, we let the *time-abstract trace* of ω be the sequence of alternating locations and actions $q_0 a_0 q_1 a_1 \cdots a_{n-1} q_n$. We let $\mathsf{trace} : \mathit{Path} \to (Q \times \Sigma)^*$ be the function associating the time-abstract trace with each path of Path. Then the *time-abstract trace distribution* of σ and state $s \in S$ is the probability measure over traces denoted by td^σ_s defined according to trace and the trace distribution construction of Segala [22]. Although we do not consider the details of the construction of trace distributions in this paper, we note that, for example, the probability assigned by td^σ_s to traces in which a certain location is reached is defined to be the same as the probability assigned by Prob^σ_s to the set of paths in which this location is reached. The set of time-abstract trace distributions of the TPS T_A is denoted by $\mathsf{tdist}(\mathsf{T}_\mathsf{A}) = \{\mathsf{td}^\sigma_s \mid \sigma \text{ is a scheduler of } \mathsf{T}_\mathsf{A} \text{ and } s \in S_0\}$.

Example 4 Consider the PTA of Figure 3 (left). In this case, the uncountable number of schedulers of the PTA results in a finite number of trace distributions. More precisely, the set of time-abstract trace distributions of the PTA comprises td and td', which take the following form: $\mathsf{td}(qbq'w) = 1$, $\mathsf{td}'(qaq'w) = \frac{1}{3}$, $\mathsf{td}'(qaq''w') = \frac{1}{3}$ and $\mathsf{td}'(qaq'''cq''w') = \frac{1}{3}$, where w is a sequence comprised of a finite number of repetitions of dq', and w' is a sequence comprised of a finite number of repetitions of dq''. Intuitively, the time-abstract trace distribution td (td', respectively) corresponds to the class of schedulers which nondeterministically chooses the edge labelled by b (a, respectively) from location q.

3 Parametric Probabilistic Timed Automata

3.1 Definition of Parametric Probabilistic Timed Automata

In this section, we extend the definition of probabilistic timed automata to the parametric case.

Let $P = \{p_1, \ldots, p_M\}$ be a set of *parameters*. A *parameter valuation* π is a function $\pi : P \to \mathbb{R}_{\geq 0}$ assigning a non-negative real value to each parameter. We will often identify a valuation π with the point $(\pi(p_1), \ldots, \pi(p_M)) \in \mathbb{R}^M_{\geq 0}$. A *constraint on the parameters* P (*constraint on the clocks* X *and the parameters* P, respectively) is a constraint on the set of variables P (on the set of variables $X \cup P$, respectively). In the sequel, the letter K (C, respectively) denotes a constraint on the parameters (on the clocks and the parameters, respectively). We consider `true` as a constraint on P, corresponding to the set of all possible values for P.

Given a parameter valuation π and a constraint C, we denote by $C[\pi]$ the constraint obtained by replacing each parameter p in C with $\pi(p)$. Likewise, given a clock valuation w, we denote by $C[\pi][w]$ the expression obtained by replacing each clock x in $C[\pi]$ with $w(x)$. We say that π *satisfies* C, denoted by $\pi \models C$, if the set of clock valuations that satisfy $C[\pi]$ is nonempty. Similarly, we say that π *satisfies* K, denoted by $\pi \models K$, if the expression obtained by replacing each parameter p in K with $\pi(p)$ evaluates to `true`.

The following definition is an extension of the class of probabilistic timed automata to the parametric case. Parametric probabilistic timed automata allow the use of parameters in place of constants within guards and invariants, and are based on parametric timed automata [2]. A *parametric probabilistic timed automaton (PPTA)* $\mathcal{A}$ is a tuple of the form $\mathcal{A} = (\Sigma, Q, \overline{q}, X, P, I, \mathit{prob})$, where:

- Σ is a finite set of *actions*,
- Q is a finite set of *locations* with an *initial location* $\overline{q} \in Q$,
- X is a finite set of clocks,
- P is a finite set of parameters,
- I is the *invariant* function, assigning to every $q \in Q$ a constraint $I(q)$ on the clocks X and the parameters P, and
- prob is the *probabilistic edge relation* consisting of elements of the form (q, g, a, η), where $q \in Q$, g is a constraint on the clocks X and the parameters P, $a \in \Sigma$ and $\eta \in \mathsf{Dist}(2^X \times Q)$.

Given a PPTA $\mathcal{A} = (\Sigma, Q, \overline{q}, X, P, I, \mathit{prob})$ and a parameter valuation π, we denote by $\mathcal{A}[\pi] = (\Sigma, Q, \overline{q}, X, I', \mathit{prob}')$ the PTA obtained by fixing π in the constraints of $\mathcal{A}$: more precisely, for each $q \in Q$, let $I'(q) = I(q)[\pi]$, and let $\mathit{prob}' = \{(q, g[\pi], a, \eta) \mid (q, g, a, \eta) \in \mathit{prob}\}$.

In this paper, given a PPTA $\mathcal{A}$, we will consider only parameter valuations which guarantee that the PTA $\mathcal{A}[\pi]$ exhibits determinism on actions, reset unicity and well-formedness.

Anchored PPTAs. It has been shown that the inverse method can be applied to obtain a constraint on the parameter valuations such that all parameter valuations satisfying the constraint result in equivalent models on time-abstract traces [4]. However, we give an example below in which we show that equivalence on time-abstract traces is not enough to guarantee the equivalence on time-abstract trace distributions. Then we will define a restriction on PPTA so that equivalence on time-abstract traces is enough to guarantee the equivalence of time-abstract trace distributions.

First, we present an example of a PPTA $\mathcal{A}$ and two parameter valuations π_1 and π_2 such that $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_2]$ are time-abstract trace equivalent but not time-abstract trace distribution equivalent. Consider the PPTA $\mathcal{A}$ given in Figure 4.

Consider the following valuations of the parameters:

$$\pi_1 : p_1 = p_2 = p_3 = p_4 = 0$$

and

$$\pi_2 : p_1 = p_2 = p_4 = 1, p_3 = 0.$$

In the following, we consider the maximum probability of reaching q_4 from q_0 with $x = y = 1$. Two cases arise, depending on the valuation of the parameters.

1. For π_1, the aforementioned maximum probability is 1: we leave q_0 with $x = 0$, then we can take edges going to q_4, regardless of whether we are in q_2 or q_3;

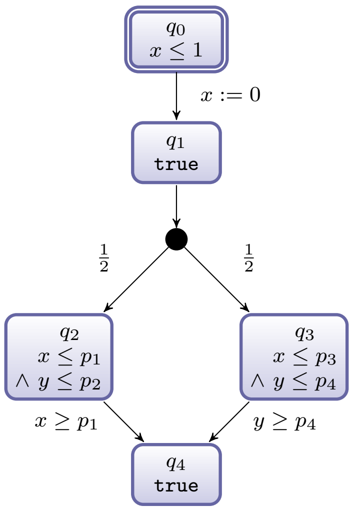

Fig. 4 An example justifying the restriction to a subclass of PPTA in Proposition 1

2. For π_2, the aforementioned maximum probability is 1/2: either we leave q_0 with $x = 0$, then we can take the edge from q_2 to q_4 after letting 1 time unit elapse in q_2, but not the edge from q_3 to q_4; or we leave q_0 with $x = 1$, then can take the edge from q_3 to q_4, but not the edge from q_2 to q_4.

Therefore we have exhibited two different probabilistic timed automata $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_2]$ that are time-abstract trace equivalent (both can perform traces leading to q_4 via the same edges), but that feature different maximum reachability probabilities. Because time-abstract trace distribution equivalent PTA must have the same maximum reachability probabilities, we conclude that $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_2]$ are not time-abstract trace distribution equivalent.

This case was not considered in Proposition 1 in the conference version of this paper [7], which explains the restriction to "anchored PPTAs" (introduced below) that we apply to the forthcoming Proposition 1 of this paper. Before formally stating this restriction, we first introduce some terminology. Let $\mathcal{A} = (\Sigma, Q, \bar{q}, X, P, I, \mathit{prob})$ be a PPTA. A *probability-1 edge* is probabilistic edge $(q, g, a, \eta) \in \mathit{prob}$ such that η is a point distribution. A probabilistic edge which is not a probability-1 edge is called a *probabilistically-branching edge*. A *unique-time probabilistic edge* is a probabilistic edge (q, g, a, η) such that $x \leq \theta$ and $\theta \leq x$ are two of the conjuncts of g for some clock $x \in X$ and $\theta \in \mathbb{N} \cup P$ (hence, we have $x = \theta$). Intuitively, a unique-time probabilistic edge can be taken when the value of a clock equals a particular value (a constant or a parameter value).

Next we introduce some terminology to reason about sequences of probabilistic edges that can potentially be executed consecutively: a *structural path* of $\mathcal{A}$ is a sequence $(q_0, g_0, a_0, \eta_0) \cdots (q_{n-1}, g_{n-1}, a_{n-1}, \eta_{n-1}) q_n$ in $(\mathit{prob})^* Q$ such that $\eta_i(\_, q_{i+1}) > 0$ for $0 \leq i < n$. Structural paths correspond to a path through the graph structure of the PPTA: in the example of Figure 3, $(q, (x \geq 2 \wedge x \leq 3), a, \eta)(q''', \mathtt{true}, c, \eta_{\emptyset,q''})(q'', \mathtt{true}, d, \eta_{\emptyset,q''})q''$ is a structural path, where $\eta(\{x\}, q') = \eta(\{x\}, q'') = \eta(\{y,z\}, q''') = \frac{1}{3}$ and $\eta_{\emptyset,q''}(\emptyset, q'') = 1$, as is $(q'', \mathtt{true}, d, \eta_{\emptyset,q''})(q'', \mathtt{true}, d, \eta_{\emptyset,q''})q''$.

The PPTA $\mathcal{A}$ is *anchored* if there does not exist a structural path $(q_0, g_0, a_0, \eta_0) \cdots (q_{n-1}, g_{n-1}, a_{n-1}, \eta_{n-1})q_n$ such that:

1. $(q_{n-1}, g_{n-1}, a_{n-1}, \eta_{n-1})$ is a probabilistically-branching edge,
2. (q_0, g_0, a_0, η_0) is not a unique-time probabilistic edge, and
3. $\eta_i(X, q_{i+1}) = 0$ for each $0 \leq i < n$.

Condition 3 states that the full clock set X is never reset to 0 along the structural path. Intuitively, in an anchored PPTA, a probabilistically-branching edge cannot be taken after a non-unique-time probabilistic edge without first a reset of all clocks to 0.

Note that the PPTA of Figure 4 is not an anchored PPTA: for example, the probabilistic edge from q_1 is probabilistically branching, but it is not unique time. Note that, even if the probabilistic edge from q_1 is transformed into a unique-time probabilistic edge, the resulting PPTA would not be anchored, because the probabilistic edge from q_0, which precedes that from q_1 in a structural path, is not unique time.

3.2 Time-abstract Trace Distribution Equivalence for Parametric Probabilistic Timed Automata

In this subsection we introduce time-abstract trace distribution equivalence as a means of reasoning whether two parameter values of the same PPTA exhibit equivalent behaviour. Let $\mathcal{A} = (\Sigma, Q, \overline{q}, X, P, I, prob)$ be a PPTA, and let π and π' be valuations of the parameters in P. We say that $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi']$ are *time-abstract trace distribution equivalent*, written $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi']$, if $\mathsf{tdist}(\mathsf{T}_{\mathcal{A}[\pi]}) = \mathsf{tdist}(\mathsf{T}_{\mathcal{A}[\pi']})$. If $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi']$, we can conclude that the TPSs have time-abstract equivalent finite behaviours: for example, they assign the same maximum and minimum probabilities of reaching a certain location [18] (in general, they assign the same maximum and minimum probabilities to linear-time properties on finite traces).

First we introduce some notation. The path $\omega = (q_0, w_0) \xrightarrow{d_0, a_0, \mu_0} \cdots \xrightarrow{d_{n-1}, a_{n-1}, \mu_{n-1}} (q_n, w_n)$ of $\mathsf{T}_{\mathcal{A}[\pi]}$, is *time-abstract path equivalent* to the path $\omega' = (q'_0, w'_0) \xrightarrow{d'_0, a'_0, \mu'_0} \cdots \xrightarrow{d'_{n-1}, a'_{n-1}, \mu'_{n-1}} (q'_n, w'_n)$ of $\mathsf{T}_{\mathcal{A}[\pi']}$, written $\omega \equiv \omega'$, if $q_i = q'_i$, $a_i = a'_i$, and $\mu_i(q_{i+1}, w_i) = \mu'_i(q'_{i+1}, w'_i)$ for all $i = 0, \ldots, n-1$, and $q_n = q'_n$. We extend the notion of time-abstract path equivalence to sets of paths: two sets of paths $\Omega \subseteq Path^{\mathcal{A}[\pi]}$ and $\Omega' \subseteq Path^{\mathcal{A}[\pi']}$ are time-abstract path equivalent, written $\Omega \equiv \Omega'$, if (1) for each path $\omega \in \Omega$, there exists $\omega' \in \Omega'$ such that $\omega \equiv \omega'$, and (2) conversely, for each path $\omega \in \Omega'$, there exists $\omega' \in \Omega$ such that $\omega \equiv \omega'$.

The following result allows us to relate time-abstract equivalence on paths to time-abstract trace distribution equivalence for anchored PPTAs.

Proposition 1 *Let $\mathcal{A}$ be an anchored PPTA, and let π and π' be valuations of parameters P. If $Path^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv Path^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, then $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi']$.*

The proof of Proposition 1 is given in Appendix A.

3.3 Non-probabilistic Version of a PPTA

In this subsection, along the lines of [18,19], we explain how probability values can be abstracted away from a PPTA to result in a non-probabilistic parametric timed automaton. First we explain how a PPTA can be transformed into a PPTA featuring point distributions only. This is done by replacing probabilistic choice within a single probabilistic edge by nondeterministic choice between multiple probabilistic edges, each of which corresponds to a point distribution.

The *non-probabilistic version* of $\mathcal{A} = (\Sigma, Q, \overline{q}, X, P, I, \mathit{prob})$, written $\mathcal{A}^* = (\Sigma, Q, \overline{q}, X, P, I, \mathit{prob}^*)$, is a PPTA which agrees with $\mathcal{A}$ on all elements apart from the probabilistic edge relation: let prob^* be the smallest probabilistic edge relation such that for every edge $(q, g, a, \eta) \in \mathit{prob}$ and $(\rho, q') \in \mathsf{support}(\eta)$, we have $(q, g, a, \eta_{(\rho,q')}) \in \mathit{prob}^*$ (recall that $\eta_{(\rho,q')}$ denotes the point distribution assigning probability 1 to the element (ρ, q')). Observe that the state sets of $\mathsf{T}_{\mathcal{A}[\pi]}$ and $\mathsf{T}_{\mathcal{A}^*[\pi]}$ are equal.

Proposition 2 *Let π be a valuation of P and (q, w) be a state of $\mathsf{T}_{\mathcal{A}[\pi]}$ (and $\mathsf{T}_{\mathcal{A}^*[\pi]}$). For each step $(q, w) \xrightarrow{d,a,\mu} (q', w')$ of $\mathsf{T}_{\mathcal{A}[\pi]}$, there exists the step $(q, w) \xrightarrow{d,a,\mu_{(q',w')}} (q', w')$ of $\mathsf{T}_{\mathcal{A}^*[\pi]}$. Conversely, for each step $(q, w) \xrightarrow{d,a,\mu_{(q',w')}} (q', w')$ of $\mathsf{T}_{\mathcal{A}^*[\pi]}$, there exists a step $(q, w) \xrightarrow{d,a,\mu} (q', w')$ of $\mathsf{T}_{\mathcal{A}[\pi]}$.*

Proposition 2 allows us to obtain a one-to-one mapping between transitions of $\mathcal{A}[\pi]$ and $\mathcal{A}^*[\pi]$. By reasoning inductively, we can extend the proposition to obtain a one-to-one mapping between paths of $\mathcal{A}[\pi]$ and $\mathcal{A}^*[\pi]$. Note that, by the combination of determinism on actions and reset unicity, the probability of the transitions of $\mathcal{A}[\pi]$ is encoded in the actions and target locations of the associated transitions of $\mathcal{A}^*[\pi]$. This implies that, for any pair ω_*, ω'_* of paths such that $\omega_* \in \mathit{Path}^{\mathcal{A}^*[\pi]}(\overline{q}, \mathbf{0})$, $\omega'_* \in \mathit{Path}^{\mathcal{A}^*[\pi']}(\overline{q}, \mathbf{0})$ and $\omega_* \equiv \omega'_*$, we can generate ω and ω' from ω_* and ω'_*, respectively, via the one-to-one mapping between paths of $\mathcal{A}[\pi]$ and $\mathcal{A}^*[\pi]$, such that $\omega \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$, $\omega' \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ and $\omega \equiv \omega'$. Together, these facts allow us to show the following.

Proposition 3 *Let $\mathcal{A}$ be a PPTA, and let π and π' be valuation of parameters P. If $\mathit{Path}^{\mathcal{A}^*[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}^*[\pi']}(\overline{q}, \mathbf{0})$, then $\mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$.*

We note that a PPTA featuring point distributions only has a one-to-one mapping with a classical parametric timed automaton: a probabilistic edge $(q, g, a, \eta_{(\rho,q')})$ of the PPTA corresponds to the edge (q, g, a, ρ, q') of a classical parametric timed automaton. In subsequent sections of the paper, this allows us to apply methods for classical parametric timed automata to PPTAs featuring point distributions.

4 The Inverse Method for PPTAs

4.1 The Inverse Problem

Given an anchored PPTA $\mathcal{A}$ and a valuation π_0 of the parameters, we present in this section a method allowing to synthesise a constraint K_0 on the parameters of $\mathcal{A}$ such that $\pi_0 \models K_0$ and, for all $\pi \models K_0$, $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are time-abstract trace distribution equivalent. As a consequence, the PTAs $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ assign the same maximum and minimum probabilities to linear-time properties on finite traces.

The problem can be stated as follows.

The Inverse Problem for PPTAs
Let $\mathcal{A}$ be an anchored PPTA and π_0 a valuation of the parameters. Find a constraint K_0 such that :

1. $\pi_0 \models K_0$, and
2. $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi_0]$, for all $\pi \models K_0$.

4.2 The Inverse Method on Classical Parametric Timed Automata

In [4], we introduced the *inverse method algorithm IM*, which allows us to solve the following problem: given a (non-probabilistic) parametric timed automaton $\mathcal{T}$ and a reference valuation π_0, $IM(\mathcal{T}, \pi_0)$ synthesises a constraint K_0 such that

1. $\pi_0 \models K_0$, and
2. $Path^{\mathcal{T}[\pi_0]}(\overline{q}, \mathbf{0}) \equiv Path^{\mathcal{T}[\pi]}(\overline{q}, \mathbf{0})$ for all $\pi \models K_0$.

In particular, this method guarantees that, for any $\pi \models K_0$, time-abstract linear time properties are preserved. Note however that *timed* properties, i.e. making use of a deadline inside the property, are not preserved in general because they are not time-abstract. A brief explanation of the algorithm is given in Appendix B.

Recall from [4] that the constraint output by this inverse method *IM* is not (in general) the weakest constraint satisfying this problem. One reason for this is that the constraint is always in conjunctive form. In contrast, the weakest constraint may be in disjunctive form (see [3]).

The inverse method *IM* is a semi-decidable algorithm whose termination has been proven for subclasses of parametric timed automata, and usually terminates in practice. See [3] for a more exhaustive description of the inverse method and its properties.

4.3 The Inverse Method for PPTAs

Given an anchored PPTA $\mathcal{A}$, we can now solve the inverse problem for $\mathcal{A}$ by applying the algorithm *IM* to the non-probabilistic version $\mathcal{A}^*$ of $\mathcal{A}$.

Theorem 1 *Given an anchored PPTA $\mathcal{A}$ and a reference valuation π_0, the constraint K_0 returned by $IM(\mathcal{A}^*, \pi_0)$ solves the inverse problem for anchored PPTAs, i.e.:*

1. $\pi_0 \models K_0$*, and*
2. $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi_0]$*, for all* $\pi \models K_0$.

Proof Since K_0 is a solution of the inverse problem for $\mathcal{A}^*$, we have $Path^{\mathcal{A}^*[\pi]}(\overline{q}, \mathbf{0}) \equiv Path^{\mathcal{A}^*[\pi_0]}(\overline{q}, \mathbf{0})$ for all $\pi \models K_0$; hence, we have by Proposition 3 that $Path^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv Path^{\mathcal{A}[\pi_0]}(\overline{q}, \mathbf{0})$ for all $\pi \models K_0$. From Proposition 1, we conclude that $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi_0]$ for all $\pi \models K_0$. □

Remark. The constraint K_0 output by our method is not the weakest constraint satisfying the inverse problem as defined in Section 4.1 because, as said above, the constraint output by the inverse method defined in [4] is itself not the weakest constraint satisfying the inverse problem for (non-probabilistic) parametric timed automata. We will address this problem in Section 6.

Application to the computation of probabilities. Given an anchored PPTA $\mathcal{A}$ and a valuation π_0 of the parameters, in order to determine the minimum or maximum probability pr of satisfying a linear-time property φ on finite traces of $\mathcal{A}[\pi_0]$, it is sufficient to proceed as follows:

1. Compute $K_0 = \mathit{IM}(\mathcal{A}^*, \pi_0)$;
2. Compute pr (using a probabilistic model-checking tool, such as Prism) for $\mathcal{A}[\pi_1]$, for some $\pi_1 \models K_0$.

As a consequence of Theorem 1, given the computation of K_0 using $\mathit{IM}(\mathcal{A}^*, \pi_0)$, the minimum and maximum probabilities of satisfying linear-time properties on finite traces will be the same in $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_0]$.

An advantage of our method is that one can take π_1 small enough in order to make the computation of probabilistic model-checking tools such as Prism easier, because their performance depends on the size of the state space of the model used as input, which in turn depends on the size of the constants used in the PTA (see Section 5 below for a comparison for various case studies).

5 Application of the Inverse Method to PPTAs: Case Studies

In this section, we show the interest of the inverse method in the context of two case studies. More precisely, we consider two protocols, each modeled as a PPTA, and each with an associated standard reference valuation π_0. Our approach consists of the following two phases:

1. Using the tool Imitator [6], which implements the inverse method in the non-probabilistic framework, we synthesise a constraint K_0 for the *non-probabilistic* version of the protocol.
2. Using the probabilistic model-checking tool Prism [12,23], we compute minimum/maximum probabilities for various reachability properties with regard to a number of parameter valuations. For parameter valuations satisfying K_0, the probabilities computed by Prism are equal (as stated by Theorem 1); we also compute the probabilities for some parameter valuations not satisfying K_0.

We also consider the way in which the obtained constraint K_0 can be used to increase the efficiency of the model-checking process. More precisely, for each case study, we consider the smallest possible integer values for the parameters while satisfying K_0 (in each case study, such a parameter valuation is uniquely defined). First, we compare the time used for verification of properties for such a smallest parameter valuation to other parameter valuations in the context of model checking using Prism and the integer-time semantics for PTAs, and show that savings of a factor of at least 10 can be obtained with respect to the reference valuation (or a rescaled version of the reference valuation, noting that models in which parameters are rescaled by a constant factor are equivalent according to time-abstract trace distribution equivalence). In some cases, the savings are considerably greater. Second, we compare the time used for verification for the integer-time semantics with the smallest parameter valuation to the continuous-time, game-based verification method also implemented in Prism [15]. In both cases, we find that the integer-time semantics with the smallest parameter valuation results in lower verification times than other approaches.

Experiments were performed on an Intel Core 2 Duo with 2GB of RAM.

5.1 CSMA/CD Protocol

We apply here our method to the CSMA/CD Protocol described in Section 1.

Validity of the Example. First, let us explain why this case study satisfies the criterion of anchored PPTAs defined in Section 3.1, by recalling the model of the medium (Figure 1 page 3) and the stations (Figure 2 page 4). The only probabilistic choice is when station i is in location COLLIDE. Such a location can be reached through a cd action, or through a $busy_i$ action.

In the case of a cd action, observe that all three automata synchronise on cd: furthermore, they all reset their clocks while taking this transition. Hence, all three clocks are reset just before the probabilistic choice, which satisfies the criterion.

In the case of a $busy_i$ action, only two automata synchronise (station i and the medium), and they both reset their clock. However, the third clock is not reset. But then the $busy_1$ signal can only be sent after a sequence of actions of the form $cd; \tau_2; send_2; (\tau_1; busy_1)^*$, where τ_i represents the unlabelled transition from location COLLIDE to location WAIT in station i. (The case for $busy_2$ is dual.) All these transitions are either urgent (recall that the invariant of location COLLIDE is $x_i = 0$) or labelled with a guard with a clock equal to a constant (the value of $backoff_i$), which corresponds in both cases to a unique-time probabilistic edge. Noting that a cd action corresponds to resetting all clocks to 0, we conclude that this case also satisfies the criterion of anchored PPTAs.

Parameter synthesis. Recall that we consider the three parameters λ, δ and $slot$. Also recall that the following π_0 is the reference valuation taken from the IEEE standard 802.3 for 10 Mbps Ethernet: $\lambda = 808$ microseconds, $slot = 52$ microseconds and $\delta = 26$ microseconds. As described in Section 4, from a PPTA describing this system, we can then obtain a non-probabilistic parametric timed automaton. Applying Imitator to this non-probabilistic model and the reference valuation π_0, we obtain the following constraint in less than 0.3 seconds:

$$K_0 : \quad 0 < \delta < slot \quad \wedge \quad 15 slot < \lambda < 16 slot.$$

As noted in Section 4, the constraint synthesised by our method is not necessarily the weakest. This constraint is such that $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are time-abstract trace-distribution equivalent, for any $\pi \models K_0$. We consider the following three probabilities:

- $Prob_j$, for $j \in \{1, 2\}$: minimum probability that station 1 transmits its message after exactly j collisions.
- $Prob_{\leq 3}$: minimum probability that station 1 transmits its message with no more than 3 collisions.

Table 1 Results of Prism for the CSMA/CD

Name	λ	*slot*	δ	$\models K_0$	States	Constr.	$Prob_1$	$Prob_2$	$Prob_{<3}$	$= \pi_0$
π_0	808	52	26	yes	36335	10	4	5	7	-
π_1	31	2	1	yes	1746	0.1	0.1	0.1	0.1	yes
π_2	940	60	59	yes	42260	19	6	10	14	yes
π_3	940	60	60	no	42753	23	5	11	16	no
π_4	52	52	26	no	6212	1	0.6	1	2	yes
π_5	404	26	13	yes	18350	3	1	2	2	yes

Computation of probabilities. We apply Prism to the system with the parameters set to different valuations (including π_0). The three probabilities for the reference valuation π_0 are 0.5, 0.38 and 0.97, respectively.

Information concerning the use of the constraint K_0 in the context of the integer-time semantics approach is given in Table 1. The first to fourth columns describe the parameter valuations considered. The fifth column indicates whether the parameter valuation satisfies K_0. The sixth and seventh columns give the number of states corresponding to the valuation in the integer-time semantics, together with the time in seconds required by Prism for the construction of the model. The following three columns give the time in seconds used by Prism to perform verification of the three considered properties.[1] For the parameter valuations π_1 to π_5, the final column indicates whether the probabilities computed for the properties are the same as those computed for π_0.

From the results of Section 4, the probabilities for π_0, π_1 and π_2 are identical. Instead, for π_3, in which the constraint K_0 is violated by considering the limit case where $\delta = slot$, the probabilities of the properties are 0, 0.19 and 0.61, respectively. A further observation is that, even if the value of λ violates K_0 (see, e.g. π_4), the probabilities can remain the same as for π_0.

The smallest integer parameter valuation satisfying K_0 is π_1. We observe that the use of π_1 in the PPTA model of the system results in a smaller state space, and reduces verification times for all three properties, in comparison to the other parameter valuations considered. It is instructive to compare the results for π_1 with those for π_5, which is a rescaling of the reference valuation π_0 by a factor of $\frac{1}{2}$: in this case the construction and verification times for π_1 are at least a factor of 10 better than those for π_5. Furthermore, the verification times for the integer-time semantics with parameter valuation π_1 were similar to those for the continuous-time, game-based verification engine of Prism applied to the model with parameter valuation π_0 (the results for π_1 are also similar, which agrees with the widely-recognized fact that continuous-time verification methods, using specialized data structures, can help to alleviate the state-space blow up caused by the magnitude of constants used in the guards and invariants). However, when considering a more realistic model in which the upper bound *bcmax* on the variables used by the stations to count the number of collisions is greater than 1, verification using the integer-time semantics with parameter valuation π_1 can be significantly faster than that using the continuous-time engine. We consider the cases in which *bcmax* is equal to 5 or to 9, in both cases using Imitator to verify that the model with π_1 is equivalent to the model with π_0 (for $bcmax = 9$ we used a 2.66GHz Intel Core I7 with 4GB, which used 199 seconds to compute the constraint), and the property

[1] The verification engine used was the sparse matrix engine.

$\mathit{Prob}_{\leq 3}$. For *bcmax* equal to 5, the integer-time semantics with π_1 uses in total 1.2 seconds (including construction time), whereas the continuous-time engine used 5.6 seconds. For *bcmax* equal to 9, the integer-time semantics with π_1 uses in total 98 seconds (including construction time), whereas the continuous-time engine did not terminate within 2 hours (we also note that the integer-time semantics for *bcmax* equal to 9, but with π_5, terminated after 117 minutes, and only when using the MTBDD engine of PRISM, with the other engines running out of memory).

5.2 IEEE 802.11 Wireless Local Area Network Protocol

We also applied our method to the IEEE 802.11 Wireless Local Area Network Protocol.

Validity of the Example. This case study is obviously an anchored PPTA (defined in Section 3.1) because all transitions of the synchronised model are labelled with at least one clock equality (a clock constrained to be equal to a parameter). By recursion (the clocks are initially set to 0), all paths feature a unique-time probabilistic edge, and the criterion is satisfied.

Parameter synthesis. We consider the following valuation π_0 of the parameters corresponding to the IEEE 802.11 standard and given, e.g. in [23,18].[2] (Timing values are given in microseconds.)

$$\begin{array}{llll} \mathit{ASLOTTIME} = 50 & \mathit{DIFS} = 128 & \mathit{VULN} = 48 & \mathit{TRANSTIME} = 224 \\ \mathit{ACKTO} = 300 & \mathit{ACK} = 205 & \mathit{SIFS} = 28 & \end{array}$$

As in the case of the CSMA/CD protocol, the probabilistic behaviour of this case study concerns a backoff process used in the case of simultaneous transmission.

Taking a parametric timed automaton version of the model and the parameter valuation π_0 as input, IMITATOR computes the following constraint K_0 in 13 s:

$$\begin{array}{lrcl} & \mathit{VULN} + \mathit{TRANSTIME} & > & 5 * \mathit{ASLOTTIME} \\ \wedge & \mathit{DIFS} & > & 2 * \mathit{ASLOTTIME} \\ \wedge & \mathit{ASLOTTIME} & > & \mathit{VULN} \\ \wedge & \mathit{ASLOTTIME} & > & \mathit{VULN} + \mathit{TRANSTIME} \\ \wedge & \mathit{ACK} & > & 4 * \mathit{ASLOTTIME} \\ \wedge & 3 * \mathit{ASLOTTIME} & > & \mathit{DIFS} \\ \wedge & 4 * \mathit{ASLOTTIME} + \mathit{VULN} & > & \mathit{ACK} \\ \wedge & 6 * \mathit{ASLOTTIME} & = & \mathit{ACKTO} \\ \wedge & 6 * \mathit{ASLOTTIME} & = & \mathit{VULN} + \mathit{TRANSTIME} + \mathit{SIFS} \end{array}$$

[2] The only difference with regard to [23,18] is the use of a single parameter *TRANSTIME* for the length of a packet transmission, instead of lower and upper bounds on this length, namely *TRANSTIMEMIN* and *TRANSTIMEMAX*, respectively. This simplifies the model with no consequence, since *TRANSTIMEMAX* had no incidence on the (time-abstract) behaviour of the system, and was only constrained to be greater or equal to *TRANSTIMEMIN*. An advantage of considering a single transmission time is that the model trivially satisfies the criterion of anchored PPTAs. Furthermore, in contrast to [23,18], we set the upper limit of the backoff counter to 1.

Table 2 Results of PRISM for the IEEE 802.11 Protocol

Name	p_1	p_2	p_3	p_4	p_5	p_6	p_7	States	Constr.	$Prob_2$	$Prob_3$
π_0	50	128	48	224	300	205	28	1,671,933	173	585	692
π_1	50	128	48	75	300	205	28	1,527,254	121	459	553
π_2	2	5	1	5	12	1	1	117,510	0.76	2.09	2.45
π_3	4	10	2	10	24	2	2	169,881	1.22	4.55	5.31

Computation of probabilities. We consider the maximum probability that either station's collision counter reaches k, for $k = 2, 3$, as considered in [18]. The results of the application of PRISM are given in Table 2, where the probabilities are denoted by $Prob_i$ for $k = 2, 3$. The parameters $p_1, p_2, \dots, p_7$ stand for *ASLOTTIME*, *DIFS*, *VULN*, *TRANSTIME*, *ACKTO*, *ACK*, *SIFS* respectively. The significance of the columns is similar to that for Table 1. Note that all the parameter valuations considered satisfy K_0, and therefore the computed probabilities for all parameter valuations are the same: the probabilities $Prob_2$ and $Prob_3$ are 0.0625 and 0.001953125, respectively. Note that π_2 corresponds to one of the smallest possible integer valuations according to K_0: the computation time in this case is dramatically decreased compared to, e.g. π_0, thus showing the interest of our method.

Remark. This model features "asap" transitions, viz. transitions that must be fired as soon as the corresponding guard is satisfiable. Unfortunately, IMITATOR does not support the use of such "asap" transitions. However, in this particular example, the "asap" semantics is not strictly needed, as all these "asap" transitions have a guard equal to true. Hence, we implemented them using transitions that must be fired immediately after entering their source location. This was done in a straightforward manner by adding an extra clock reset to 0 in any transition leading to such a location and letting the invariant condition of the location constrain the value of the extra clock to 0.

6 Cartography of Probabilistic Timed Automata

In this section, we address the following weakness of the inverse method: Given an anchored PPTA $\mathcal{A}$ and a valuation π_0, the constraint $K_0 = IM(\mathcal{A}^*, \pi_0)$ may not be the largest set of parameter valuations solving the inverse problem.

We presented in [5] an algorithm iterating the inverse method in the framework of non-probabilistic parametric timed automata. This algorithm allows us to cover (part of) the parametric space with *behavioural tiles*, i.e. constraints for which the sets of time-abstract traces are uniform. Formally, a constraint K is said to be a *behavioural tile* (or more simply a *tile*) if, for all $\pi_1, \pi_2 \in K$, the sets of time-abstract traces of $\mathcal{A}[\pi_1]$ and $\mathcal{A}[\pi_2]$ are equal. Examples of tiles are constraints synthesised by the inverse method; in that case, tiles are always convex.

In this section, we give details about the extension of this algorithm to the probabilistic case mentioned in [5]. We first briefly recall the Behavioural Cartography Algorithm (Section 6.1) and describe its extension to the probabilistic framework (Section 6.2).

6.1 The Behavioural Cartography Algorithm

We briefly recall the Behavioural Cartography Algorithm, defined in [5] in the framework of (non-probabilistic) parametric timed automata. This algorithm relies on the idea of covering the parametric space within a rectangular real-valued parameter domain V_0. By iterating the inverse method IM described in Section 4.1 over all the *integer* valuations of the rectangle V_0 (of which there are a finite number), one is able to decompose the parametric space included into V_0 into a list *Tiling* of behavioural tiles. We recall this algorithm $BC(\mathcal{A}, V_0)$ in Appendix C.

In practice, not only the integer valuations of V_0 are covered by *Tiling*, but also most of the real-valued space of V_0. Furthermore, the space covered by *Tiling* often largely exceeds the limits of V_0. We can show in particular that, for a rectangle V_0 large enough and a grid tight enough, the full coverage of the whole real-valued parametric space (inside and outside V_0) is ensured for some classes of parametric timed automata, in particular for acyclic systems (see [3] for details).

6.2 Extending the Cartography to the Probabilistic Framework

Using the Behavioural Cartography Algorithm and the application of the inverse method to PPTAs described in Section 4, we can construct a cartography of a PPTA $\mathcal{A}$. This can be done in a straightforward manner by applying the algorithm BC to the non-probabilistic version $\mathcal{A}^*$ of $\mathcal{A}$.

From Theorem 1, we have the following proposition.

Proposition 4 *Let $\mathcal{A}$ be an anchored PPTA and let V_0 be a rectangle. Let Tiling $= BC(\mathcal{A}^*, V_0)$. Then for all tiles $K \in$ Tiling, for all $\pi, \pi' \models K$, $\mathcal{A}[\pi] \approx^{\mathsf{tdist}} \mathcal{A}[\pi']$.*

Given a reachability property, one can then construct a *probabilistic cartography* of the system. Formally, given an anchored PPTA $\mathcal{A}$, a rectangle V_0 and a linear temporal logic property on finite traces φ:

1. Compute $Tiling = BC(\mathcal{A}^*, V_0)$;
2. For each tile $K \in Tiling$, select $\pi \models K$, and compute the minimum or maximum probability pr of satisfying φ in $\mathcal{A}[\pi]$ (using, e.g. Prism).

An advantage of the cartography algorithm is that, if one wants to consider another property φ', one can keep *Tiling* as computed in step 1, and only perform again step 2. Only the value of the considered probability in each tile changes, leading to different partitions into good and bad subspaces.

Note also that, as we proposed in general for the inverse method, for the sake of efficiency when using a probabilistic model-checking tool, one should rather select in step 2 a "small" π for each tile K (i.e. a valuation with small constants).

7 Final Remarks

In this paper we have shown that the inverse method presented in [4] can be applied, not just to non-probabilistic parametric timed automata, but also to their probabilistic extension, for proving time-abstract properties. The method relies on the conversion of PPTAs to non-probabilistic parametric timed automata, then on the application of the inverse method of [4]. To our knowledge, no other method allows the synthesis of constraints on the parameters within which the values of reachability probabilities are preserved.

We envisage that the main benefit of this work to be twofold. First, the inverse method may be applied as a (generally fast) non-probabilistic pre-processing step before integer-time semantics verification, given that it can be used to obtain parameter values smaller than those of the reference valuation, which reduces the size of the integer-time PTA models prior to probabilistic model checking. In certain cases, this makes possible probabilistic verification of systems which cannot be model checked directly, due to the prohibitive size of the constants of the reference valuation. Furthermore, recall that, for the CSMA/CD case study that we considered, the use of a "small" parameter valuation with integer-time semantics verification yielded results that were competitive with and often better than those obtained using the continuous-time engine of Prism.

Second, the method gives information concerning the *robustness* of the considered PPTA: we guarantee that the minimum and maximum probabilities computed for π_0 are the same for other valuations "around" π_0, i.e. for any valuation π within K_0. More generally, we have shown that we can obtain a (sometimes partial) covering of the space of parameter valuations with tiles in which probabilities of satisfying properties are equal: this can allow us to obtain a more complete view of the effect of differences between parameter valuations on the probabilities of satisfying properties without necessarily having to repeat probability computations for a large number of parameter valuations. This point applies both to the case in which the underlying verification method uses the integer-time semantics and to the case in which a continuous-time approach is used.

As future work, we aim at relaxing the restriction of anchored PPTAs. This will allow us in particular to consider case studies such as the Root Contention Protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus, considered in the parametric framework in [13]. Indeed, although the condition of anchored PPTAs does not apply to this case study, the application of the inverse method to this example yields correct results in practice.

Let us finally point out that, in [18,19,16], another class of properties, named "soft deadline properties", is treated: for example, the minimum probability of a station delivering a packet *within some deadline*. Such properties are not "time-abstract", and fall beyond the class of those considered here. We plan to explore soft deadline properties in future work.

Acknowledgements We are grateful to the anonymous referees for their helpful comments. Étienne André and Laurent Fribourg have been partially supported by the Agence Nationale de la Recherche, grant ANR-06-ARFU-005, and by Institute Farman (project SIMOP). Jeremy Sproston is supported in part by the project AMALFI - Advanced Methodologies for the AnaLysis and management of the Future Internet (Università di Torino/Compagnia di San Paolo).

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2), 183–235 (1994) 2, 7

2. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, STOC '93, pp. 592–601. ACM, New York, NY, USA (1993) 2, 11
3. André, É.: An inverse method for the synthesis of timing parameters in concurrent systems. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France (2010) 15, 21
4. André, É., Chatain, Th., Encrenaz, E., Fribourg, L.: An inverse method for parametric timed automata. International Journal of Foundations of Computer Science **20**(5), 819–836 (2009) 2, 3, 11, 15, 16, 21, 22
5. André, É., Fribourg, L.: Behavioral cartography of timed automata. In: A. Kučera, I. Potapov (eds.) Proceedings of the 4th Workshop on Reachability Problems in Computational Models (RP'10), *Lecture Notes in Computer Science*, vol. 6227, pp. 76–90. Springer, Brno, Czech Republic (2010) 3, 5, 20, 21
6. André, É., Fribourg, L., Kühne, U., Soulat, R.: IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In: 18th International Symposium on Formal Methods (FM'12), Lecture Notes in Computer Science. Springer, Paris, France (2012). To appear 5, 16
7. André, É., Fribourg, L., Sproston, J.: An extension of the inverse method to probabilistic timed automata. In: M. Roggenbach (ed.) AVoCS'09, *Electronic Communications of the EASST*, vol. 23. European Association of Software Science and Technology, Swansea, UK (2009) 5, 12
8. Chamseddine, N., Duflot, M., Fribourg, L., Picaronny, C., Sproston, J.: Computing expected absorption times for parametric determinate probabilistic timed automata. In: Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08), pp. 254–263. IEEE Computer Society Press, Saint-Malo, France (2008) 5
9. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: Proc. ICTAC'04, *LNCS*, vol. 3407, pp. 280–294. Springer (2004) 5
10. Gregersen, H., Jensen, H.E.: Formal design of reliable real time systems. Master's thesis, Department of Mathematics and Computer Science, Aalborg University (1995) 2, 7
11. Han, T., Katoen, J.P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: Proc. RTSS'08, pp. 173–182. IEEE (2008) 5
12. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: TACAS'06, *LNCS*, vol. 3920, pp. 441–444. Springer (2006) 16
13. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.: Linear parametric model checking of timed automata. Journal of Logic and Algebraic Programming (2002) 22
14. Kemeny, J.G., Snell, J.L., Knapp, A.W.: Denumerable Markov Chains, 2nd edn. Graduate Texts in Mathematics. Springer (1976) 6, 24
15. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: FORMATS'09, *LNCS*, vol. 5813, pp. 212–227. Springer (2009) 2, 3, 17
16. Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Form. Methods Syst. Des. **29**, 33–78 (2006) 2, 22
17. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theoretical Computer Science **282**, 101–150 (2002) 2, 6, 7, 9
18. Kwiatkowska, M., Norman, G., Sproston, J.: Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In: Proc. PAPM/PROBMIV'02, *LNCS*, vol. 2399, pp. 169–187. Springer (2002) 13, 14, 19, 20, 22
19. Kwiatkowska, M., Norman, G., Sproston, J.: Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. Formal Aspects of Computing **14**(3), 295–318 (2003) 8, 14, 22
20. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Information and Computation **205**(7), 1027–1077 (2007) 2, 3, 9
21. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. Formal Aspects of Computing **19**(1), 93–109 (2007) 5
22. Segala, R.: Modeling and verification of randomized distributed real-time systems. Ph.D. thesis, Massachusetts Institute of Technology (1995) 6, 10
23. PRISM Web page: Prism web page. `http://www.prismmodelchecker.org/` 16, 19

A Proof of Proposition 1

In order to prove Proposition 1, we show that, for any scheduler σ of $\mathsf{T}_{\mathcal{A}[\pi]}$, we can construct a scheduler σ' of $\mathsf{T}_{\mathcal{A}[\pi']}$ such that σ and σ' generate the same time-abstract trace distributions (from the initial state). For this task, we require a number of preliminary definitions and results. First, we present a sufficient condition for two schedulers to generate the same time-abstract trace distributions. Recall that, given that we assume reset unicity, for all of the distributions $\mu \in \mathsf{Dist}(Q \times (X \to \mathbb{R}_{\geq 0}))$ we consider in the transition relation of $\mathsf{T}_{\mathcal{A}[\pi]}$ and $\mathsf{T}_{\mathcal{A}[\pi']}$, for each location q there will be at most one clock valuation w such that $\mu(q, w) > 0$. We will use w_q^{μ} to denote this clock valuation. In the following, given two distributions $\mu, \mu' \in \mathsf{Dist}(Q \times (X \to \mathbb{R}_{\geq 0}))$, we write $\mu \simeq \mu'$ if, for each $q \in Q$, we have $\mu(q, w_q^{\mu}) = \mu'(q, w_q^{\mu'})$. Given a triple $(d, a, \mu) \in \mathbb{R}_{\geq 0} \times \Sigma \times \mathsf{Dist}(Q \times (X \to \mathbb{R}_{\geq 0}))$, we let $dist(d, a, \mu) = \mu$.

Lemma 1 *Let σ be a scheduler of $\mathsf{T}_{\mathcal{A}[\pi]}$ and σ' be a scheduler of $\mathsf{T}_{\mathcal{A}[\pi']}$. If $dist(\sigma(\omega)) \simeq dist(\sigma(\omega'))$ for each $\omega \in Path^{\sigma}(\overline{q}, \mathbf{0})$ and $\omega' \in Path^{\sigma'}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \omega'$, then $\mathsf{td}^{\sigma}_{(\overline{q},\mathbf{0})} = \mathsf{td}^{\sigma'}_{(\overline{q},\mathbf{0})}$.*

Proof The scheduler σ induces a Markov chain M^{σ} (see [14]), the states of which are finite paths (starting from $(\overline{q}, \mathbf{0})$), and the transition matrix of which assigns to a transition from path ω to path $\omega \xrightarrow{d,a,\mu} (q, w)$ probability $\mu(q, w)$ if $\sigma(\omega) = (d, a, \mu)$ (probability 0 is assigned to transitions from ω to paths not resulting from ω by appending the choice of $\sigma(\omega)$). Similarly, scheduler σ' induces a Markov chain $\mathsf{M}^{\sigma'}$. The Markov chains M^{σ} and $\mathsf{M}^{\sigma'}$ are isomorphic: that is, given a bijection $f : Path^{\sigma}(\overline{q}, \mathbf{0}) \to Path^{\sigma'}(\overline{q}, \mathbf{0})$ such that $f(\omega) = \omega'$, where ω' is the unique path of $Path^{\sigma'}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \omega'$, we have that the Markov chain obtained from M^{σ} by substituting each $\omega \in Path^{\sigma}(\overline{q}, \mathbf{0})$ by $f(\omega)$ (in the state space and transition matrix) is equal to $\mathsf{M}^{\sigma'}$. Because f preserves traces (that is, $\mathsf{trace}(\omega) = \mathsf{trace}(f(\omega))$), we can then derive that $\mathsf{td}^{\sigma}_{(\overline{q},\mathbf{0})} = \mathsf{td}^{\sigma'}_{(\overline{q},\mathbf{0})}$. □

Recall that the assumption of determinism on actions implies that, for any transition $(q, w) \xrightarrow{d,a,\mu} (q', w')$, the probabilistic edge $(q, \_, a, \_) \in prob$ associated with the transition is unique. A transition $(q, w) \xrightarrow{d,a,\mu} (q', w')$ is a *unique-time transition* if the probabilistic edge $(q, \_, a, \_) \in prob$ is a unique-time probabilistic edge. Similarly, a transition $(q, w) \xrightarrow{d,a,\mu} (q', w')$ is a *probability-1 transition* if the probabilistic edge $(q, \_, a, \_) \in prob$ is a probability-1 edge, otherwise it is a *probabilistically-branching transition*. A state (q, w) is *clock-0 state* if $w = \mathbf{0}$. The next lemma follows immediately from the definition of anchored PPTAs.

Lemma 2 *Let $\omega = (q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$ be a path in either $Path^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ or $Path^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$. Then there do not exist indices $0 \leq i < j \leq n$, determining the sub-path $\omega = (q_i, w_i) \xrightarrow{d_i,a_i,\mu_i} \cdots \xrightarrow{d_{j-1},a_{j-1},\mu_{j-1}} (q_j, w_j)$ such that (1) $(q_{j-1}, w_{j-1}) \xrightarrow{d_{j-1},a_{j-1},\mu_{j-1}} (q_j, w_j)$ is a probabilistically-branching transition, (2) $(q_i, w_i) \xrightarrow{d_i,a_i,\mu_i} (q_{i+1}, w_{i+1})$ is not a unique-time transition, and (3) (q_k, w_k) is not a clock-0 state for each $i \leq k < j$.*

The following lemma states that $\equiv$ preserves the "type" of transitions (where by "type" we mean unique-time transition/non-unique-time transition and probability-1/probabilistically branching transition), and follows immediately from the definition of $\equiv$.

Lemma 3 *Let $\omega = (q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$ be a path in $Path^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ and let $\omega' = (q'_0, w'_0) \xrightarrow{d'_0,a_0,\mu'_0} \cdots \xrightarrow{d'_{n-1},a_{n-1},\mu'_{n-1}} (q'_n, w'_n)$ be a path in $Path^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$. Then if $\omega \equiv \omega'$, we have that the i-th transition $(q_{i-1}, w_{i-1}) \xrightarrow{d_{i-1},a_{i-1},\mu_{i-1}} (q_i, w_i)$ of ω is a unique-time transition (probability-1 transition, respectively) if and only if the i-th transition $(q'_{i-1}, w'_{i-1}) \xrightarrow{d'_{i-1},a_{i-1},\mu'_{i-1}} (q'_i, w'_i)$ of ω is a unique-time transition (probability-1 transition, respectively), for $1 \leq i \leq n$.*

In the following, for any path $\omega = (q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$ and any $0 \leq i \leq n$, we recall that $\mathsf{pref}(\omega, i)$ is the path prefix $(q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{i-1},a_{i-1},\mu_{i-1}} (q_i, w_i)$ comprising the transitions up to the $(i+1)$-th state. We also write $\mathsf{suf}(\omega, i)$ to denote the path suffix $(q_i, w_i) \xrightarrow{d_i,a_i,\mu_i} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$ comprising the transitions from the $(i+1)$-th state (as previously, we also refer to states as being paths of length 0, so $\mathsf{pref}(\omega, 0)$ is (q_0, w_0) and $\mathsf{suf}(\omega, n)$ is (q_n, w_n)). For $0 \leq i \leq j \leq n$, we write $\omega_{i..j}$ for the path $(q_i, w_i) \xrightarrow{d_i,a_i,\mu_i} \cdots \xrightarrow{d_{j-1},a_{j-1},\mu_{j-1}} (q_j, w_j)$. We use $\omega(i)$ to denote (q_i, w_i), for $0 \leq i \leq n$. We say that a path ω' is an *extension* of a path ω if $\omega = \mathsf{pref}(\omega', i)$ for some $0 \leq i \leq |\omega'|$.

Henceforth, we assume that $\mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$. Given that we will construct the scheduler σ' of $\mathcal{A}[\pi']$ by induction on the length of paths, we need to avoid blocking situations in which the paths of σ' replicate the paths of σ (in the sense of having the same time-abstract traces) only up to a certain path length, from which point at least one path of σ cannot be replicated by σ. For example, consider the path ω of σ and the path ω' of σ' such that $\omega \equiv \omega'$; our aim is to define σ' so that it replicates the choice $\sigma(\omega) = (d, a, \mu)$ in the sense of choosing some (d', a, μ') such that $\mu \simeq \mu'$. The problematic situation, that we must avoid during the construction of σ', is that in which, from $\mathit{last}(\omega')$, no transition of the form (d', a, μ') can be taken because the guard g of the probabilistic edge $(q, g, a, \_)$ cannot be enabled from $\mathit{last}(\omega')$ after letting time pass. The next technical lemma explains how this situation is avoided in the case of non-unique-time transitions: it states that, for any path ω of $\mathcal{A}[\pi]$ ending in a sequence of non-unique-time transitions, *any* path of $\mathcal{A}[\pi']$ that is time-abstract equivalent to a prefix of ω which ends in the sequence of non-unique-time transitions can be extended to a path of $\mathcal{A}[\pi']$ that is time-abstract equivalent to the entire path ω.

Lemma 4 *Let σ be a scheduler of $\mathrm{T}_{\mathcal{A}[\pi]}$ and let ω be a path of σ for which the last transition is not a unique-time transition. Let $0 \leq i < |\omega|$ be the smallest i such that $\mathsf{suf}(\omega, i)$ comprises only non-unique-time transitions. Let ω' be a path of $\mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that $\mathsf{pref}(\omega, i) \equiv \omega'$. Then there exists a path $\hat{\omega}' \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that (1) $\mathsf{pref}(\hat{\omega}', i) = \omega'$ and (2) $\omega \equiv \hat{\omega}'$.*

Proof Observe that, because $\mathcal{A}$ is an anchored PPTA, any path of either $\mathcal{A}[\pi]$ or $\mathcal{A}[\pi']$ cycles through the following phases: visit to a clock-0 state, then a (possibly empty) sequence of unique-time transitions, then a (possibly empty) sequence of non-unique-time transitions, then a visit to a clock-0 state, etc. Let $0 \leq j \leq i$ be the largest j such that $\omega(j)$ is a clock-0 state. Then from $\mathsf{pref}(\omega, i) \equiv \omega'$, we have that $\omega'(j)$ is a clock-0 state. Furthermore, $\mathsf{suf}(\omega', j)$ contains only unique-time transitions, which follows from the following facts: $\omega_{i..j}$ contains only unique-time actions, $\omega_{i..j} \equiv \mathsf{suf}(\omega', j)$, and Lemma 3.

Now, from $\mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, we have that the existence of the path $\omega \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ implies the existence of a path $\tilde{\omega} \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \tilde{\omega}$. Let $\hat{\omega}' = \omega' \cdot \mathsf{suf}(\tilde{\omega}, i)$ (where, in the usual manner, $\omega' \cdot \mathsf{suf}(\tilde{\omega}, i)$ denotes the concatenation of ω' and $\mathsf{suf}(\tilde{\omega}, i)$). Then $\hat{\omega}' \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, from the following facts.

First, note that $\hat{\omega}'(j)$ is a clock-0 state (from $\mathsf{pref}(\omega, i) \equiv \omega'$ and the fact that $\omega(j)$ is a clock-0 state).

Second, because the fragment of the path ω from point j to point i (that is, $\omega_{i..j}$) contains only unique-time transitions, together with the fact that $\omega \equiv \omega'$ and Lemma 3, we have that $\omega'_{i..j}$ contains only unique-time transitions. Furthermore, note that, after a clock-0 state followed by a sequence of unique-time transitions, there is only one possible clock valuation: this clock valuation is determined completely by the sequence of unique-time transitions.

From these facts, we can arrive at the following conclusion: after the fragment of ω' from point j to point k, there is only one possible clock valuation for the state $\omega'(k)$, and that $\omega'(k) = \tilde{\omega}(k)$. Intuitively, this means that if $\mathsf{suf}(\tilde{\omega}, i)$ is a possible extension of the path $\tilde{\omega}$ from point i, then $\mathsf{suf}(\tilde{\omega}, i)$ is a also possible extension of the path ω'. This allows us to conclude that $\tilde{\omega} \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ implies $\hat{\omega}' \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$. With regard to the two further conditions on $\hat{\omega}'$ given in the lemma, we note that condition (1) ($\mathsf{pref}(\hat{\omega}', i) = \omega'$) follows immediately from the definition of $\hat{\omega}'$, and condition (2) ($\omega \equiv \hat{\omega}'$) follows from the fact that we assume in the statement of the lemma that $\mathsf{pref}(\omega, i) \equiv \omega'$, and from the fact that $\omega \equiv \tilde{\omega}$ implies trivially that $\mathsf{suf}(\omega, i) \equiv \mathsf{suf}(\tilde{\omega}, i)$. □

Let ω be a path of σ for which the last transition is not a unique-time transition. Let ω' be a path of $\mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that $\mathsf{pref}(\omega, i) \equiv \omega'$ and where $0 \leq i < |\omega|$ be the smallest i such that $\mathsf{suf}(\omega, i)$ comprises only non-unique-time transitions. Lemma 4 allows us to choose a particular $\langle\!\langle\omega\rangle\!\rangle_{\omega'} \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, which depends on ω and ω', such that (1) $\mathsf{pref}(\langle\!\langle\omega\rangle\!\rangle_{\omega'}, i) = \omega'$ and (2) $\omega \equiv \langle\!\langle\omega\rangle\!\rangle_{\omega'}$.

We now proceed to the proof of Proposition 1. In the standard way, given $\omega = (q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n)$, we write $\omega \xrightarrow{d,a,\mu} (q, w)$ to denote the path $(q_0, w_0) \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n, w_n) \xrightarrow{d,a,\mu} (q, w)$. In the following, we write $(\omega \xrightarrow{d,a,\mu}) \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ if there exists some state (q, w) such that $\omega \xrightarrow{d,a,\mu} (q, w) \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$; analogous notation is used for $\mathcal{A}[\pi']$.

Proof (Proposition 1) By Lemma 1, it suffices to show the following result: for any scheduler σ of $\mathsf{T}_{\mathcal{A}[\pi]}$, we can construct a scheduler σ' of $\mathsf{T}_{\mathcal{A}[\pi']}$ such that, for each each $\omega \in \mathit{Path}^{\sigma}(\overline{q}, \mathbf{0})$ and $\omega' \in \mathit{Path}^{\sigma'}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \omega'$, we have $\mathit{dist}(\sigma(\omega)) \simeq \mathit{dist}(\sigma(\omega'))$.

We proceed the construction of σ' by considering paths of progressively greater length. In the following, we let $\mathit{Path}^{\sigma}_i(\overline{q}, \mathbf{0})$ be the set of paths of $\mathit{Path}^{\sigma}(\overline{q}, \mathbf{0})$ of length i; similarly, $\mathit{Path}^{\sigma'}_i(\overline{q}, \mathbf{0})$ denotes the set of paths of $\mathit{Path}^{\sigma'}(\overline{q}, \mathbf{0})$ of length i.

Let $i \geq 0$. Assume that we have defined σ' for all paths of $\mathit{Path}^{\sigma'}_j(\overline{q}, \mathbf{0})$ for all $0 \leq j < i$. Now we define σ' for paths of $\mathit{Path}^{\sigma'}_i(\overline{q}, \mathbf{0})$ Let $\omega \in \mathit{Path}^{\sigma}_i(\overline{q}, \mathbf{0})$ be a path of $\mathcal{A}[\pi]$ of length i, and let $\omega' \in \mathit{Path}^{\sigma'}_i(\overline{q}, \mathbf{0})$ be the unique (by determinism on actions) path of $\mathcal{A}[\pi']$ of length i such that $\omega \equiv \omega'$. Let $\sigma(\omega) = (d, a, \mu)$. Our aim is to show the existence of $(\mathit{last}(\omega'), d', a, \mu')$ in the probabilistic transition relation of $\mathsf{T}_{\mathcal{A}[\pi']}$ such that $\mu \simeq \mu'$. Then we let $\sigma'(\omega') = (d', a, \mu')$.

In the case in which $\mathit{last}(\omega)$ is a clock-0 state, we proceed as follows. We note that, from $\mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, the existence of $(\omega \xrightarrow{d,a,\mu}) \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ implies the existence $(\tilde{\omega} \xrightarrow{d',a,\mu'}) \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \tilde{\omega}$ and $\mu \simeq \mu'$. Given that $\omega \equiv \tilde{\omega}$ and $\omega \equiv \omega'$, and that $\mathit{last}(\omega)$ is a clock-0 state, we must have that $\mathit{last}(\omega) = \mathit{last}(\tilde{\omega}) = \mathit{last}(\omega')$. In this case it is immediate to see that the fact that $(\mathit{last}(\tilde{\omega}), d', a, \mu')$ is in the probabilistic transition relation of $\mathsf{T}_{\mathcal{A}[\pi']}$ implies that $(\mathit{last}(\omega'), d', a, \mu')$ is in the probabilistic transition relation of $\mathsf{T}_{\mathcal{A}[\pi']}$. Hence we let $\sigma'(\omega') = (d', a, \mu')$. From $\mu \simeq \mu'$, it follows that $\mathit{dist}(\sigma(\omega)) \simeq \mathit{dist}(\sigma(\omega'))$.

Now we consider the case in which $\mathit{last}(\omega)$ is not a clock-0 state. We consider two sub-cases.

Sub-case: the last transition of ω is a unique-time transition. Given that $\mathcal{A}$ is an anchored PPTA and from Lemma 2, there exists $0 \leq j < i$ such that $\omega(j)$ is a clock-0 state and $\mathsf{suf}(\omega, j)$ contains only unique-time transitions.

From $\mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0}) \equiv \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, the existence of the path $(\omega \xrightarrow{d,a,\mu}) \in \mathit{Path}^{\mathcal{A}[\pi]}(\overline{q}, \mathbf{0})$ implies the existence of a path $(\tilde{\omega} \xrightarrow{d',a,\mu'}) \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ such that $\omega \equiv \tilde{\omega}$ and $\mu \simeq \mu'$. Now consider $\mathsf{suf}(\omega, j)$ and $\mathsf{suf}(\tilde{\omega}, j)$. Observe that only unique-time transitions feature along $\mathsf{suf}(\tilde{\omega}, j)$ (this follows from the fact that $\mathsf{suf}(\omega, j)$ contains only unique-time transitions, from the fact that $\omega \equiv \tilde{\omega}$ implies that $\mathsf{suf}(\omega, j) \equiv \mathsf{suf}(\tilde{\omega}, j)$, and from Lemma 3). Given that $\mathsf{suf}(\tilde{\omega}, j)$ is a clock-0 state, and that $\mathsf{suf}(\tilde{\omega}, j)$ features only unique-time transitions, it must be the case that, for each state visited along $\mathsf{suf}(\tilde{\omega}, j)$, there is only one possible clock valuation. Hence we must have $\mathsf{suf}(\omega', j) = \mathsf{suf}(\tilde{\omega}, j)$. This implies that $\mathit{last}(\omega') = \mathit{last}(\tilde{\omega})$. Given that the existence of $(\tilde{\omega} \xrightarrow{d',a,\mu'}) \in \mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$ implies that $(\mathit{last}(\tilde{\omega}), d', a, \mu')$ is in the probabilistic transition relation of $\mathsf{T}_{\mathcal{A}[\pi']}$, it follows trivially that $(\mathit{last}(\omega'), d', a, \mu')$ is in the probabilistic transition relation of $\mathsf{T}_{\mathcal{A}[\pi']}$. Hence we let $\sigma'(\omega') = (d', a, \mu')$.

Sub-case: the last transition of ω is not a unique-time transition. Given that $\mathcal{A}$ is an anchored PPTA and from Lemma 2, there exists $0 \leq j < i$ such that $\mathsf{suf}(\omega, j)$ contains only non-unique-time transitions.

First, suppose that there exists some path of σ that is an extension of ω and which ends in a clock-0 state; then let $\omega{\uparrow}^{\sigma}_{0}$ be the shortest such path. Given that the last transition of ω is not a unique-time transition, by Lemma 2, the last transition of $\omega{\uparrow}^{\sigma}_{0}$ is not a unique-time transition. Given that $\mathsf{pref}(\omega, k) \equiv \mathsf{pref}(\omega', k)$, we can employ Lemma 4 to define the path $\langle\!\langle\omega{\uparrow}^{\sigma}_{0}\rangle\!\rangle_{\mathsf{pref}(\omega',k)}$: the path $\langle\!\langle\omega{\uparrow}^{\sigma}_{0}\rangle\!\rangle_{\mathsf{pref}(\omega',k)}$ is in $\mathit{Path}^{\mathcal{A}[\pi']}(\overline{q}, \mathbf{0})$, extends $\mathsf{pref}(\omega', k)$, and is

such that $\omega{\uparrow}_0^\sigma \equiv \langle\!\langle \omega{\uparrow}_0^\sigma \rangle\!\rangle_{\mathsf{pref}(\omega',k)}$. Let $(q,w) \xrightarrow{d',a,\mu'} (q',w')$ be the $(i+1)$-th transition of $\langle\!\langle \omega{\uparrow}_0^\sigma \rangle\!\rangle_{\mathsf{pref}(\omega',k)}$. Then we let $\sigma'(\omega') = (d',a,\mu')$. From the fact that $\omega{\uparrow}_0^\sigma \equiv \langle\!\langle \omega{\uparrow}_0^\sigma \rangle\!\rangle_{\mathsf{pref}(\omega',k)}$, we have that $\mu \simeq \mu'$ (in fact, because $(\mathit{last}(\omega),d,a,\mu)$ and $(\mathit{last}(\omega'),d',a,\mu')$ are not unique-time transitions, we must have $\mu(q') = \mu'(q') = 1$).
Alternatively, suppose that there does not exist a path of σ which extends ω and which ends in a clock-0 state. Note that, by the definition of anchored PPTAs, this means that all paths of σ that are extensions of ω feature only non-unique-time (and hence probability-1) transitions. Hence we can conclude the following: all paths of σ that are extensions of ω are of the form $\overline{\omega} \xrightarrow{d,a,\mu_{(q,w)}} (q,w)$, where $\sigma(\overline{\omega}) = (d,a,\mu_{(q,w)})$ and $\overline{\omega}$ is either ω itself or a path of σ that is an extension of ω. These extensions of ω derive a countably infinite sequence of paths progressively extending ω. We can also find a countably infinite sequence of paths progressively extending ω', given the definition of σ' up to ω', such that each extension of ω' is equivalent under $\equiv$ to the associated extension of ω with the same length. This sequence of paths is obtained by considering each extension of ω and applying Lemma 4. This countably infinite sequence defines the transitions chosen by σ' for any extension of ω'. It can then be seen that, for any extension of ω under σ, and any $\equiv$-equivalent extension of ω' under σ', the distributions in the transitions of σ and σ' are $\simeq$-equivalent.

Given Lemma 1, we have completed the proof of Proposition 1. □

B The Inverse Method

Given a (classical) parametric timed automaton $\mathcal{A}$ and a reference valuation π of parameters, the inverse method outputs a constraint K such that:

1. $\pi \models K$,
2. $\mathit{Path}^{\mathcal{A}[\pi]} \equiv \mathit{Path}^{\mathcal{A}[\pi']}$, for all $\pi' \models K$.

The algorithm *IM* can be summarized as follows. Starting with $K := \mathtt{true}$, we iteratively compute a growing set of reachable symbolic states. A symbolic state of the system is a couple (q,C), where q is a location of $\mathcal{A}$, and C a constraint on the clocks and the parameters. When a π-*incompatible* state (q,C) is encountered (i.e. when $\pi \not\models C$), K is refined as follows: a π-incompatible inequality J (i.e. such that $\pi \not\models J$) is selected within C, and $\neg J$ is added to K. The procedure is then started again with this new K, and so on, until no new reachable state is computed.

Algorithm 1: $\mathit{IM}(\mathcal{A},\pi)$

```
input  : A parametric timed automaton $\mathcal{A}$ of initial state $s_0$
input  : Valuation $\pi$ of the parameters
output: Constraint $K$ on the parameters
$i \leftarrow 0$ ;   $K \leftarrow$ true ;   $S \leftarrow \{s_0\}$
while true do
    while there are $\pi$-incompatible states in $S$ do
        Select a $\pi$-incompatible state $(q,C)$ of $S$ (i.e. s.t. $\pi \not\models C$) ;
        Select a $\pi$-incompatible $J$ in $(\exists X : C)$ (i.e. s.t. $\pi \not\models J$) ;
        $K \leftarrow K \wedge \neg J$ ;
        $S \leftarrow \bigcup_{j=0}^{i} \mathit{Post}^{j}_{\mathcal{A}(K)}(\{s_0\})$ ;
    if $\mathit{Post}_{\mathcal{A}(K)}(S) \sqsubseteq S$ then return $K \leftarrow \bigcap_{(q,C)\in S}(\exists X : C)$
    $i \leftarrow i+1$ ;
    $S \leftarrow S \cup \mathit{Post}_{\mathcal{A}(K)}(S)$ ;                // $S = \bigcup_{j=0}^{i} \mathit{Post}^{j}_{\mathcal{A}(K)}(\{s_0\})$
```

The algorithm *IM* is given in Figure 1. Given a linear inequality J of the form $e < e'$ (resp. $e \leq e'$), the expression $\neg J$ denotes the negation of J and corresponds to the linear inequality $e' \leq e$ (resp. $e' < e$). Given a constraint C on the clocks and the parameters, the expression

$\exists X : C$ denotes the constraint on the parameters obtained from C after elimination of the clocks.

We define $\mathcal{A}(K)$ as $\{\mathcal{A}[\pi] \mid \pi \models K\}$, $Post^{i}_{\mathcal{A}(K)}(S)$ as the set of states reachable from S in exactly i steps, and $Post^{*}_{\mathcal{A}(K)}(S)$ as the set of all states reachable from S in $\mathcal{A}(K)$ (i.e. $Post^{*}_{\mathcal{A}(K)}(S) = \bigcup_{i \geq 0} Post^{i}_{\mathcal{A}(K)}(S)$). Given two sets of states S and S', we write $S \sqsubseteq S'$ iff $\forall s \in S, \exists s' \in S'$ s.t. $s = s'$.

C The Behavioural Cartography Algorithm

Algorithm 2: Behavioural Cartography Algorithm $BC(\mathcal{A}, V_0)$

```
input  : A parametric timed automaton $\mathcal{A}$, a finite rectangle $V_0 \subseteq \mathbb{R}^M_{\geq 0}$
output: Tiling: list of tiles (initially empty)

repeat
    select an integer valuation $\pi \in V_0$;
    if $\pi$ does not belong to any tile of Tiling then
        Add $IM(\mathcal{A}, \pi)$ to Tiling;
until Tiling contains all the integer valuations of $V_0$ ;
```

Imitator also implements the behavioural cartography algorithm in a fully automated way.