Electronic Communications of the EASST
Volume 23 (2009)

Proceedings of the Ninth International Workshop on Automated Verification of Critical Systems (AVOCS 2009)

An Extension of the Inverse Method to Probabilistic Timed Automata

Étienne André, Laurent Fribourg, Jeremy Sproston

18 pages

Guest Editor: Markus Roggenbach
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

An Extension of the Inverse Method to Probabilistic Timed Automata

Étienne André[1], **Laurent Fribourg**[1], **Jeremy Sproston**[2]

[1] LSV – ENS de Cachan & CNRS, France
[2] Dipartimento di Informatica, Università di Torino, Italy

Abstract: Probabilistic timed automata can be used to model systems in which probabilistic and timing behavior coexist. Verification of probabilistic timed automata models is generally performed with regard to a single reference valuation of the timing parameters. Given such a parameter valuation, we present a method for obtaining automatically a constraint 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". Our approach is useful for avoiding repeated executions of probabilistic model checking analyses for the same model with different parameter valuations. We provide examples of the application of our technique to models of randomized protocols.

Keywords: Probabilistic Model Checking, Parametric Timed Automata, Randomized Protocols

1 Introduction

Timed automata are finite control automata equipped with *clocks*, which are real-valued variables which increase uniformly [1]. This model 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 labeling 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 [14, 17]. This model 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 and then employing a probabilistic model-checking tool such as PRISM [12, 23].

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 synthesizing such parameters in timed automata have first been presented in [2]. In [3], the following *inverse problem* has been considered: given a parametric timed automaton and a *reference valuation*, 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 the

reference valuation is known to exhibit good behavior, such as the impossibility of reaching an error state, then our aim is to find a constraint on the parameters within which such good behavior is guaranteed. In particular, this allows the system designer to optimize some parameters of the system.

In this paper, we consider the application of the inverse method to *probabilistic* timed automata models. We aim at synthesizing a constraint such that, for any valuation of the parameters satisfying K_0, the model is "time-abstract" equivalent to the model for the reference valuation. In the context of probabilistic timed automata, 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 a certain location) are all equal. Therefore, given the computation of K_0, it suffices to compute a minimum (resp., maximum) probability for a single parameter valuation satisfying K_0. In order to infer such a constraint K_0, we transform the system into a *non-probabilistic* timed automaton, and apply the original inverse method of [3] to this timed automaton. As in the method of [18], some attention has to be dedicated to the non-probabilistic model construction so that the results of the inverse method apply to the original probabilistic timed automaton.

Motivation. This approach is particularly important for probabilistic timed automata for the following reason. As mentioned above, model checking for probabilistic timed automata in practice generally relies on the reduction of the model to a finite-state system. The effectiveness of the discrete-time semantics method most commonly used is sensitive to the timing constants used in the description of the model: more precisely, the greater the timing constants, the larger the state space of the finite-state system obtained from the discrete-time semantics construction, and the more difficult the verification (the zone-based algorithm of [20] does not have this property, but does not always perform better in practice than the discrete-time semantics approach). In case studies it is standard to rescale the time unit used in the model to reduce the magnitude of the timing constraints in order to reduce the size of the resulting finite-state system. This rescaling operation possibly involves rounding lower bounds on clocks downwards, and upper bounds upwards [16], which results in an abstraction in which the computed maximum (minimum, respectively) probabilities may be greater (or less than, respectively) the actual probabilities in the original model. The inverse method presents an alternative to this rescaling approach. By applying the inverse method to obtain the constraint K_0, we can choose the parameter valuation satisfying K_0 with the lowest possible values for the timing constraints: then, by performing analysis on the model with this parameter valuation, we can obtain the same minimum and maximum probabilities as on the model using the reference valuation of the parameters. Hence, as in the rescaling approach, we can obtain discrete-time models of limited size by reducing the magnitude of the timing constants; however, in contrast to the rescaling approach, we can avoid rounding of lower and upper bounds, thereby resulting in a model exhibiting the same probabilities as obtained for the model corresponding to the reference valuation. We note that this motivation also applies when considering discrete-time approaches to timed automata verification, such as in [11, 7, 6].

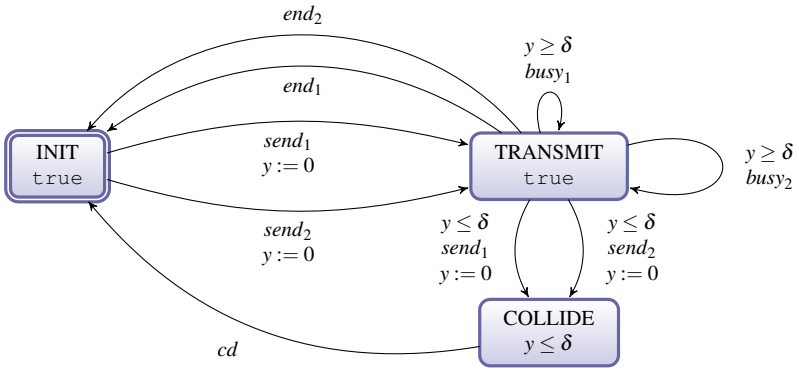

Figure 1: CSMA/CD Medium

Comparison to Related Work. In contrast to [21, 9, 10], we do not consider parameters over probabilities, but only over timing constraints. A parametric probabilistic timed automata framework which did not feature non-deterministic choice was considered in [8]. In contrast, the framework we introduce in this paper features *both* nondeterministic and probabilistic choice.

An Illustrative Example. Consider the CSMA/CD protocol, as studied in the context of probabilistic timed automata in [20]. We consider the case when 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.

The probabilistic timed automaton representing the medium is given in Fig. 1. 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 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.

The probabilistic timed automaton representing a station i ($i = 1,2$) is given in Fig. 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 delay before retransmitting is an integer number of time slots (each of length $slot$). The number of slots that station i waits after the nth transmission failure is chosen as a uniformly distributed *random* integer in the range: $0, 1, 2, \ldots, 2^{bc_i+1} - 1$, 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 Fig. 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.

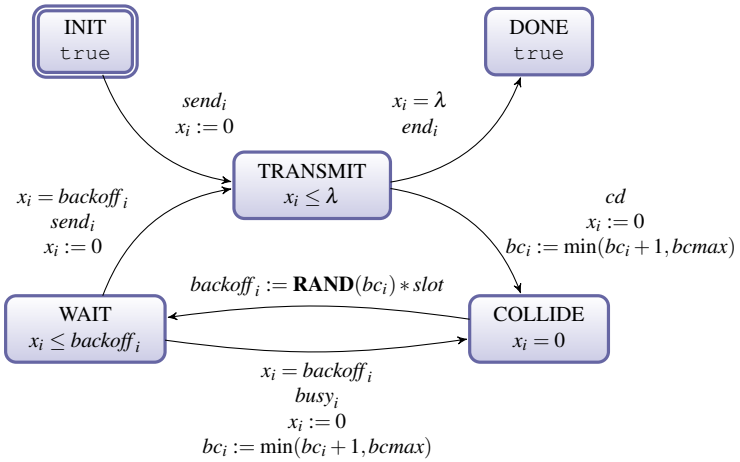

Figure 2: CSMA/CD Station i

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: $\delta = 26\mu s$, $\lambda = 808\mu s$, and $slot = 2\delta = 52\mu s$. The method for inferring a constraint K_0 on the parameters, which is satisfied by the reference valuation and in which the behavior of the model remains the same, consists in transforming the system into a non-probabilistic parametric timed automaton. We replace the random choice $backoff_i := \mathbf{RAND}(bc_i) * slot$ with a *non-deterministic* choice, i.e., a set of 2^{bc_i+1} transitions associated with assignments of the form $backoff_i := j * slot$, for $j = 0, 1, 2, \ldots, 2^{bc_i+1} - 1$. In the case where $bcmax = 1$, the application of the inverse method to the non-probabilistic parametric timed automaton (see Section 4.1) infers for K_0 the following constraint: $(0 < \delta < slot) \wedge (15slot < \lambda < 16slot)$. In particular, the minimum and maximum probabilities for a message sent by a station to be transmitted (i.e., to reach the location DONE) after having collided exactly k times with another message (action *cd*) are the same under the reference valuation and another parameter valuation satisfying K_0. This has two practical implications. Firstly, in order to compute the aforementioned minimum and maximum probabilities for $\delta = 26, \lambda = 808, slot = 52$, it suffices to compute the minimum and maximum probabilities for $\delta = 1, \lambda = 31, slot = 2$ (because both valuations satisfy the constraint $(0 < \delta < slot) \wedge (15slot < \lambda < 16slot)$ generated by the inverse method). Note that the valuation $\delta = 26, \lambda = 808, slot = 52$ results in a model with 5240 states using the discrete-time semantics construction, whereas the valuation $\delta = 1, \lambda = 31, slot = 2$ results in a model with 282 states. The second practical implication concerns the case in which the system designer wishes to understand the behavior 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.

Plan of the paper. In Section 2, we present the definition of parametric probabilistic timed automata. In Section 3, 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 4, we apply the method to three probabilistic protocols with timing parameters (CSMA/CD, IEEE 1394 root contention, IEEE 802.11 WLAN). We conclude in Section 5.

2 Parametric Probabilistic Timed Automata

In Section 2.1, we review the definition of *timed probabilistic systems*, as defined in [17], which is a variant of Segala's probabilistic timed automata [22]. In Section 2.2, we extend the definition of probabilistic timed automata [17] to the parametric case, and give its semantics in terms of timed probabilistic systems (Section 2.3).

2.1 Timed Probabilistic Systems

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 $\mu(z) = 1$.

A *timed probabilistic system (TPS)* is a tuple $\mathsf{T} = (S, S_0, Act, \Rightarrow)$ where: S is a set of *states*, including a set S_0 of *initial states*; Act is a finite set of *actions* (disjoint from $\mathbb{R}_{\geq 0}$); $\Rightarrow \subseteq S \times \mathbb{R}_{\geq 0} \times Act \times \mathsf{Dist}(S)$ is a *probabilistic transition relation*.

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 $last(\omega) = s_n$. The set of paths of a TPS T is denoted by $Path_{fin}^{\mathsf{T}}$. When clear from the context we omit the superscript T and write $Path_{fin}$. We let $Path_{fin}(s)$ denote the set of paths commencing in the state $s \in S$.

A *scheduler* is a partial function which chooses an outgoing transition in the last state of a path, if a transition from the last state of a path exists. Formally, a scheduler of a TPS is a partial function σ such that, for each path ω of the TPS, we have (1) if $\sigma(\omega) = (d,a,\mu)$ then $(last(\omega),d,a,\mu) \in \Rightarrow$, and (2) if $\sigma(\omega)$ is undefined then there does not exist any (d,a,μ) such that $(last(\omega),d,a,\mu) \in \Rightarrow$. A scheduler resolves the nondeterminism by choosing a transition based on the path executed so far. Intuitively, if a TPS is guided by scheduler σ and has the path ω as its history, then it will be in state s in the next step with probability $\mu(s)$, where $\sigma(\omega) = (d,a,\mu)$; alternatively, if there is no available transition from ω, then the system deadlocks, and the choice of the scheduler will be undefined. We denote the set of paths induced by a given scheduler σ to be $Path_{fin}^{\sigma} = \{\omega = s_0 \xrightarrow{d_0,a_0,\mu_0} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} s_n \mid \sigma(\omega{\downarrow_i}) = (d_i,a_i,\mu_i)$ *for all* $i <$

$n\}$, where $\omega\downarrow_i$ returns the prefix of ω up to length i. Then we define $Path^{\sigma}_{fin}(s) = Path^{\sigma}_{fin} \cap Path_{fin}(s)$. A scheduler σ a TPS is said to be *admissible* if, for all $s \in S_0$ and $\omega \in Path^{\sigma}_{fin}(s)$, there exists some transition $(last(\omega), d, a, \mu) \in \Rightarrow$. For each $s \in S$ and scheduler σ, we can define the probability measure $Prob^{\sigma}_{s}$ over measurable sets of paths in the standard way [15].

2.2 Syntax of Parametric Probabilistic Timed Automata

We now extend the definition of probabilistic timed automata [17] to the parametric case. 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 $X = \{x_1, \ldots, x_H\}$. 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 $(w(x_1), \ldots, w(x_H)) \in \mathbb{R}^H_{\geq 0}$. 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.

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 *linear inequality on the parameters P* (*linear inequality on the clocks X and the parameters P*, respectively) is an inequality $e \prec e'$, where $\prec \in \{<, \leq\}$, and e, e' are two linear terms of the form:

$$\sum_i \alpha_i p_i + d, \quad \left(\sum_i \alpha_i p_i + \sum_j \beta_j x_j + d, \text{respectively}\right)$$

where $1 \leq i \leq M, 1 \leq j \leq H$ and $\alpha_i, \beta_j, d \in \mathbb{N}$. A *constraint on the parameters P* (*constraint on the clocks X and the parameters P*, respectively) is a conjunction of inequalities on P (on X and 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 $w(x)$. A clock valuation w *satisfies* $C[\pi]$, denoted by $w \models C[\pi]$, if $C[\pi][w]$ evaluates to true. 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)* $\mathscr{A}$ is a tuple of the form $\mathscr{A} = (\Sigma, Q, \overline{q}, X, P, I, prob)$, 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 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 $\eta \in \mathsf{Dist}(2^X \times Q)$.

We make the following assumptions on PPTAs.

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 $\eta(\rho, q') > 0$.

Neither of these assumptions is restrictive, because a PPTA not satisfying the assumptions can be transformed into a PPTA which does: for determinism on actions, it is necessary to add and rename actions, whereas, for reset unicity, it suffices to add an extra clock and additional locations. The assumptions of determinism on actions and reset unicity are commonly met in practice, and they simplify the proofs of our subsequent results.

Let $\mathscr{A}$ be a PPTA. If, for each location $q \in Q$, we have that $I(q)$ is a constraint only on clocks, and, for each edge $(q, g, a, \eta) \in prob$, we have that g is a constraint only on clocks, we say that $\mathscr{A}$ is a *probabilistic timed automaton* (PTA).

Remark. We make use in Figure 2 of several forms of syntactic sugar consisting in merging several transitions issued from the same location, using assignments of the form $bc_i := \min(bc_i + 1, bcmax)$, or $backoff_i := \mathbf{RAND}(bc_i) * slot$, following the conventions used, e.g., in [16].

2.3 Semantics of Parametric Probabilistic Timed Automata

In this section, we will consider the PPTA $\mathscr{A} = (\Sigma, Q, \overline{q}, X, P, I, prob)$. Given a parameter valuation $\pi = (\pi(p_1), \ldots, \pi(p_M))$, we denote by $\mathscr{A}[\pi]$ the PTA obtained from $\mathscr{A}$ by substituting every occurrence of a parameter p_i by $\pi(p_i)$ in the guards and invariants. Formally, $\mathscr{A}[\pi] = (\Sigma, Q, \overline{q}, X, P, I', prob')$, where I' and $prob'$ are defined in the following way: for each location $q \in Q$, we let $I'(q) = I(q)[\pi]$, and we let $prob'$ be the smallest set such that, for each $(q, g, a, \eta) \in prob$, we have $(q, g[\pi], a, \eta) \in prob'$.[1]

In the following, we consider the PTA $\mathscr{A}[\pi]$ resulting from a given valuation π of the parameters. A *state* of $\mathscr{A}[\pi]$ is a pair $(q, w) \in Q \times \mathbb{R}^H_{\geq 0}$ such that $w \models I(q)[\pi]$. Informally, the behavior of $\mathscr{A}[\pi]$ 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 either (1) making a *discrete (probabilistic) transition* or (2) letting *time pass*. In case (1), a discrete transition can

[1] Strictly speaking, $\mathscr{A}[\pi]$ is a PTA only when π assigns a natural number (rather than a real) to each parameter, but this does not matter in our context.

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 $\eta(\rho,q')$. In case (2), the option of letting time pass is available only if the invariant $I(q)$ is satisfied while time elapses.

Formally, we define the semantics of a PTA as an associated infinite-state, infinite-branching TPS, defined as follows. The *TPS* (or *semantics*) *associated with* $\mathscr{A}[\pi]$ is $\mathsf{T}_{\mathscr{A}[\pi]} = (S, S_0, \Sigma, \Rightarrow)$ with $S = \{(q,w) \in Q \times (X \to \mathbb{R}_{\geq 0}) \mid w \models I(q)[\pi]\}$, $S_0 = \{(\overline{q}, \mathbf{0})\}$ 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: $w + d \models I(q)[\pi]$;

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

Observe that the rule for discrete transitions is a simplified version of the standard rule [17], and relies on the assumption of reset unicity. The definition of $\mathsf{T}_{\mathscr{A}[\pi]}$ also relies on the fact that $\mathscr{A}$ and π satisfy the following well-formedness assumption. First, a PTA is said to be *well-formed* if whenever a probabilistic edge is enabled it can be taken, i.e.: all of the probabilistic alternatives (pairs of target location and clock reset) result in states. Formally, a PTA is said to be well-formed if, for each probabilistic edge $(q,g,a,\eta) \in \mathit{prob}$ and state $(q,w) \in S$ such that $w \models g[\pi]$, we require that $(q', \rho(w)) \in S$ for each $(\rho,q') \in \mathsf{support}(\eta)$. [2] Then we say that a PPTA is *well-formed* if, for each parameter valuation, the resulting PTA is well-formed. A PPTA can be transformed into a well-formed PPTA by incorporating the invariant associated to the target location into the guard of each probabilistic edge (along the lines of the transformation in [20]). For the remainder of the paper we assume that all of the PPTAs we consider are well-formed.

Note that a (P)PTA $\mathscr{A}$ for which all probabilistic edges feature point distributions can be interpreted as a (parametric) timed automaton. More precisely, the (parametric) timed automaton differs from $\mathscr{A}$ only in the edge relation: we represent a probabilistic edge $(q,g,a,\eta) \in \mathit{prob}$ of $\mathscr{A}$, for which $\eta(\rho,q') = 1$ for some $\rho \subseteq X$ and $q' \in Q$ (recall that $\mathscr{A}$ features point distributions only) by a single edge in the (parametric) timed automaton.

Networks of PPTAs can be defined by using parallel composition based on the synchronization of discrete transitions of different components sharing the same action in a similar manner to networks of PTAs [19].

Given a path $\omega = (q_0,w_0) \xrightarrow{d_0,a_0,\mu_0} (q_1,w_1) \xrightarrow{d_1,a_1,\mu_1} \cdots \xrightarrow{d_{n-1},a_{n-1},\mu_{n-1}} (q_n,w_n)$, we let the *time-abstract trace* of ω be the sequence of alternating locations and actions $q_0a_0q_1a_1\cdots a_{n-1}q_n$. Given a scheduler σ, we let $\mathsf{trace}^\sigma : \mathit{Path}^\sigma_{\mathit{fin}} \to (Q \times \Sigma)^*$ be the function associating the time-abstract trace with each path of $\mathit{Path}^\sigma_{\mathit{fin}}$. 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^σ 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

[2] A counter-example of well-formed PTA is the following: let (q,w) be a state where $w(x) = 2$, let $(q, x \leq 2, a, \eta)$ be a probabilistic edge such that $\eta(q',\emptyset) = \frac{1}{2}$ and $\eta(q'',\emptyset) = \frac{1}{2}$, and let $\mathit{inv}(q') = (x \leq 1)$ and $\mathit{inv}(q'') = (x \leq 2)$. Then the invariant of q' is not satisfied when taking the probabilistic edge $(q, x \leq 2, a, \eta)$, followed by the probabilistic selection of $(q',\emptyset)$, from (q,w).

assigned by td_s^σ to traces in which a certain location is reached is defined to be the same as the probability assigned by $Prob_s^\sigma$ to the set of paths in which this location is reached. The set of time-abstract trace distributions of the TPS $\mathsf{T}_{\mathscr{A}[\pi]}$ is denoted by $\mathsf{tdist}(\mathsf{T}_{\mathscr{A}[\pi]}) = \{\mathsf{td}_s^\sigma \mid \sigma \text{ is a scheduler of } \mathsf{T}_{\mathscr{A}[\pi]} \text{ and } s \in S_0\}$.

We say that $\mathscr{A}[\pi]$ and $\mathscr{A}[\pi']$ are *time-abstract trace distribution equivalent*, written $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi']$, if $\mathsf{tdist}(\mathsf{T}_{\mathscr{A}[\pi]}) = \mathsf{tdist}(\mathsf{T}_{\mathscr{A}[\pi']})$. If $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi']$, we can conclude that the TPSs have time-abstract equivalent finite behaviors: 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). Finally, we write $Path_{fin}^{\mathscr{A}[\pi]}$ for $Path_{fin}^{\mathsf{T}_{\mathscr{A}[\pi]}}$.

3 Analysis of PPTAs Using the Inverse Method

In this section we consider an application of the inverse method to PPTAs. Our approach consists in applying the inverse method to a *non-probabilistic version* of the PPTA. The constraint output by the inverse method is also a solution to the inverse problem for the PPTA and the reference instantiation. Ideally, we would like to generate a constraint which is as *weak* as possible (i.e., satisfied by as many valuations as possible).

We first present formally the problem we intend to resolve, then introduce a method for obtaining (non-probabilistic) parametric timed automata from PPTAs. Finally we explain how the results on the inverse method applied to parametric timed automata can be used to infer constraints on parameters of PPTAs.

3.1 The Inverse Problem on PPTAs

Consider the PPTA $\mathscr{A} = (\Sigma, Q, \overline{q}, X, P, I, prob)$, which we assume is fixed throughout this section. Let π be a valuation of parameters in P, and let $((q,w),d,a,\mu) \in \Rightarrow$ be a transition of $\mathsf{T}_{\mathscr{A}[\pi]}$. Recall that, by reset unicity and the definition of $\mathsf{T}_{\mathscr{A}[\pi]}$, for each distinct $(q,w),(q',w') \in \mathsf{support}(\mu)$, we have $q \neq q'$. We define the distribution $\mathsf{loc}(\mu) \in \mathsf{Dist}(Q)$ over locations in the following way: for each $(q,w) \in S$, we let $\mathsf{loc}(\mu)(q) = \mu(q,w)$.

Let π' be a valuation of parameters in P. 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 $\mathsf{T}_{\mathscr{A}[\pi]}$, 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 $\mathsf{T}_{\mathscr{A}[\pi]}$, written $\omega \equiv^{\mathsf{path}} \omega'$, if $q_i = q'_i$, $a_i = a'_i$, and $\mathsf{loc}(\mu_i) = \mathsf{loc}(\mu'_i)$ for all $i = 0,\ldots,n$. We extend the notion of time-abstract path equivalence to sets of paths: two sets Ω and Ω' of paths are time-abstract path equivalent, written $\Omega \equiv^{\mathsf{path}} \Omega'$, if (1) for each path $\omega \in \Omega$, there exists $\omega' \in \Omega'$ such that $\omega \equiv^{\mathsf{path}} \omega'$, and (2) conversely, for each path $\omega \in \Omega'$, there exists $\omega' \in \Omega$ such that $\omega \equiv^{\mathsf{path}} \omega'$.

We recall below a result from [18, 19] which allows to relate time-abstract equivalence on paths to time-abstract trace distribution equivalence.

Proposition 1 *Let $\mathscr{A}$ be a PPTA, and let π and π' be instantiations of parameters P. If $Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}[\pi']}(\overline{q},\mathbf{0})$, then $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi']$.*

In this paper, starting from an instantiation π_0 of the set P of parameters, we are interested in finding a constraint K_0 on the parameters such that $\pi_0 \models K_0$, and for any valuation π of P satisfying K_0 we have time-abstract trace distribution equivalence between $\mathscr{A}[\pi_0]$ and $\mathscr{A}[\pi]$. Furthermore, we suppose that $\mathscr{A}[\pi_0]$ has an admissible scheduler. The problem can be stated as follows.

> Consider a PPTA $\mathscr{A}$ and a valuation π_0 of the parameters such that $\mathscr{A}[\pi_0]$ has an admissible scheduler. Find a constraint K_0 such that:
>
> 1. $\pi_0 \models K_0$,
> 2. $\mathscr{A}[\pi]$ has an admissible scheduler, for all $\pi \models K_0$, and
> 3. $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi_0]$, for all $\pi \models K_0$.

3.2 Non-probabilistic Version of a PPTA

In this subsection, we state formal properties relating a PPTA to its non-probabilistic version. As explained in Section 2.3, it is straightforward to obtain a non-probabilistic parametric timed automaton from a PPTA featuring point distributions only. Given a PPTA $\mathscr{A}$, an *edge generated from* $(q,g,a,\eta) \in prob$ is a tuple (q,g,a,η,ρ,q') such that $\eta(\rho,q') > 0$. Let $\mathtt{edges}(q,g,a,\eta)$ be the set of the edges generated from (q,g,a,η), and let $\mathtt{edges} = \bigcup_{(q,g,a,\eta)\in prob} \mathtt{edges}(q,g,a,\eta)$ denote the set of all edges of $\mathscr{A}$.

The *non-probabilistic version* of $\mathscr{A}$, written $\mathscr{A}^*$, is a PPTA which agrees with $\mathscr{A}$ on all elements apart from the probabilistic edge relation. Formally, let $\mathscr{A}^* = (\Sigma, Q, \overline{q}, X, P, I, prob^*)$ be the PPTA for which $prob^*$ is the smallest probabilistic edge relation such that for every edge $(q,g,a,\eta,\rho,q') \in \mathtt{edges}$, we have $(q,g,\langle\langle q,g,a,\eta,\rho,q'\rangle\rangle,\eta_{(\rho,q')}) \in prob^*$ (recall that $\eta_{(\rho,q')}$ denotes the point distribution assigning probability 1 to the element (ρ,q')). Observe that the state sets of $\mathsf{T}_{\mathscr{A}[\pi]}$ and $\mathsf{T}_{\mathscr{A}^*[\pi]}$ are equal. As noted in Section 2, from a PPTA for which all probabilistic edges feature point distributions, we can obtain the corresponding parametric timed automaton.

In the following proposition, we use $\rightarrow$ to refer to a transition of the semantic TPS of $\mathscr{A}[\pi]$, and $\rightarrow_*$ to refer to a transition of the semantic TPS of $\mathscr{A}^*[\pi]$.

Proposition 2 *Let π be an instantiation of P and (q,w) be a state of $\mathsf{T}_{\mathscr{A}[\pi]}$ (and $\mathsf{T}_{\mathscr{A}^*[\pi]}$). For each step $(q,w) \xrightarrow{d,a,\mu} (q',w')$ of $\mathsf{T}_{\mathscr{A}[\pi]}$, there exists the step $(q,w) \xrightarrow{d,\langle\langle q,g,a,\eta,\rho,q'\rangle\rangle,\mu_{(q',w')}}_* (q',w')$ of $\mathsf{T}_{\mathscr{A}^*[\pi]}$, where $(q,g,a,\eta) \in prob$ is such that $\mu(q',w') = \eta(\rho,q')$, and hence $\mathsf{loc}(\mu)(q') = \eta(\rho,q')$. Conversely, for each step $(q,w) \xrightarrow{d,\langle\langle q,g,a,\eta,\rho,q'\rangle\rangle,\mu_{(q',w')}}_* (q',w')$ of $\mathsf{T}_{\mathscr{A}^*[\pi]}$, there exists the step $(q,w) \xrightarrow{d,a,\mu} (q',w')$ of $\mathsf{T}_{\mathscr{A}[\pi]}$ such that $\mu(q',w') = \eta(\rho,q')$, and hence $\mathsf{loc}(\mu)(q') = \eta(\rho,q')$.*

Proposition 2 allows us to obtain a one-to-one mapping between transitions of $\mathscr{A}[\pi]$ and $\mathscr{A}^*[\pi]$. By reasoning inductively, we can extend the proposition to obtain a one-to-one mapping between paths of $\mathscr{A}[\pi]$ and $\mathscr{A}^*[\pi]$. Note that the probability of the transitions of $\mathscr{A}[\pi]$ is encoded in the actions of the associated transitions of $\mathscr{A}^*[\pi]$. This, together with the one-to-one mapping between paths of $\mathscr{A}[\pi]$ and $\mathscr{A}^*[\pi]$, implies that, for any pair ω_*, ω'_* of paths such that

$\omega_* \in Path_{fin}^{\mathscr{A}^*[\pi]}(\overline{q},\mathbf{0})$, $\omega'_* \in Path_{fin}^{\mathscr{A}^*[\pi']}(\overline{q},\mathbf{0})$ and $\omega_* \equiv^{\mathsf{path}} \omega'_*$, we can generate the paths ω,ω', such that $\omega \in Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0})$, $\omega' \in Path_{fin}^{\mathscr{A}[\pi']}(\overline{q},\mathbf{0})$ and $\omega \equiv^{\mathsf{path}} \omega'$. Together, these facts allow us to show the following.

Proposition 3 *Let $\mathscr{A}$ be a PPTA, and let π and π' be instantiations of parameters P. If $Path_{fin}^{\mathscr{A}^*[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}^*[\pi']}(\overline{q},\mathbf{0})$, then $Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}[\pi']}(\overline{q},\mathbf{0})$.*

Proposition 4 *Let $\mathscr{A}$ be a PPTA, and let π and π' instantiations of parameters P. If $Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}[\pi']}(\overline{q},\mathbf{0})$, then $\mathscr{A}[\pi]$ has an admissible scheduler iff $\mathscr{A}[\pi']$ has an admissible scheduler.*

3.3 Resolution of the Inverse Problem for PPTAs

In [3], we have presented a method which solves the inverse problem for (the special case of) non-probabilistic parametric timed automata. The inverse problem can be described formally in the following way: given a non-probabilistic parametric timed automaton $\mathscr{A}$ and a valuation π_0 of the parameters, find a constraint K_0 such that $\pi_0 \models K_0$ and $Path_{fin}^{\mathscr{A}[\pi_0]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0})$ for all $\pi \models K_0$. A brief explanation of the method is given in the appendix.

The following algorithm explains how to use the inverse method in the extended framework of PPTAs.

ALGORITHM *InverseMethodPPTA*$(\mathscr{A},\pi_0)$
Input $\mathscr{A}$: PPTA
π_0 : Valuation of the parameters
Output K_0 : Constraint on the parameters

1. Construct the non-probabilistic version $\mathscr{A}^*$ of $\mathscr{A}$.
2. Construct K_0 for $\mathscr{A}^*$ using the classical inverse method.

Theorem 1 *Given a PPTA $\mathscr{A}$ and a reference valuation π_0 such that $\mathscr{A}[\pi_0]$ has an admissible scheduler, the constraint K_0 returned by InverseMethodPPTA$(\mathscr{A},\pi_0)$ solves the inverse problem, i.e., $\pi_0 \models K_0$, and, for all $\pi \models K_0$, $\mathscr{A}[\pi]$ has an admissible scheduler, and $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi_0]$.*

Proof. Since K_0 is a solution of the inverse problem for $\mathscr{A}^*$, we have $Path_{fin}^{\mathscr{A}^*[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}^*[\pi_0]}(\overline{q},\mathbf{0})$ for all $\pi \models K_0$; hence, we have by Proposition 3 that $Path_{fin}^{\mathscr{A}[\pi]}(\overline{q},\mathbf{0}) \equiv^{\mathsf{path}} Path_{fin}^{\mathscr{A}[\pi_0]}(\overline{q},\mathbf{0})$ for all $\pi \models K_0$. From Proposition 1, we conclude that $\mathscr{A}[\pi] \approx^{\mathsf{tdist}} \mathscr{A}[\pi_0]$ for all $\pi \models K_0$. Furthermore, since $\mathscr{A}[\pi_0]$ has an admissible scheduler, $\mathscr{A}[\pi]$ has an admissible scheduler, for all $\pi \models K_0$ (by Proposition 4). □

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

Remark. The constraint K_0 output by our method is *not* (in general) the weakest constraint satisfying the inverse problem as defined in Section 3.1. One reason for this is that the constraint output by the inverse method defined in [3] for (non-probabilistic) parametric timed automata is always in conjunctive form. In contrast, the weakest constraint may be in disjunctive form (see final remarks of [4]).

4 Application of the Inverse Method to PPTAs: Case Studies

In this section, we show the interest of the inverse method in the context of three case studies. More precisely, we consider three protocols, each modeled as a PPTA, and each with an associated standard reference valuation π_0.[3]

Our approach consists of the following two phases:

1. Using the tool IMITATOR [5], which implements the inverse method in the non-probabilistic framework, we generate 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 reachability probabilities for various 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.

4.1 CSMA/CD Protocol

We first apply our method to the CSMA/CD Protocol described in Section 1. We consider the three parameters λ, δ and *slot* as described in [20, 23]. The following instantiation π_0 of the parameters is the reference valuation taken from the IEEE standard 802.3 for 10 Mbps Ethernet: $\lambda = 808\mu s$, $slot = 52\mu s$ and $\delta = 26\mu s$. As sketched in Section 1, the non-probabilistic parametric timed automaton $\mathscr{A}^*$ is obtained as follows: we compose the (non-probabilistic) medium of Fig. 1 with the non-probabilistic version of the station, obtained by replacing in Fig. 2 the random choice $backoff_i := \mathbf{RAND}(bc_i) * slot$ with a non-deterministic choice (i.e., a set of 2^{bc_i+1} transitions associated with assignments of the form $backoff_i := j * slot$, for $j = 0, 1, 2, \ldots, 2^{bc_i+1} - 1$).

Applying IMITATOR to this non-probabilistic model and the reference valuation π_0, we obtain the following constraint:

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

This constraint is such that $\mathscr{A}[\pi]$ and $\mathscr{A}[\pi_0]$ are time-abstract trace-distribution equivalent, for any $\pi \models K_0$.

We consider the following four properties, described also with their associated PRISM syntax:

- $Prop_j$, for $j \in \{0, 1, 2\}$: minimum probability that station 1 transmits its message after exactly j collisions, i.e., $\texttt{Pmin} = ?[\texttt{F}(\texttt{s}_1 = \texttt{done} \ \& \ \texttt{nbCol} = j)]$.

[3] Note that we consider acyclic versions of those protocols (roughly speaking, by bounding the maximal number of collisions in Section 4.1 and Section 4.3, and by bounding the number of rounds in Section 4.2).

Name	λ	*slot*	δ	$\models K_0$	$Prop_0$	$Prop_1$	$Prop_2$	$Prop_{\leq 3}$	Same as π_0
π_0	808	52	26	yes	0	0.5	0.375	0.96875	-
π_1	404	26	13	yes	0	0.5	0.375	0.96875	yes
π_2	31	2	1	yes	0	0.5	0.375	0.96875	yes
π_3	47	3	2	yes	0	0.5	0.375	0.96875	yes
π_4	940	60	59	yes	0	0.5	0.375	0.96875	yes
π_5	940	60	60	no	0	0	0.1875	0.609375	no
π_6	832	52	26	no	0	0.5	0.375	0.96875	yes
π_7	52	52	26	no	0	0.5	0.375	0.96875	yes

Table 1: Minimum probability that station 1 transmits its message

- $Prop_{\leq 3}$: minimum probability that station 1 transmits its message with no more than 3 collisions, i.e., $\texttt{Pmin}=?[\texttt{F}(\texttt{s}_1 = \texttt{done} \,\&\, \texttt{nbCol} \leq 3)]$.

We apply PRISM to the system with the parameters set to different valuations (including π_0). The results are given in Table 1. For all $\pi \models K_0$ (i.e., π_0 to π_4), the probabilities remain the same. An observation is that, as soon as we violate the constraint K_0 by considering the limit case where $\delta = slot$ (i.e., π_5), the probabilities are different. A further observation is that, even if the value of λ violates K_0 (i.e., π_6 and π_7), the probabilities can remain the same. Indeed, the constraint generated by our method is not necessarily the weakest, i.e., we can find valuations π of P (e.g., π_6 and π_7) s.t. $\pi \not\models K_0$ but the values of the probabilities remain the same as for π_0.

4.2 IEEE 1394 Root Contention Protocol

This case study concerns the root contention protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus, also considered in the parametric framework in [13]. We consider the rescaled instantiation of the parameters given in [19, 23]. This instantiation π_0 is as follows[4]: $rc\_fast\_max = 85$, $rc\_fast\_min = 76$, $rc\_slow\_max = 167$, $rc\_slow\_min = 159$, and $delay = 30$. Applying IMITATOR to a (non-probabilistic) parametric timed automaton version of this model and the reference valuation π_0, we obtain the following constraint:

$$K_0 : 2delay < rc\_fast\_min \wedge rc\_fast\_max + 2delay < rc\_slow\_min.$$

Note that this constraint is exactly the same as the one synthesized in [13]. We consider the following properties:

- $Prop_{\leq 3}$: minimum probability that a leader is elected after 3 rounds or less: $\texttt{Pmin}=?[\texttt{F}((\texttt{nbRounds1} \leq 3) \,\&\, (((\texttt{s1}=8) \,\&\, (\texttt{s2}=7)) | ((\texttt{s1}=7) \,\&\, (\texttt{s2}=8)))))]$.
- $Prop_{\leq 5}$: minimum probability that a leader is elected after 5 rounds or less: $\texttt{Pmin}=?[\texttt{F}((\texttt{nbRounds1} \leq 5) \,\&\, (((\texttt{s1}=8) \,\&\, (\texttt{s2}=7)) | ((\texttt{s1}=7) \,\&\, (\texttt{s2}=8)))))]$.

The results of the application of PRISM to different parameter valuations are given in Table 2 (some parameter names are abbreviated for reasons of space). We notice as expected that, for

[4] Note that the model given on PRISM's webpage allows both values 30 and 360 for *delay*. Moreover, the IEEE reference instantiation is given in *ns* but, due to the rescaling, we omit the unit here.

Name	rf_max	rf_min	rs_max	rs_min	$delay$	$\models K_0$	$Prop_{\leq 3}$	$Prop_{\leq 5}$	Same as π_0
π_0	85	76	167	159	30	yes	0.875	0.96875	-
π_1	40	35	80	75	15	yes	0.875	0.96875	yes
π_2	4	3	8	7	1	yes	0.875	0.96875	yes
π_3	85	61	167	159	30	yes	0.875	0.96875	yes
π_4	85	76	167	146	30	yes	0.875	0.96875	yes
π_5	85	76	167	159	36	yes	0.875	0.96875	yes
π_6	85	76	167	159	37	no	0.71875	0.841796875	no

Table 2: Minimum probability that a leader is elected within 3 or 5 rounds

all $\pi \models K_0$, the probabilities remain the same. Further experiments (including π_6) show that, as soon as the value of the parameters violates K_0, the probabilities become different from π_0. We note that the parameter valuation π_0 results in a state space of size 693107, for which $Prop_{\leq 5}$ could be verified in time 319.512s, whereas π_2 results in a state space of size 1393, for which $Prop_{\leq 5}$ could be verified in time 0.781s.[5]

4.3 IEEE 802.11 Wireless Local Area Network Protocol

We also applied our method to the IEEE 802.11 Wireless Local Area Network Protocol, considering the following instantiation π_0 of the parameters mentioned in [23] after rescaling from [18][6]:

$$ASLOTTIME = 1 \quad DIFS = 2 \quad VULN = 1 \quad TTMAX = 315$$
$$TTMIN = 4 \quad ACK\_TO = 6 \quad ACK = 4 \quad SIFS = 1$$

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

$$\begin{array}{lllll} & VULN > 0 & \wedge & SIFS > 0 & \wedge \quad ACK\_TO + DIFS < 15 ASLOTTIME \\ \wedge & DIFS > 0 & \wedge & ASLOTTIME > 0 & \wedge \quad TTMIN + DIFS \leq TTMAX \\ \wedge & ACK \leq 2DIFS & \wedge & DIFS < TTMIN & \wedge \quad ACK\_TO + DIFS \leq ACK + TTMIN \\ \wedge & SIFS < TTMIN & \wedge & TTMIN \geq ACK & \wedge \quad TTMIN \leq ACK\_TO \\ \wedge & VULN < ACK & & & \end{array}$$

We consider the maximum probability that either station's backoff counter reaches k, for $k = 1,2,3$, as considered in [18]. The results of the application of PRISM are given in Table 3, where the properties are denoted by $Prop_{k=i}$ for $k = 1,2,3$. The parameters $p_1, p_2, \ldots, p_8$ stand for $ASLOTTIME$, $DIFS$, $VULN$, $TTMAX$, $TTMIN$, ACK_TO, ACK, $SIFS$ respectively. Note that the real timings originating from the IEEE 802.11 standard (viz., in μs, $ASLOTTIME = 50$, $DIFS = 128$, $VULN = 48$, $TTMAX = 15,717$, $TTMIN = 224$, $ACK\_TO = 300$, $ACK = 205$, $SIFS = 28$) satisfy themselves the constraint K_0. Our approach thus provides us with a justification of the abstraction done in [18] consisting in reducing the time scale of the model. Also observe that, as expected from the form of K_0, the value of $Prob_{k=i}$ is insensitive to important variations of $TTMAX$, i.e. parameter p_4 (provided its value remains greater or equal to $TTMIN + TTDIFS$, i.e $p_4 \geq p_2 + p_5$). For example, $Prob_{k=i}$ is equal for π_0 and π_2, in spite of the fact that p_4 is changed from 315 to 6.

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

[6] Note that, due to rescaling, the model given on PRISM's webpage allows several values for the parameters, e.g., 2 and 3 for $DIFS$.

Name	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	$\models K_0$	$Prop_{k=1}$	$Prop_{k=2}$	$Prop_{k=3}$	Same as π_0
π_0	1	2	1	315	4	6	4	1	yes	1	0.183593	0.017032	-
π_1	1	2	1	150	4	6	4	1	yes	1	0.183593	0.017032	yes
π_2	1	2	1	6	4	6	4	1	yes	1	0.183593	0.017032	yes
π_3	1	2	1	315	10	12	4	1	yes	1	0.183593	0.017032	yes
π_4	1	2	1	12	10	12	4	1	yes	1	0.183593	0.017032	yes
π_5	2	4	2	630	8	12	8	2	yes	1	0.183593	0.017032	yes
π_6	2	4	2	315	8	10	7	2	yes	1	0.183593	0.017032	yes

Table 3: Maximum probability of either station's backoff counter reaching k

5 Final Remarks

In this paper we have shown that the inverse method presented in [3] 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 [3]. The method has been shown to be successful in obtaining smaller parameter values, which can be helpful in reducing the size of the integer-time semantic PTA models prior to 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.

Since the constraint output by our method is not the weakest in general (see the remark in Section 3.3), it is interesting to design methods for weakening it further. In particular, the incremental method sketched in [3] could also be used in the probabilistic framework.

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 note that soft deadline properties can be reduced to time-abstract location reachability properties, but with the addition of constraints within the model: hence, the inverse problem must be solved on the modified model. We plan to explore this approach in future work.

Acknowledgement. Étienne André and Laurent Fribourg are 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 MIUR-PRIN project PaCo - Performability-Aware Computing: Logics, Models and Languages.

Bibliography

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

[2] R. Alur, T. Henzinger, and M. Vardi. Parametric real-time reasoning. In *Proc. STOC '93*, pages 592–601. ACM, 1993.

[3] É. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, 2009.

[4] É. André, E. Encrenaz, and L. Fribourg. Synthesizing parametric constraints on various case studies using IMITATOR. Research Report LSV-09-13, Laboratoire Spécification et Vérification, ENS Cachan, France, 2009.

[5] Étienne André. IMITATOR: A tool for synthesizing constraints on timing bounds of timed automata. In *Proc. ICTAC'09*, volume 5684 of *LNCS*, pages 336–342. Springer, 2009.

[6] Dirk Beyer. Improvements in BDD-based reachability analysis of timed automata. In *Proc. FME'01*, volume 2021 of *LNCS*, pages 313–343. Springer, 2001.

[7] M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In *Proc. CHARME'99*, volume 1703 of *LNCS*. Springer, 1999.

[8] N. Chamseddine, M. Duflot, L. Fribourg, C. Picaronny, and J. Sproston. Computing expected absorption times for parametric determinate probabilistic timed automata. In *Proc. QEST'08*, pages 254–263. IEEE, 2008.

[9] Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *Proc. ICTAC'04*, volume 3407 of *LNCS*, pages 280–294. Springer, 2004.

[10] T. Han, J.-P. Katoen, and A. Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *Proc. RTSS'08*, pages 173–182. IEEE, 2008.

[11] T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. ICALP'92*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.

[12] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

[13] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 2002.

[14] Henrik Ejersbo Jensen. Model checking probabilistic real time systems. In *Proc. of the 7th Nordic Work. on Progr. Theory*. Chalmers Institute of Technology, 1996.

[15] J. G. Kemeny, J. L. Snell, and A. W Knapp. *Denumerable Markov Chains*. Graduate Texts in Mathematics. Springer, 2nd edition, 1976.

[16] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *FMSD*, 29:33–78, 2006.

[17] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *TCS*, 282:101–150, 2002.

[18] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

[19] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14(3):295–318, 2003.

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

[21] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Weak bisimulation for probabilistic timed automata and applications to security. In *Proc. SEFM'03*, pages 34–43. IEEE, 2003.

[22] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[23] PRISM web page. `http://www.prismmodelchecker.org/`.

ALGORITHM *InverseMethod*($\mathscr{A}$, π_0)

Inputs $\mathscr{A}$: Parametric timed automaton of initial location $\overline{q}$
π_0 : Reference valuation of the parameters

Output K_0 : Constraint on the parameters

Variables i : Current iteration
S : Current set of symbolic states ($S = \bigcup_{j=0}^{i} Post^j_{\mathscr{A}(K)}(\{(\overline{q},K)\})$)
K : Current constraint on the parameters

$i := 0$; $K := True$; $S := \{(\overline{q}, True)\}$

DO

DO UNTIL there are no π_0-incompatible states in S

Select a π_0-incompatible state (q,C) of S (i.e., s.t. $\pi_0 \not\models C$)

Select a π_0-incompatible J in C (i.e., s.t. $\pi_0 \not\models J$)

$K := K \wedge \neg J$; $S := \bigcup_{j=0}^{i} Post^j_{\mathscr{A}(K)}(\{(\overline{q},K)\})$

OD *%% S is π_0-compatible*

IF $Post_{\mathscr{A}(K)}(S) = \emptyset$ **THEN RETURN** $K_0 := \bigcap_{(q,C)\in S}(\exists X : C)$ **FI**

$i := i+1$; $S := S \cup Post_{\mathscr{A}(K)}(S)$ *%% $S = \bigcup_{j=0}^{i} Post^j_{\mathscr{A}(K)}(\{(\overline{q},K)\})$*

OD

Figure 3: Algorithm *InverseMethod*

Appendix: The Inverse Method

Given a (classical) parametric timed automaton $\mathscr{A}$ and a reference instantiation π_0 of parameters, the inverse method outputs a constraint K_0 such that:

1. $\pi_0 \models K_0$,
2. $Path^{\mathscr{A}[\pi_0]}_{fin} \equiv^{\mathsf{path}} Path^{\mathscr{A}[\pi]}_{fin}$, for all $\pi \models K_0$.

The algorithm *InverseMethod* can be summarized as follows. Starting with $K := True$, 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 $\mathscr{A}$, and C a constraint on the parameters[7]. When a π_0-*incompatible* state (q,C) is encountered (i.e., when $\pi_0 \not\models C$), K is refined as follows: a π_0-incompatible inequality J (i.e., such that $\pi_0 \not\models J$) 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 (we focus here on *acyclic* systems: see [3] for details).

A simplified version of algorithm *InverseMethod* is given in Fig. 3, where the clocks have been disregarded for the sake of simplicity. We denote by $Post^i_{\mathscr{A}}(S)$ the set of symbolic states reachable from S in at most i steps of $\mathscr{A}$.

There is an implementation of this algorithm, called IMITATOR, which is written in Python, and makes use of HYTECH for the computation of the *Post* operation. The Python program contains about 1500 lines of code, and its writing took about 4 man-months of work (see [5]).

[7] Strictly speaking, C is a constraint on the parameters *and* the clocks, but the clocks are omitted here for the sake of simplicity. See [3] for more details.