Computing Expected Absorption Times for Parametric Determinate Probabilistic Timed Automata

N. Chamseddine[1], M. Duflot[2], L. Fribourg[1], C. Picaronny[1] and J. Sproston[3]
[1] LSV - ENS Cachan & CNRS, France
[2] LACL - University Paris 12, France
[3] Dipartimento di Informatica, Università di Torino, Italy

Abstract

We consider a variant of probabilistic timed automata called parametric determinate probabilistic timed automata*. Such automata are fully probabilistic: there is a single distribution of outgoing transitions from each of the automaton's nodes, and it is possible to remain at a node only for a given amount of time. The residence time within a node may be given in terms of a parameter, and hence we do not assume that its concrete value is known. We claim that, often in practice, the maximal expected time to reach a given absorbing node of a probabilistic timed automaton can be captured using a parametric determinate probabilistic timed automaton. We give a method for computing the expected time for a parametric determinate probabilistic timed automaton to reach an absorbing node. The method consists in constructing a variant of a Markov chain with costs (where the costs correspond to durations), and is parametric in the sense that the expected absorption time is computed as a function of the model's parameters. The complexity of the analysis is independent from the maximal constant bounding the values of the clocks, and is polynomial in the number of edges of the original parametric determinate probabilistic timed automaton.*

1 Introduction

Quantitative analysis of models of probabilistic systems typically involves the computation of performance measures such as the probability of reaching some state, the expected cost incurred or reward obtained before reaching some state, or values indicating the long-run average behavior of the system. We consider the problem of computing the expected time to reach a certain state in timed probabilistic systems. Such a measure is useful in cases in which we are interested in the average time-efficiency of a system before reaching some specific goal state.

As a modelling framework, we consider probabilistic timed automata [20, 23], a probabilistic extension of timed automata [2], which has been shown to be useful for the modelling of timed randomized protocols [24, 22, 16, 17]. Probabilistic timed automata permit the modelling of probabilistic systems, representing the system's timing constraints through the use of clock variables, which can be tested and reset on the automaton's edges. Our aim is to provide an efficient, parameterized technique for computing the expected time until absorption (which corresponds to reaching the target state) in a restricted class of probabilistic timed automata, namely Parametric Determinate Probabilistic Timed Automata (PDPTA). As in the case of parametric timed automata [3], a PDPTA can use parameters to refer to timing requirements. The aim is then to obtain the expected absorption time of a PDPTA in terms of an expression over the parameters of the model; apart from providing an insight into the manner in which the parameters can affect the expected absorption time, this allows us to obtain the set of parameter valuations of the model which attain an expected time above, below or equal to some value. We limit PDPTA to purely probabilistic choice, rather than nondeterministic and probabilistic choice in general probabilistic timed automata. More precisely, in PDPTA, (1) there is a single distribution of outgoing transitions per PDPTA node, and (2) the amount of time that can elapse after entering some node is unique. Points (1) and (2) mean that there can be no nondeterminism between discrete transitions, and over time delays, respectively. We observe that PDPTA can be regarded alternatively as Markov chains with cost or reward functions specified in a compact, high-level way through the use of clocks. The formal definition of PDPTA is given in Section 2.

We claim that, for several significant protocol models of the literature previously studied with probabilistic timed automata, for example, the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol [25, 1], and the IEEE1394 FireWire root contention protocol [24], the behavior of the system under the worst scheduler (the

scheduler which resolves nondeterminism in such a way as to obtain an expected absorption time at least that of any other scheduler) can be captured by a PDPTA. This observation arises principally from two observations: first, models for these protocols often feature a restricted amount of nondeterminism between outgoing distributions from nodes, with the majority of such nodes having a single outgoing distribution, and, second, the timing constraints of these models are sufficiently simple to allow the restriction of our attention on longest possible delays, thereby removing the nondeterminism over exact timing delays which is present in general probabilistic timed automata.

The advantage of focusing on PDPTAs is that the absorption time can be computed efficiently. Our analysis method, presented in Section 3, takes the form of a syntactic analysis of the timing constraints and clock resets of the PDPTA, followed by the construction of a graph similar to a Markov chain with costs which contains the required information with regard to probabilities and time durations. We then show that this graph can be used to obtain the expected absorption time efficiently through the use of linear equation solution techniques. The overall solution method is of polynomial time complexity, in contrast to the EXPTIME-complete problems on general probabilistic timed automata [23, 28]. We also note that the only previous approach to the computation of expected absorption time of probabilistic timed automata relies on a discretization of the state space [22], resulting in a Markov decision process with transitions of cost 0 (corresponding to the traversal of an edge of the automaton) and cost 1 (corresponding to the elapse of one time unit). This method is sensitive to the magnitude of the maximal constants used in timing constraints of the model. In contrast to the discretization method, our method avoids the explicit representation of clock values.

The cost of such an efficient method is paid by the restrictions (1) and (2) above imposed on general probabilistic timed automata in order to obtain determinate probabilistic timed automata (and hence PDPTA). In fact we do not propose that PDPTA offer a general solution for the computation of absorption times of probabilistic timed automata models. However, we note that the model of IEEE1394 FireWire root contention protocol which we will discuss in Section 2 is an abstract model, obtained from a detailed by a combination of manual proof techniques in [31] and non-probabilistic, tool-aided analysis in [30]: from this abstract model, it then possible to obtain a PDPTA model which represents the worst-case behavior of the system. We envisage that manual techniques and automatic analysis can be used to obtain tractable PDPTA models from general probabilistic timed automata in the case of other protocols.

Related work. As noted above, expected-time properties of general probabilistic timed automata can be verified using a discretization approach [22]. The problem of verifying properties of parametric timed automata has been considered in [3]. Although the verification problem is in general undecidable, subclasses of parametric timed automata have been identified which have decision algorithms [3, 19, 7, 9]. Non-timed systems in which parameters can be used to represent probabilities have been presented in [15, 26].

2 Parameterized Determinate Probabilistic Timed Automata

In this section, we define the parametric, fully probabilistic variant of probabilistic timed automata [23] which will be the focus of the paper. This formalism allows us to reason about the timing delays of the model in a parametric manner; that is, the timing delays can be described in terms of unknown constants. In the following, we use both real-valued variables called *clocks*, which increase at the same rate as real-time, and natural-numbered variables called *parameters* to define timing constraints. A *probability distribution* over a finite set Q is a function $\mu : Q \to [0,1]$ such that $\sum_{q \in Q} \mu(q) = 1$. Let $\mathsf{Dist}(Q)$ be the set of probability distributions over Q. We assume that probability distributions take rational values only (that is are of the form $\mu : Q \to [0,1] \cap \mathbb{Q}$, where $\mathbb{Q}$ denotes the set of rationals).

Definition 1 *A* Parametric Determinate Probabilistic Timed Automaton (PDPTA) *is a tuple* $\mathcal{A} = (L, \bar{l}, \mathcal{X}, \mathcal{P}, \phi, \psi, \mathbf{prob})$ *which comprises: (1) a finite set L of* locations*, including the* initial location $\bar{l} \in L$*; (2) a finite set $\mathcal{X}$ of clocks; (3) a finite set $\mathcal{P}$ of parameters; (4) a* clock assignment function $\phi : L \to \mathcal{X}$ *assigning to each location a clock; (5) a* delay assignment function $\psi : L \to (\mathcal{P} \cup \mathbb{N})$ *assigning to each location a parameter or a natural number; (6) a* probabilistic transition function $\mathbf{prob} : L \to \mathsf{Dist}(2^{\mathcal{X}} \times L)$.

Before defining formally the semantics of PDPTA, we give a brief description of its behavior. The behavior of a PDPTA is defined with respect to a function $\kappa : \mathcal{P} \to \mathbb{N}$, called a *parameter valuation*, where $\mathbb{N}$ denotes the set of natural numbers. Given ψ, we define $\psi_\kappa : L \to \mathbb{N}$ as the function such that $\psi_\kappa(l) = \psi(l)$ for all $l \in L$ such that $\psi(l) \in \mathbb{N}$, and $\psi_\kappa(l) = \kappa(\psi(l))$ for all $l \in L$ such that $\psi(l) \in \mathcal{P}$. A PDPTA starts in the initial location $\bar{l}$ with the values of all clocks in $\mathcal{X}$ equal to 0. The PDPTA remains in $\bar{l}$ as time elapses, and hence the values of all clocks increase by the same amount. Let $x = \phi(\bar{l})$ and $d = \psi_\kappa(\bar{l})$ (hence $d = \psi(\bar{l})$ if $\psi(\bar{l}) \in \mathbb{N}$, and $d = \kappa(\psi(\bar{l}))$ otherwise). Time continues to elapse until the value of the clock x equals d; when the value d is reached, a transition is made from $\bar{l}$ according to the distribution $\mathbf{prob}(\bar{l})$. More precisely, with probability $\mathbf{prob}(\bar{l})(X, l)$, the PDPTA makes a transition to location l, resetting the values of the clocks in X to 0.

The process then continues from l. In this case, the PDPTA makes a transition after a time delay required for the current value of the clock $\phi(l)$ to increase to equal the value $\psi_\kappa(l)$.

Readers familiar with the standard probabilistic timed automaton model [23] will note that PDPTA plus a parameter valuation form a subclass of probabilistic timed automata: for every location, the amount of time that can elapse on entry to the location of a PDPTA is determined, rather than being a (generally) nondeterministic choice as in probabilistic timed automata. Furthermore, in PDPTA, for a given location l, there is only one probability distribution, namely $\mathbf{prob}(l)$, that can be used to choose the next location and the set of clocks to reset to 0.

We now proceed to the definition of the semantics of PDPTA. Let $\nu : \mathcal{X} \to \mathbb{N}$ be a function assigning a natural number to each of the clocks in $\mathcal{X}$. Such a function is called a *clock valuation*. We denote the set of all clock valuations of $\mathcal{X}$ by $\mathbb{N}^{\mathcal{X}}$. Let $\mathbf{0}$ be the clock valuation that assigns 0 to all clocks in $\mathcal{X}$. For some $X \subseteq \mathcal{X}$, we write $\nu[X := 0]$ for the clock valuation that assigns 0 to clocks in X, and agrees with ν for all clocks in $\mathcal{X} \setminus X$. For every $t \in \mathbb{N}$, $\nu + t$ denotes the clock valuation defined by $\nu(x) + t$ for all clocks $x \in \mathcal{X}$. A *state* of a PDPTA is a pair (l, ν), where $l \in L$ is a location and $\nu \in \mathbb{N}^{\mathcal{X}}$ is a clock valuation. The amount of time that can elapse from a state (l, ν) before an outgoing edge must be taken is the amount of time necessary to increase the clock $\phi(l)$ from its value in ν in order to reach the value $\psi_\kappa(l)$, and is denoted by $\mathsf{Exit}_\kappa(l, \nu) = \psi_\kappa(l) - \nu(\phi(l))$. Given a state (l, ν) such that $\nu(\phi(l)) \leq \psi_\kappa(l)$, and a state (l', ν'), we let $\mathsf{Reset}((l, \nu), \nu') = \{X \subseteq \mathcal{X} \mid \nu' = (\nu + \mathsf{Exit}_\kappa(l, \nu))[X := 0]\}$. An *edge* of the PDPTA $\mathcal{A}$ is a tuple $e = (l, X, l')$ where $l, l' \in L$ are locations and $X \subseteq \mathcal{X}$ is a set of clocks such that $\mathbf{prob}(l)(X, l') > 0$. We denote by $\mathcal{E}_\mathcal{A}$ the set of all edges of $\mathcal{A}$.

In the following, we use the notation $(l, \nu) \xrightarrow{e}_\kappa (l', \nu')$ to denote a transition from state (l, ν) to (l', ν'), by first letting $\mathsf{Exit}_\kappa(l, \nu)$ time units elapse, then traversing edge e. Formally, given states $(l, \nu), (l', \nu') \in L \times \mathbb{N}^{\mathcal{X}}$, edge $(l, X, l') \in \mathcal{E}_\mathcal{A}$, we write $(l, \nu) \xrightarrow{(l,X,l')}_\kappa (l', \nu')$ if (1) $\mathbf{prob}(l)(X, l') > 0$, (2) $\mathsf{Exit}_\kappa(l, \nu) \geq 0$, and (3) $\nu' = (\nu + \mathsf{Exit}_\kappa(l, \nu))[X := 0]$.

Definition 2 *A* (finite) path of $\mathcal{A}$ with κ *is a sequence of consecutive transitions:*

$$\omega = (l_0, \nu_0) \xrightarrow{e_1}_\kappa (l_1, \nu_1) \xrightarrow{e_2}_\kappa \cdots \xrightarrow{e_m}_\kappa (l_m, \nu_m) \ .$$

The length of ω*, denoted by* $|\omega|$*, is* m*, and the last location of* ω*, denoted by* $last(\omega)$*, is* l_m*. The* probability of ω*, denoted by* $\Pr(\omega)$*, is defined by* $\Pr(\omega) = \prod_{k=0}^{m-1} \mathbf{prob}(l_k)(X_{k+1}, l_{k+1})$*. The* duration of ω*, denoted by* $\mathrm{Dur}_\kappa(\omega)$*, is defined by* $\mathrm{Dur}_\kappa(\omega) = \sum_{k=0}^{m-1} \mathsf{Exit}_\kappa(l_k, \nu_k)$.

A path ω *is a* prefix *of a path* $\omega' = (l_0, \nu_0) \xrightarrow{e_1}_\kappa \cdots \xrightarrow{e_m}_\kappa (l_m, \nu_m)$ *if* $\omega = (l_0, \nu_0) \xrightarrow{e_1}_\kappa \cdots \xrightarrow{e_n}_\kappa (l_n, \nu_n)$ *for some* $n < m$*. The set of paths of* $\mathcal{A}$ *with* κ *such that the first state is* $(\bar{l}, \mathbf{0})$ *is denoted by* $Paths(\mathcal{A}, \kappa)$.

In the standard way [21], for a set Ω of paths with common first state (l, ν) such that no path in Ω is a prefix of another path in Ω, we let $\Pr(\Omega) = \sum_{\omega \in \Omega} \Pr(\omega)$.

We also note that, for paths starting from a particular state (l, ν), the series of edges and locations visited along the path determines the time durations and clock valuations corresponding to the path. Hence we often suppress the clock valuations from all states of the path apart from the first, writing $(l_0, \nu_0) \xrightarrow{e_1}_\kappa l_1 \xrightarrow{e_2}_\kappa \cdots \xrightarrow{e_m}_\kappa l_m$.

It can be observed that we have chosen a *discrete-time* semantics for PDPTA, rather than the continuous-time semantics which is standard for probabilistic timed automata [23]. However, a PDPTA with a parameter valuation corresponds to a (determinate) *closed* probabilistic timed automaton, for which the continuous-time and discrete-time semantics are equivalent for a number of performance measures, including expected absorption time [22]. Therefore it suffices to consider a discrete-time semantics.

We restrict our attention to PDPTA and parameter valuations for which it is not possible to reach, from the initial state $(\bar{l}, \mathbf{0})$, a state (l, ν) in which $\mathsf{Exit}_\kappa(l, \nu) < 0$.

Definition 3 *A PDPTA* $\mathcal{A}$ *with a parameter valuation* κ *is* well formed *if, for each path* $(l_0, \nu_0) \xrightarrow{e_1}_\kappa \cdots \xrightarrow{e_m}_\kappa (l_m, \nu_m)$ *of* $\mathcal{A}$ *such that* $(l_0, \nu_0) = (\bar{l}, \mathbf{0})$*, we have* $\mathsf{Exit}_\kappa(l_m, \nu_m) \geq 0$.

Whether a PDPTA is well formed can be checked by a executing the reachability algorithm of [3] on a parametric timed automaton corresponding to the PDPTA, obtained in the following way: (1) probabilistic choice between edges is replaced by a nondeterministic choice; (2) every edge (l, X, l') of the PDPTA is represented by two edges of the parametric timed automaton, one of which has the guard $\phi(l) = \psi(l) \wedge \phi(l') \leq \psi(l')$ and which goes to the location l', the other having the guard $\phi(l) = \psi(l) \wedge \phi(l') > \psi(l')$ and which goes to a new location l_{sink}. Then l_{sink} is not reachable in the parametric timed automaton if and only if the PDPTA is well formed. Reachability is undecidable for parametric timed automata, and therefore the algorithm is not guaranteed to terminate. This approach can be used also to synthesize parameter valuations which guarantee well formedness; in this case, we can restrict subsequent analysis of the expected absorption time only to those parameter valuations. We note that a condition of well-formedness is also required of general probabilistic timed automata, but can be tested syntactically, and enforced by imposing timing constraints referring to possibly multiple clocks [25], which is

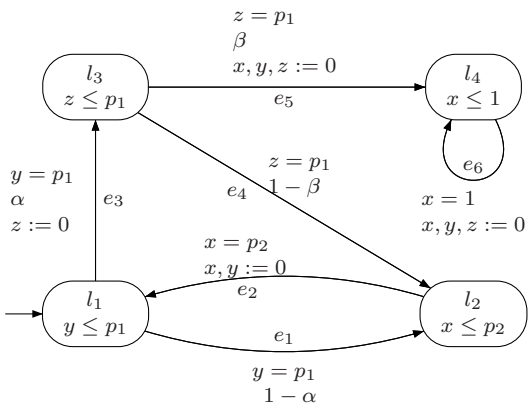

Figure 1. PDPTA of Example 1

not possible in the PDPTA framework given here. Furthermore, parametric analysis of a non-probabilistic timed automaton obtained from a PDPTA can be used to synthesize bounds on the parameter valuations which guarantee certain required qualitative behavior, such as rechability of certain locations. We can then restrict the computation of expected absorption times, as presented in Section 3, to the parameter valuations satisfying the synthesized bounds.

We also consider the class of *structurally non-Zeno* well-formed PDPTAs and parameter valuations (originally adapted from [32] and considered for probabilistic systems in [27]), in which all loops of the graph of the PDPTA obtained from the set of locations L and set of edges $\mathcal{E}_{\mathcal{A}}$ involve the passage of at least 1 time unit.

Definition 4 *A PDPTA $\mathcal{A}$ with a parameter valuation κ is* structurally non-Zeno *if, for all sequences of edges $(l_0, X_1, l_1)(l_1, X_2, l_2) \cdots (l_{m-1}, X_m, l_m)$ of $\mathcal{E}_{\mathcal{A}}$ such that $l_0 = l_m$, there exists a clock $x \in \mathcal{X}$ and $1 \leq i, j \leq m$ such that $x \in X_i$, $\phi(l_j) = x$, and and $\psi_\kappa(l_j) \geq 1$.*

Example 1 *Let us consider the PDPTA depicted in Figure 1. This PDPTA is inspired by a simplified version of the sender in the Bounded Retransmission Protocol model of [13, 12]. There are four locations l_1, l_2, l_3 and l_4, and three clocks x, y and z. We use standard conventions for the graphical representation of timed automata. The initial location is l_1. The clock and delay assignment functions are represented by a combination of an invariant condition labelling locations and a guard labelling outgoing edges (for example, $\phi(l_1) = y$ and $\psi(l_1) = p_1$ is represented by the invariant $y \leq p_1$ and guard $y = p_1$). Probabilities of edges are represented by values denoted by α, $1 - \alpha$, β and $1 - \beta$, with labels for probability 1 omitted for simplicity. To guarantee well formedness, we require that $p_2 \geq p_1$.*

We note that PDPTA correspond to finite-state Markov chains, which are a widely-used for probabilistic analysis and performance evaluation, with a cost or reward function defined (in a high-level and parametric manner) by clocks and the associated clock assignment and delay assignment functions. Markov chains are more tractable than Markov decision processes [29], and similarly our results also show that efficient algorithms for PDPTA can be developed, in contrast to the EXPTIME-hardness results for general probabilistic timed automata [28]. In order to model some systems, a fully probabilistic framework can be applied directly; for others, as in the case of Markov chains, some effort or trade-offs may be required to obtain fully probabilistic models from high-level system descriptions. We mention some factors which may be taken into consideration in order to obtain PDPTA (more precisely, fully probabilistic timed automata models, from which PDPTA can be obtained). Firstly, in certain cases, timing constraints may be sufficiently simple to allow us to fix a scheduler of nondeterminism which always chooses, for example, the longest time to be elapsed in a location in order to generate the maximal expected absorption time. In the case of nondeterminism between probability distributions over edges from a location, it may similarly be possible to identify manually the scheduler which results in the maximal expected time. Secondly, the imposition of constraints on parameters which guarantee a notion of correct qualitative behavior (such as the correct behavior of the system) can reduce the amount of nondeterministic choice of the model. Thirdly, particularly in the context of the parallel composition of components, nondeterminism can be reduced by partial-order reduction [4, 14]; this may allow us to obtain a simplified, intermediate model, containing nondeterminism, from which it is possible to apply manual techniques to remove the remaining nondeterminism. Finally, we point out that nondeterminism is often used to reason about partially unknown timing constraints in probabilistic timed automata; in PDPTA, parameters (possibly with valuations in some defined set) can be used to represent unknown timing constraints. For example, we may know that a delay occurs between 4 and 6 time units after entry to a location, which can be represented by a parameter and considering valuations 4, 5 and 6.

Example 2 *We now give a brief example of a PDPTA in order to explain why clocks and their associated functions can be considered as a high-level mechanism for specifying costs and rewards; this example will also illustrate the differences between the PDPTA approach and the discrete-time approach to probabilistic timed automata [22]. Consider the PDPTA in Figure 2, which has two clocks x and y, and k "triangular" segments, in which the ith such segment has either 0 or 2^i duration. On entry to location l, the amount of time elapsed since the start of the system is any number between 0 and $2^k - 1$. Therefore, in l, it is possible that there is a time delay of any length from 1 to 2^k. In*

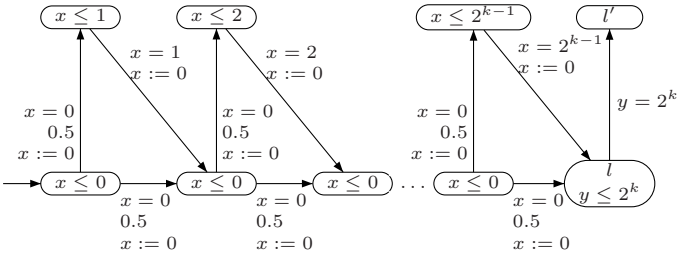

Figure 2. PDPTA of Example 2

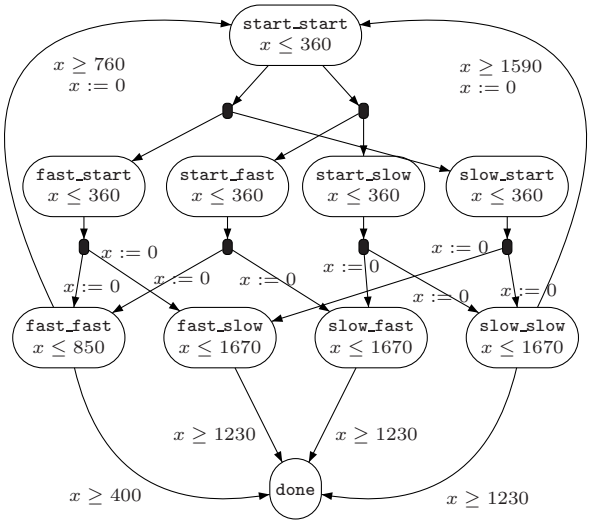

Figure 3. Probabilistic timed automaton of the root contention protocol

a Markov chain with a function associating a (single) cost or reward to each state [18], we must distinguish all such cases by a separate state: therefore, to represent the location l, we require a set of states which is exponential in k. Furthermore, in the discrete-time approach to probabilistic timed automata, a state corresponds to a location and a valuation of the clocks, and hence the number of states generated by this approach is also exponential in k, because the clock x may take values between 0 and 2^k. In Section 3, we present a method for the computation of the expected absorption time that avoids such an exponential cost.

Example 3 *We also illustrate an approach to identify manually a scheduler of nondeterminism on the IEEE1394 (FireWire) root contention protocol. We assume familiarity with the general probabilistic timed automata formalism [23]. We consider the abstract probabilistic timed automaton $\mathtt{I}_1^p$ [30, 24], as shown in Figure 3.*

In this probabilistic timed automaton, there are two

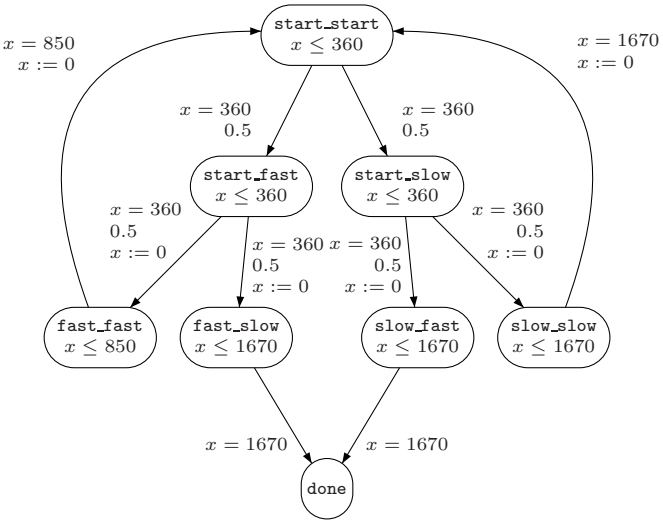

Figure 4. PDPTA for a scheduler of the root contention protocol

sources of nondeterminism: on the one hand, the choice of the distribution over edges to be taken from locations `start_start`, `fast_fast` *and* `slow_slow`*; on the other hand, the amount of time to stay at each location (as given by the invariant condition labelling the location, and by the guard of the outgoing distributions).*

In this simple example, it is possible to identify a scheduler (depending only on the current location) of nondeterministic choice which captures the behavior resulting in the maximal expected time to reach the location `done`*. With regard to the choice of distributions, the scheduler should select the edge which goes from* `fast_fast` *(respectively,* `slow_slow`*) to* `start_start`*, and either choice of distribution over edges from* `start_start` *can be taken. With regard to the amount of time elapsed in each location, the scheduler should stay as much as possible as allowed by the invariant. The corresponding behavior corresponds to the PDPTA illustrated in Figure 4 (for simplicity, we use constants rather than parameters).*

As noted above, we are interested in computing the expected time to absorption in a PDPTA, given a parameter valuation, or in obtaining an expression over parameters which gives the expected time to absorption. For convenience, we assume that there is a special location l_{end} in L which is terminal, or "absorbing", for $\mathcal{A}$ which has the following properties: (1) the outgoing probability distribution from l_{end} loops in l_{end} with probability 1, (2) the probability that a path (starting at 0) of length m contains l_{end} tends to 1 when m tends to ∞, and (3) all incoming edges of l_{end} reset all clocks to 0.

Definition 5 *A PDPTA $\mathcal{A}$ with parameter valuation κ is* absorbing *if there exists a location $l_{end} \in L$ such that: (1) $\mathbf{prob}(l_{end})(\mathcal{X}, l_{end}) = 1$; (2) $\lim_{m\to\infty} \Pr(\{\omega \in Paths(\mathcal{A},\kappa) \mid last(\omega) = l_{end} \wedge |\omega| = m\}) = 1$; (3) for each $(l, X, l_{end}) \in \mathcal{E}_{\mathcal{A}}$, we have $X = \mathcal{X}$.*

The PDPTA of Example 1 is absorbing, by taking $l_{end} = l_4$. Whether a PDPTA is absorbing can be checked by verifying that there is only a single outgoing edge of l_{end}, by a syntactic analysis of the incoming and outgoing edges of l_{end}, and by verifying that l_{end} can be reached from $\bar{l}$ with probability 1 in the Markov chain obtained for the PDPTA by removing all timing information (clocks and delay assignment functions). Henceforth, we assume that PDPTA are well-formed, structurally non-Zeno, and absorbing.

Our results also apply to the case in which we compute the expected time to reach a certain set of locations of the PDPTA: all edges to this set of locations are redirected to the location l_{end} (maintaining the same probabilities). The following definition of expected absorption time is standard [8].

Definition 6 *Let $\Omega(l_{end}) \subseteq Paths(\mathcal{A},\kappa)$ be the set of paths of $\mathcal{A}$ with κ such that, for all paths $\omega \in \Omega(l_{end})$, we have $last(\omega) = l_{end}$, and there exists no prefix ω' of ω such that $last(\omega') = l_{end}$.*

The expected absorption time of the PDPTA $\mathcal{A}$ with the parameter valuation κ, *denoted by* $\mathsf{ExpAbs}(\mathcal{A},\kappa)$, *is defined by* $\mathsf{ExpAbs}(\mathcal{A},\kappa) = \sum_{\omega\in\Omega(l_{end})} \Pr(\omega)\mathrm{Dur}_{\kappa}(\omega)$.

3 Computation of the Absorption Time

Given a PDPTA $\mathcal{A}$, we now explain how to compute an expression over the set $\mathcal{P}$ of $\mathcal{A}$ within which we can substitute parameter valuations κ to obtain $\mathsf{ExpAbs}(\mathcal{A},\kappa)$. The idea is to abstract a sequence of consecutive edges of $\mathcal{A}$ with fixed duration a, where a is a natural number or a parameter, as a unique transition called a "macro-step" with cost a. By this method, we transform a PDPTA into a graph where each transition is supplied with a cost and a weight (where the weight is derived from probabilities of the PDPTA). This graph can then be used to compute the required expression over $\mathcal{P}$. Proofs of the results in the following sections can be found in [11].

3.1 The Graph of Macro-Steps

Consider a finite sequence of transitions of a PDPTA such that a clock x is reset immediately before the sequence, is not reset in any transition of the sequence, and is tested in the source location l of the final transition of the sequence (that is, we have $x = \phi(l)$). Then the total duration of traversing the sequence of transitions must be equal to $\psi(l)$ time units. In the following, a *point of* $\mathcal{A}$ is a pair $(X, l) \in 2^{\mathcal{X}} \times L$ such that there exists some $(\_, X, l) \in \mathcal{E}_{\mathcal{A}}$. We also let $(\mathcal{X}, \bar{l})$ be a point. We consider macro-steps of a PDPTA not from location to location, but from point to point.

Definition 7 (Macro-step via σ.) *Let (X, l) be a point of $\mathcal{A}$. Let $\sigma = (l_0, Y_1, l_1)(l_1, Y_2, l_2)\cdots(l_{m-1}, Y_m, l_m)$ be a sequence of edges of $\mathcal{A}$ such that $l_0 = l$, the clock $\phi(l_{m-1})$ is such that $\phi(l_{m-1}) \in X$ and $\phi(l_{m-1}) \notin Y_i$ for all $1 \leq i < m$. We say that there is a* macro-step from point (X, l_0) to point (Y_m, l_m) via σ, *denoted by* $(X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m)$.

The weight $\mathsf{Wgt}((X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m))$ of the macro-step $(X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m)$ *is defined as* $\prod_{k=0}^{m-1} \mathbf{prob}(l_k)(Y_{k+1}, l_{k+1})$.

The duration $\mathsf{Dur}((X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m))$ of the macro-step $(X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m)$ *is defined as* $\psi(l_m)$.

The length of the macro-step $(X, l_0) \stackrel{\sigma}{\Rightarrow} (Y_m, l_m)$ *is defined to be* m.

Example 4 *Consider the PDPTA of Example 1. There is a sequence σ_1 of edges going from l_1 to l_1 of the form e_1e_2. The clock x is not reset to 0 by the edge e_1. It follows that there is a macro-step via σ_1 of the form $(\mathcal{X}, l_1) \stackrel{\sigma_1}{\Rightarrow} (\{x, y\}, l_1)$. We have $\mathsf{Wgt}((\mathcal{X}, l_1) \stackrel{\sigma_1}{\Rightarrow} (\{x, y\}, l_1)) = 1-\alpha$ and $\mathsf{Dur}((\mathcal{X}, l_1) \stackrel{\sigma_1}{\Rightarrow} (\{x, y\}, l_1)) = p_2$.*

Likewise, there is a sequence σ_2 of edges going from l_1 to l_1 of the form $e_3e_4e_2$. The clock x is not reset to 0 either by the edge e_3 or the edge e_4. Hence there is a macro-step via σ_2 of the form $(\mathcal{X}, l_1) \stackrel{\sigma_2}{\Rightarrow} (\{x, y\}, l_1)$, with $\mathsf{Wgt}((\mathcal{X}, l_1) \stackrel{\sigma_2}{\Rightarrow} (\{x, y\}, l_1)) = \alpha(1 - \beta)$ and $\mathsf{Dur}((\mathcal{X}, l_1) \stackrel{\sigma_2}{\Rightarrow} (\{x, y\}, l_1)) = p_2$.

A macro-step from (X, l) to (Y, l') via σ can be seen as inducing a set of paths Ω, each of which corresponds to the traversal of edges of σ after clocks in X have been reset to 0. Each path in Ω will correspond to a different valuation of the clocks in $\mathcal{X}\setminus X$ (that is, the point (X, l) specifies that the values of clocks in X must start at 0 at the beginning of the path, but does not specify the values of other clocks). In particular, the associated probabilities and durations are consistent: a macro-step via σ, as defined in Definition 7, corresponding to a path $\omega \in \Omega$ satisfies $\mathsf{Wgt}((X, l) \stackrel{\sigma}{\Rightarrow} (Y, l')) = \Pr(\omega)$, and $\kappa(\mathsf{Dur}((X, l) \stackrel{\sigma}{\Rightarrow} (Y, l'))) = \mathrm{Dur}_{\kappa}(\omega)$, where, for a linear expression E over parameters $\mathcal{P}$ with rational coefficients, we let $\kappa(E)$ be the value obtained from E by substituting $\kappa(p)$ in the place of p in E, for all $p \in \mathcal{P}$.

Proposition 1 *Consider a macro-step via σ. The length of such a macro-step is bounded from above by $|L|$.*

The proposition follows from the fact that σ cannot contain a cycle. Assume that a cycle is contained in σ: then,

by well-formedness, a PDPTA path could repeat the cycle an arbitrary number of times. By structural non-Zenoness (which specifies that an cycle must take at least 1 time unit) the cycle could be repeated so that the duration of the path exceeds any bound. This implies that the bound on the clock $\phi(l)$ imposed by $\psi_\kappa(l)$, where l is the source location of the last edge in σ, can be exceeded by repeating the cycle at least $\psi_\kappa(l)+1$ times, which contradicts well-formedness.

Given two points $(X,l),(Y,l')$ and edge e of the form $(\_,Y,l')$ (that is, edge e resets the set of clocks Y to 0 and has target location l'), we define the set of all sequences σ of consecutive edges ending in e forming a macro-step from (X,l) to (Y,l'), written as $\mathsf{EndSet}((X,l),e,(Y,l'))$, as the set $\{\sigma \in (\mathcal{E}_\mathcal{A})^* \cdot e \mid (X,l) \stackrel{\sigma}{\Rightarrow} (Y,l')\}$. Proposition 1 ensures that every $\sigma \in \mathsf{EndSet}((X,l),e,(Y,l'))$ is of bounded length, and hence the set $\mathsf{EndSet}((X,l),e,(Y,l'))$ is finite.

Definition 8 (Macro-step according to an edge e.) *Let $(X,l),(Y,l')$ be two points and let e be an edge of the form $(\_,Y,l')$. We say that there is a* macro-step from (X,l) to (Y,l') according to e, *denoted by $(X,l) \stackrel{e}{\Rightarrow} (Y,l')$, if $\mathsf{EndSet}((X,l),e,(Y,l')) \neq \emptyset$.*

The weight of a macro-step from (X,l) to (Y,l') according to e, *denoted by* $\mathsf{Wgt}((X,l) \stackrel{e}{\Rightarrow} (Y,l'))$, *is given by:*

$$\sum_{\sigma \in \mathsf{EndSet}((X,l),e,(Y,l'))} \mathsf{Wgt}((X,l) \stackrel{\sigma}{\Rightarrow} (Y,l')) \ .$$

The duration associated to such a macro-step, *denoted by* $\mathsf{Dur}((X,l) \stackrel{e}{\Rightarrow} (Y,l'))$, *is equal to the common value* $\mathsf{Dur}((X,l) \stackrel{\sigma}{\Rightarrow} (Y,l'))$, *for all* $\sigma \in \mathsf{EndSet}((X,l),e,(Y,l'))$.

Definition 9 (Macro-step.) *Let $(X,l),(Y,l')$ be two points. Let $\mathcal{E}_{(Y,l')}$ be the set of all edges in $\mathcal{E}_\mathcal{A}$ of the form $(\_,Y,l')$. We say that there is a* macro-step from (X,l) to (Y,l'), *denoted by $(X,l) \Rightarrow (Y,l')$, if there exists $e \in \mathcal{E}_{(Y,l')}$ such that $\mathsf{EndSet}((X,l),e,(Y,l'))$ is not empty.*

The weight of a macro-step from (X,l) to (Y,l'), *denoted by* $\mathsf{Wgt}((X,l) \Rightarrow (Y,l'))$, *is given by:*

$$\sum_{e \in \mathcal{E}_{(Y,l')}} \mathsf{Wgt}((X,l) \stackrel{e}{\Rightarrow} (Y,l')) \ .$$

The duration of a macro-step from (X,l) to (Y,l'), *denoted by* $\mathsf{Dur}((X,l) \Rightarrow (Y,l'))$, *is defined as:*

$$\sum_{e \in \mathcal{E}_{(Y,l')}} \frac{\mathsf{Wgt}((X,l) \stackrel{e}{\Rightarrow} (Y,l')) \cdot \mathsf{Dur}((X,l) \stackrel{e}{\Rightarrow} (Y,l'))}{\mathsf{Wgt}((X,l) \Rightarrow (Y,l'))} \ .$$

The graph of macro-steps is the graph whose nodes are points and transitions are macro-steps. Without loss of generality, we assume that there is only one node of the form $(\_,\bar{l})$ (resp. $(\_,l_{end})$), namely $(\mathcal{X},\bar{l})$ (resp. $(\mathcal{X},l_{end})$), which we denote by $\bar{l}$ (resp. l_{end}).

Example 5 *Consider again the PDPTA depicted in Figure 1, and the macro-steps $(\mathcal{X},l_1) \stackrel{\sigma_1}{\Rightarrow} (\{x,y\},l_1)$ and $(\mathcal{X},l_1) \stackrel{\sigma_2}{\Rightarrow} (\{x,y\},l_1)$ from $(\mathcal{X},l_1)$ to $(\{x,y\},l_1)$ (see Example 4). It can be seen that, apart from σ_1 and σ_2, there are no other macro-steps between $(\mathcal{X},l_1)$ and $(\{x,y\},l_1)$ via any other sequence σ. We have $\mathsf{EndSet}((\mathcal{X},1),e_2,(\{x,y\},l_1)) = \{\sigma_1,\sigma_2\}$. Hence there is a macro-step $(\mathcal{X},l_1) \Rightarrow (\{x,y\},l_1)$ with $\mathsf{Wgt}((\mathcal{X},l_1) \Rightarrow (\{x,y\},l_1)) = 1-\alpha+\alpha(1-\beta) = 1-\alpha\beta$ and $\mathsf{Dur}((\mathcal{X},l_1) \Rightarrow (\{x,y\},l_1)) = p_2$.*

3.2 Building the Graph of Macro-steps

Intuitively, each macro-step corresponds to a set of sequences of transitions. The idea is now to transform the PDPTA into the "compact" form of a graph of macro-steps. In order to construct such a graph, there are *a priori* two possible techniques: (1) a forward reachability technique, which, given a point (X,l), constructs all the possible "successors" (Y,l') via $\Rightarrow$; (2) a backward reachability technique, which, given a point (X,l), constructs all the possible "predecessors" (Y,l') via $\Rightarrow$.

The *macro-step graph* is a sub-graph of the graph with vertices the set of points of the PDPTA $\mathcal{A}$, and the edges corresponding to the macro-step relation $\Rightarrow$ (that is, there exists an edge from point (X,l) to point (Y,l') if and only if $(X,l) \Rightarrow (Y,l')$). The macro-step graph is obtained by performing first a backward reachability analysis, then a subsequent forward reachability analysis. Given a point (X,l), the set of *predecessor points* of (X,l), denoted by $Pre(X,l)$, is defined by $\{(Y,l') \mid (Y,l') \Rightarrow (X,l)\}$. This set can be computed in the following way. For each edge $(l',X,l) \in \mathcal{E}_{(X,l)}$, we consider the clock $\phi(l')$; then we traverse the edges of the graph of the PDPTA backwards from l', terminating when an edge e which resets $\phi(l')$ is found. Then the point (Y,l''), where $e=(\_,Y,l'')$, is included in $Pre(X,l)$. The operator Pre can be generalized to sets of points: given the set V of points, we let $Pre(V) = \bigcup_{(X,l)\in V} Pre(X,l)$. Recalling that we are interested in computing the absorption time with regard to the point l_{end}, we compute iteratively $Pre^*(l_{end}) = \bigcup_{i\in\mathbb{N}} Pre^i(l_{end})$. Because the number of points of a PDPTA $\mathcal{A}$ is bounded by $|\mathcal{E}_\mathcal{A}|+1$, we have that there exists some $k \leq |\mathcal{E}_\mathcal{A}|+1$ such that $\bigcup_{i\in\mathbb{N}} Pre^i(l_{end}) = \bigcup_{i=1}^{k} Pre^i(l_{end})$. Note that this backward reachability analysis does not require the computation of sets of states of the PDPTA (in contrast to the zone-based algorithm for the full class of probabilistic timed automata of [25]), but instead relies on a syntactic analysis (involving checks of reset sets of edges and clock assignment functions) of the graph of the PDPTA.

A forward reachability analysis is then performed. Formally, given a point (X,l), we let $Post(X,l) = \{(Y,l') \mid (X,l) \Rightarrow (Y,l')\}$, and, given a set V of points, we let

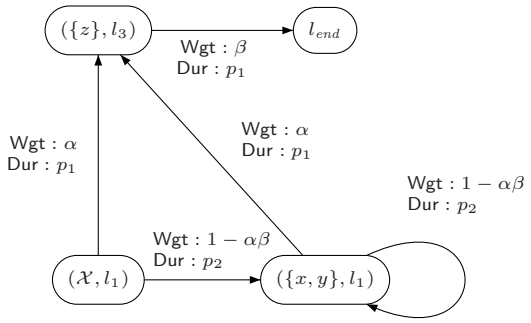

Figure 5. Graph of Macro-Steps for Example 1

$Post(V) = \bigcup_{(X,l)\in V} Post(X, l)$. Then we compute iteratively $Post^*(\bar{l}) = \bigcup_{i\in\mathbb{N}} Post^i(\bar{l})$ (as above, this computation terminates within $|\mathcal{E}_\mathcal{A}| + 1$ iterations). Finally, let $\mathcal{V} = Post^*(\bar{l}) \cap Pre^*(l_{end})$ be the set of vertices of the macro-steps graph. In the following, we consider only points in the set $\mathcal{V}$.

Example 6 *Consider the PDPTA of Example 1 (see Section 2), and let $l_{end} = l_4$. The graph of macro-steps is shown in Figure 5. Let us compute the set $Pre^*(l_{end})$ of iterated predecessors of l_{end}. The only macro-step arriving to l_{end} is $(\{z\}, l_3) \Rightarrow l_{end}$, and hence $Pre(l_{end}) = (\{z\}, l_3)$. From the fact that $\phi(l_3) = z$ and $\psi(l_3) = p_1$, we have that $\mathsf{Dur}((\{z\}, l_3) \Rightarrow l_{end}) = p_1$. Furthermore, we have $\mathsf{Wgt}((\{z\}, l_3) \Rightarrow l_{end}) = \beta$.*

We then compute $Pre(\{z\}, l_3)$. The macro-steps arriving at $(\{z\}, l_3)$ are of the form $(\{x, y\}, l_1) \Rightarrow (\{z\}, l_3)$ and $(\mathcal{X}, l_1) \Rightarrow (\{z\}, l_3)$, because (noting that l_1 is the source location of the only edge leading to l_3) $\phi(l_1) = y$, and the edge e_2 resets y to 0, as does the initialization of the PDPTA. We therefore have $Pre(\{z\}, l_3) = \{(\{x, y\}, l_1), (\mathcal{X}, l_1)\}$. Furthermore $\mathsf{Dur}((\{x, y\}, l_1) \Rightarrow (\{z\}, l_3)) = \mathsf{Dur}((\mathcal{X}, l_1) \Rightarrow (\{z\}, l_3)) = p_1$, and $\mathsf{Wgt}((\{x, y\}, l_1) \Rightarrow (\{z\}, l_3)) = \mathsf{Wgt}((\mathcal{X}, l_1) \Rightarrow (\{z\}, l_3)) = \alpha$.

In the next step, we obtain $Pre(\{x, y\}, l_1) = \{(\{x, y\}, l_1), (\mathcal{X}, l_1)\}$. This follows from the fact that, as seen in Example 5, there is a macro-step $(\mathcal{X}, l_1) \Rightarrow (\{x, y\}, l_1)$ (with $\mathsf{Wgt}((\mathcal{X}, l_1) \Rightarrow (\{x, y\}, l_1)) = 1 - \alpha\beta$ and $\mathsf{Dur}((\mathcal{X}, l_1) \Rightarrow (\{x, y\}, l_1)) = p_2$). Furthermore, there is a macro-step $(\{x, y\}, l_1) \Rightarrow (\{x, y\}, l_1)$ (again with $\mathsf{Wgt}((\{x, y\}, l_1) \Rightarrow (\{x, y\}, l_1)) = 1 - \alpha\beta$ and $\mathsf{Dur}((\{x, y\}, l_1) \Rightarrow (\{x, y\}, l_1)) = p_2$).

This ends the process of generating $Pre^*(l_{end})$. By inspection of the macro-steps graph in Figure 5, and recalling that $\bar{l} = (\mathcal{X}, l_1)$, we can see that $Post^*(\bar{l}) \cap Pre^*(l_{end}) = Pre^*(l_{end})$, hence $\mathcal{V} = Pre^*(l_{end})$.

3.3 Computation of Expected Absorption Times

A *macro path* is a sequence $\tau = (X_0, l_0) \Rightarrow (X_1, l_1) \Rightarrow \cdots \Rightarrow (X_m, l_m)$, where all points visited along the sequence belong to $\mathcal{V}$. We say that the *length of* τ, denoted by $|\tau|$, is m. We define the *weight of* τ, denoted by $\pi(\tau)$, and the *duration of* τ, denoted by $\delta(\tau)$ as:

$$\pi(\tau) = \prod_{k=0}^{m-1} \mathsf{Wgt}((X_k, l_k) \Rightarrow (X_{k+1}, l_{k+1}))$$

$$\delta(\tau) = \sum_{k=0}^{m-1} \mathsf{Dur}((X_k, l_k) \Rightarrow (X_{k+1}, l_{k+1}))$$

Recall that $\delta(\tau)$ will take the form of a linear expression over parameters with natural coefficients.

Let (X, l) be a point, let $\mathsf{MPaths}(X, l)$ be the set of macro paths from (X, l) to l_{end} in the macro-steps graph, and let:

$$\mathsf{MExpAbs}(X, l) = \sum_{\tau\in\mathsf{MPaths}(X,l)} \delta(\tau) \cdot \pi(\tau) \ .$$

Note that, as $\delta(\tau)$ is a linear expression over parameters with natural coefficients, and $\pi(\tau)$ is a rational number, $\mathsf{MExpAbs}(X, l)$ is a linear expression over parameters with rational coefficients. The value $\mathsf{MExpAbs}(X, l)$ is related to the notion of "average cost" on finite Markov chains with *costs* (or rewards) [5, 6]. We define $\mathsf{MExpAbs}(\mathcal{A}) = \mathsf{MExpAbs}(\bar{l})$, where $\bar{l}$ is the initial location of $\mathcal{A}$.

Proposition 2 *Let $\mathcal{A}$ be a PDPTA and κ be a parameter valuation. We have $\mathsf{ExpAbs}(\mathcal{A}, \kappa) = \kappa(\mathsf{MExpAbs}(\mathcal{A}))$.*

We now provide a method for computing $\mathsf{MExpAbs}(\mathcal{A})$. Let $\mathcal{V}' = \mathcal{V} \setminus \{l_{end}\}$. We assume that the elements of $\mathcal{V}$ are ordered (with $\bar{l}$ as the least element). Consider the $|\mathcal{V}'| \times |\mathcal{V}'|$-matrix $\mathbf{M}$ defined by $\mathbf{M}((X, l), (Y, l')) = \mathsf{Wgt}((X, l) \Rightarrow (Y, l'))$ for all $(X, l), (Y, l') \in \mathcal{V}'$ such that $(X, l) \Rightarrow (Y, l')$, otherwise $\mathbf{M}((X, l), (Y, l')) = 0$. We introduce, for each $(X, l) \in \mathcal{V}'$, a "correcting factor", denoted by $w(X, l)$, defined by:

$$w(X, l) = \sum_{\tau\in\mathsf{MPaths}(X,l)} \pi(\tau) \ .$$

The correcting factor $w(X, l)$ is the total weight of the paths reaching l_{end} from (X, l).

Lemma 1 *Let $W = (w(X, l))_{(X,l)\in\mathcal{V}'}$. Let B be the $|\mathcal{V}'|$-dimensional vector equal to $(\mathsf{Wgt}((X, l) \Rightarrow l_{end}))_{(X,l)\in\mathcal{V}'}$. Then the vector W is the unique solution of the system $W = \mathbf{M}W + B$.*

For any two points $(X,l),(Y,l') \in \mathcal{V}'$, let $\zeta((X,l),(Y,l'))$ equal:

$$\mathsf{Dur}((X,l) \Rightarrow (Y,l')) \cdot \mathbf{M}((X,l),(Y,l')) \cdot w(Y,l') \ ,$$

and let $\eta(X,l)$ equal:

$$\mathsf{Dur}((X,l) \Rightarrow l_{end}) \cdot \mathsf{Wgt}((X,l) \Rightarrow l_{end}) \ .$$

Proposition 3 *Let C be the $|\mathcal{V}'|$-dimensional vector equal to:*

$$\left(\sum_{(Y,l') \in \mathcal{V}'} \zeta((X,l),(Y,l')) + \eta(X,l) \right)_{(X,l) \in \mathcal{V}'}$$

Then the vector $T = (\mathsf{MExpAbs}(X,l))_{(X,l) \in \mathcal{V}'}$ is the unique solution of the system $T = \mathbf{M}T + C$.

Let us point out that Proposition 3 allows us to compute $\mathsf{MExpAbs}(\mathcal{A})$ (and, from Proposition 2, hence $\mathsf{ExpAbs}(\mathcal{A},\kappa)$) in a *parametric* manner with the parameters of "cost" appearing in row vector C.

Example 7 *Consider the macro-steps graph of Figure 5. Here $\mathcal{V}' = \{(\mathcal{X},l_1),(\{x,y\},l_1),(\{z\},l_3)\}$. Let I be the identity matrix of size $|\mathcal{V}'|$. The vector B is then given by $B_{(\mathcal{X},l_1)} = \mathsf{Wgt}((\mathcal{X},l_1) \Rightarrow l_{end}) = 0$, $B_{(\{x,y\},l_1)} = \mathsf{Wgt}((\{x,y\},l_1) \Rightarrow l_{end}) = 0$, and $B_{(\{z\},l_3)} = \mathsf{Wgt}((\{z\},l_3) \Rightarrow l_{end}) = \beta$. The vector W is given by $(I - \mathbf{M})^{-1}B = (1,1,\beta)$. Then the vector C is given by $C_{(\mathcal{X},l_1)} = p_2 \cdot \mathbf{M}((\mathcal{X},l_1),(\{x,y\},l_1)) \cdot 1 + p_1 \cdot \mathbf{M}((\mathcal{X},l_1),(\{z\},l_3)) \cdot \beta = (1-\alpha\beta)p_2 + \alpha\beta p_1$, $C_{(\{x,y\},l_1)} = p_2 \cdot \mathbf{M}((\{x,y\},l_1),(\{x,y\},l_1)) \cdot 1 + p_1 \cdot \mathbf{M}((\{x,y\},l_1),(\{z\},l_3)) \cdot \beta = (1-\alpha\beta)p_2 + \alpha\beta p_1$, and $C_{(\{z\},l_3)} = p_1 \cdot \mathsf{Wgt}((\{z\},l_3) \Rightarrow l_{end}) = \beta p_1$. Finally the vector T, computed as $(I - \mathbf{M})^{-1}C$, is given by $T_{(\mathcal{X},l_1)} = \frac{1-\alpha\beta}{\alpha\beta}p_2 + 2p_1$, $T_{(\{x,y\},l_1)} = \frac{1-\alpha\beta}{\alpha\beta}p_2 + 2p_1$, and $T_{(\{z\},l_3)} = \beta p_1$. Given that $\mathsf{MExpAbs}(\mathcal{A}) = \mathsf{MExpAbs}(\bar{l})$, and the point $\bar{l}$ is here $(\mathcal{X},l_1)$, we have that $\mathsf{MExpAbs}(\mathcal{A}) = \frac{1-\alpha\beta}{\alpha\beta}p_2 + 2p_1$. Using Proposition 2, we can then obtain $\mathsf{ExpAbs}(\mathcal{A},\kappa)$.*

Observe that the complexity of constructing the graph of macro-steps of a given PDPTA is polynomial in the number of edges. First note that, in order to generate the macro-step relation $\Rightarrow$, we must avoid an explicit computation of the sets $\mathsf{EndSet}()$, which can be of size exponential in $|\mathcal{E}_{\mathcal{A}}|$. However, our technique does not require such an explicit representation: in particular, for points $(X,l),(Y,l')$ and edge $(l'',Y,l') \in \mathcal{E}_{\mathcal{A}}$, the weight $\mathsf{Wgt}((X,l) \stackrel{(l'',Y,l')}{\Longrightarrow} (Y,l'))$ of Definition 8 can be computed as $\Pr(\{\omega \mid \mathit{first}(\omega) = l \wedge \mathit{last}(\omega) = l''\}) \cdot \mathbf{prob}(l'')(Y,l')$, where $\mathit{first}(\omega)$ denotes the first element of a path, and where $\Pr(\{\omega \mid \mathit{first}(\omega) = l \wedge \mathit{last}(\omega) = l''\})$ can be computed with standard probabilistic reachability algorithms on the underlying, untimed Markov chain of the PDPTA. Second, note that the number of points of $\mathcal{A}$ is, by definition, bounded by $|\mathcal{E}_{\mathcal{A}}| + 1$, and hence the computation of $\mathcal{V}$ can be done in time polynomial in $|\mathcal{E}_{\mathcal{A}}|$. Finally, we note that the computation of the expected absorption time requires the solution of two sets of linear equations (one set for Lemma 1, the other for Proposition 3), the number of each of which is bounded by $|\mathcal{V}|$, and therefore can be done in polynomial time. The computation of the expected time of a PDPTA can thus be done in polynomial time. In particular, we note that the time complexity of the technique does not depend exponentially on the number of clocks of the PDPTA, nor on the magnitude of the timing constants used, as is the case for general probabilistic timed automata [23, 28].

3.4 Example: CSMA/CD

We applied our technique to the protocol CSMA/CD (Carrier Sense Multiple Access/Collision Detection), as modeled in [22, 1]. We take the case when there are two stations and trying to send data at the same time through a channel. As in [1], we model the case when the stations collide initially. If there is no collision, then, after λ time units, the station finishes sending its data. On the other hand, if there is a collision, 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 equal to 2θ time units in length) in which the station waits after the nth transmission failure is chosen as a uniformly distributed random integer in $\{0,1,...,2^{k+1}-1\}$, where $k = \min(n, bcmax)$, with $bcmax$ taken here equal to 2, as in experiments of [1]. The overall model is a PDPTA obtained by a parallel composition of three probabilistic timed automata representing the two senders and the channel.

We compute $\mathsf{ExpAbs}(\mathcal{A},\kappa)$, the expected time for one of the sender to finish the sending of its data, via $\mathsf{MExpAbs}(\mathcal{A})$ using the macro-steps graph and Proposition 3: the macro-steps graph is, in this particular case, a Markov chain, and therefore we know the vector W is the vector with all components equal to 1. We use `Maple` to compute the inverse of matrix $I - \mathbf{M}$, which is a $(59,59)$-matrix, and to compute $T = (I - \mathbf{M})^{-1}C$. We obtain that $\mathsf{MExpAbs}(\mathcal{A}) = (30/7)\theta + \lambda$. We have also checked the correctness of this formula by computing the relevant expected time for different values of λ and θ in PRISM [1]. For details, see [10].

4 Conclusions

We have explained how to compute the expected time of reaching an absorbing node for the class of parametric

determinate probabilistic timed automata. This method is parametric regarding the timing constants appearing in the automaton, and can be done in polynomial time in the size of the system description. We claim that, often in practice, the worst behavior of a general probabilistic timed automaton modelling a protocol can be captured under the form of a PDPTA, perhaps obtained by manual intervention in order to remove nondeterminism. We conjecture that the requirement of structural non-Zenoness could be lifted from PDPTA.

References

[1] Prism web site. http://www.prismmodelchecker.org/.

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

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

[4] C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Proc. QEST'04*, pages 230–239. IEEE Computer Society, 2004.

[5] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[6] D. Bertsekas and J. Tsitsiklis. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research 16:3*, pages 580–595, 1991.

[7] L. Bozzelli and S. La Torre. Decision problems for lower/upper bound parametric timed automata. In *Proc. ICALP'07*, volume 4596 of *LNCS*, pages 925–936. Springer, 2007.

[8] P. Brémaud. *Markov Chains*. Springer, 1999.

[9] V. Bruyère and J.-F. Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1), 2007.

[10] N. Chamseddine, M. Duflot, L. Fribourg, and C. Picaronny. Determinate probabilistic timed automata as Markov chains with parametric costs. Technical Report LSV-07-21, Laboratory Specification and Verification, May 2007.

[11] N. Chamseddine, M. Duflot, L. Fribourg, C. Picaronny, and J. Sproston. Computing expected absorption times for parametric determinate probabilistic timed automata. Technical Report LSV-08-22, Laboratory Specification and Verification, July 2008.

[12] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability Analysis of Probabilistic Systems by Successive Refinements. In *PAPM-PROBMIV'01, LNCS 2165, Springer*, pages 39–56, 2001.

[13] P. D'Argenio, J.-P. Katoen, T. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proc. TACAS'97*, volume 1217 of *LNCS*, pages 416–431. Springer, 1997.

[14] P. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Proc. QEST'04*, pages 240–249. IEEE Computer Society, 2004.

[15] C. Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *Proc. ICTAC'04*, volume 3407 of *LNCS*, pages 280–294. Springer, 2007.

[16] M. Fruth. Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol. In *Proc. ISOLA'06*, 2006.

[17] F. He, L. Baresi, C. Ghezzi, and P. Spoletini. Formal analysis of publish-subscribe systems by probabilistic timed automata. In *Proc. FORTE'07*, volume 4574 of *LNCS*, pages 247–262. Springer, 2007.

[18] R. A. Howard. *Dynamic Probabilistic Systems*. John Wiley and Sons, 1971.

[19] T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programmming*, 52-53:183–220, 2002.

[20] H. E. Jensen. Model checking probabilistic real time systems. In *Proc. Nordic Workshop on Programming Theory*, pages 247–261. Chalmers Institute of Technology, 1996.

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

[22] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance Analysis of Probabilistic Timed Automata using Digital Clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

[23] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002.

[24] 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.

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

[26] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Parametric probabilistic transition systems for system design and analysis. *Formal Aspects of Computing*, 19(1):93–109, 2007.

[27] F. Laroussinie and J. Sproston. Model checking durational probabilistic systems. In *Proc. FOSSACS'05*, volume 3441 of *LNCS*, pages 140–154. Springer, 2005.

[28] F. Laroussinie and J. Sproston. State explosion in almost-sure probabilistic reachability. *Information Processing Letters*, 102(6):236–241, 2007.

[29] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[30] D. P. L. Simons and M. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. *Software Tools for Technology Transfer*, 3(4):469–485, 2001.

[31] M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. In *Proc. ARTS'99*, volume 1601 of *LNCS*, pages 53–74. Springer, 1999.

[32] S. Tripakis, S. Yovine, and A. Bouajjani. Checking timed Büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, 2005.