

Parameter Synthesis for Probabilistic Timed Automata Using Stochastic Game Abstractions

Aleksandra Jovanović, Marta Kwiatkowska

Department of Computer Science, University of Oxford, Oxford, UK

Abstract

We propose a symbolic method to synthesise optimal values of timing parameters for probabilistic timed automata, in the sense that the probability of reaching some set of states is either maximised or minimised. Our first algorithm, based on forward exploration of the symbolic states, can only guarantee parameter values that correspond to upper (resp. lower) bounds on maximum (resp. minimum) reachability probability. To ensure precise reachability probabilities, we adapt the game-based abstraction refinement method. In the parametric setting, our method is able to determine all the possible maximum or minimum reachability probabilities that arise for different values of timing parameters, and yields optimal valuations represented as a set of symbolic constraints between parameters. We report on a prototype implementation of the algorithm in the PRISM model checker and its evaluation on a case study.

Keywords: Model checking, parameter synthesis, probabilistic reachability, probabilistic timed automata, Markov decision processes, stochastic games

1. Introduction

Stochastic aspects are very important for modelling numerous classes of systems, including communication, network and security protocols, due to component failures, unreliable channels or randomisation. The correctness of such systems can be guaranteed only with some probability. Many of them also operate under certain timing constraints. In such cases, the probability of a property being true depends on the timing aspects in the system: for example, increasing a certain delay might increase the maximum or minimum probability of reaching an error state.

Automatic synthesis of timing constraints to ensure the satisfaction of a given property has received a lot of attention lately. Its aim is to overcome the disadvantage of model checking, which requires complete knowledge of the system. This is often difficult to obtain, especially in the early design stages, when the whole environment is not yet known. The use of parameters instead of concrete values gives more freedom to the designers. A parametric timed model can specify that a transition is enabled for a time units, or that a system can

stay in a location for b time units, where a and b are parameters. The goal is then to automatically synthesise the values of model’s parameters such that its behaviour is guaranteed to satisfy the specification. Parameterisation, however, makes verification more difficult, as most problems become undecidable.

In this paper, we consider the synthesis of timing parameters for probabilistic real-time systems modelled as probabilistic timed automata (PTA) [1]. PTA have been introduced as an extension of timed automata (TA) [2] for modelling and analysing systems which exhibit real-time, nondeterminism and probabilistic behaviour. They are finite-state automata extended with clocks, real-valued variables which increase at the same, constant rate. The edge relation of a PTA differs from that of a TA, in the sense that purely nondeterministic choice over the set of edges is replaced by a set of discrete probability distributions, each of which is defined over a finite set of edges. A fundamental property of PTA is the maximum/minimum probability of reaching a certain set of states in the model (i.e. the reachability probabilities). These probabilities allow one to express a broad range of properties, from quality of service to reliability, for example, the deadline properties: “the maximum probability of an airbag failing to deploy within 0.02 seconds” or “the minimum probability that a packet is correctly delivered within 1 s”. As shown in [1], model checking of more complex properties can be reduced to probabilistic reachability.

PTA arise naturally in distributed coordination and wireless communication protocols, and have been successfully used to analyse several protocols, including IEEE 1394 FireWire, Bluetooth and IEEE 802.11 WiFi, that rely on the use of randomisation and timing delays. The analyses were performed using the probabilistic model checker PRISM, which provides native support for PTA through a variety of techniques, including symbolic zone-based methods. Since these protocols are embedded in a networked environment, their properties are almost always expressed parametrically, as concrete values make sense only when the network environment is known. Further, the choice of values of timing parameters may affect the probability of certain properties to be satisfied, as has been demonstrated for IEEE 1394 FireWire in [3] using PRISM. The process of enumerating all the possible values of parameters, assuming a restriction to bounded integers/rationals, and performing verification for each instantiation is time consuming and error-prone. It is thus desirable to be able to automatically derive the constraints on parameters for probabilistic real-time systems, so that their correctness is ensured with optimal probability.

Contributions We propose an algorithm for parameter synthesis for PTA based on the symbolic zone-based exploration of the underlying probabilistic reachability graph. As the forward approach gives only upper (resp. lower) bounds on maximum (resp. minimum) reachability probability, we adapt the game-based abstraction refinement method. This method has been introduced in [4] for Markov decision processes, and extended in [5] for PTA, for the computation of exact maximum/minimum reachability probabilities. As we consider parametric models, these probabilities are not unique and depend on particular parameter valuations. In the case of a *negative* specification, such as “the maximum probability of a message being lost”, we are typically interested in

the maximum probabilities, which we want to minimise, while in the case of a *positive* specification, such as “the minimum probability of message being received”, we are interested in the minimum probabilities, which we want to maximise. Our algorithm derives a finite set of symbolic constraints on parameters for which the probability is either maximised or minimised, thus allowing us to choose optimal parameter valuations. We implement the algorithm as an extension of the PRISM model checker [6], where a PTA can be input as a parametric model and the analysis proceeds through exploration of the underlying parametric zone graph. To the best of our knowledge, this is the first paper dealing with optimal timing parameter synthesis for probabilistic timed automata.

A preliminary version of this paper appeared as [7]. This paper extends [7] with full proofs, as well as an implementation and integration of the algorithm within PRISM, and its evaluation on a case study.

Related work Parameter synthesis for untimed probabilistic models includes [8], now implemented in PRISM [9], where transition probabilities are considered as parameters for Markov chains. Given a property specified in some probabilistic logic, the goal is to find the values of parameters such that the formula holds in the model. Our work is orthogonal to this framework. We consider fixed probabilities and aim to synthesise timing constraints which maximise or minimise some reachability probability in the system.

Concerning non-probabilistic timed systems, parametric timed automata have been introduced in [10] as a means to specify parametric timing constraints. The *reachability-emptiness* problem, which asks whether there exists a parameter valuation such that the automaton has an accepting run, is undecidable. In [11], the undecidability proof is extended for parametric timed automata that use only strict inequalities. Subsequent research has thus concentrated on finding subclasses for which certain problems are decidable by restricting the use of parameters [12, 13] or by restricting the parameter domain [14]. In [15], the authors deal with deterministic networks of timed automata with priorities and parametric guards and develop an algorithm based on constraint solving and Monte Carlo sampling to synthesise timing delays for MTL extended with counting formulas. In [16], optimal synthesis of timing parameters is considered for systems modelled as (deterministic) networks of timed I/O automata and quantitative objectives, such as energy consumption, formulated as a bilevel optimisation problem. An approach for the verification of Parametric TCTL (PTCTL) formulae has been developed in [17], where the problem has been proved decidable. A more general problem is studied in [18], where parameters are allowed both in the model and the desired PTCTL property. The authors show that the model checking problem is decidable and the parameter synthesis problem is solvable, in discrete time, over a PTA with one parametric clock, if equality is not allowed in the formulae. In [19], it is shown how bounded model checking can be applied to parameter synthesis for timed automata to synthesise part of the set of all the parameter valuations under which the given property holds in a model. A construction that enables the synthesis of an implementation of a specification for real-time systems based on timed I/O automata,

which is robust under a given timed perturbation, is presented in [20].

There is little work, however, on timing parameter synthesis for probabilistic real-time systems. In [21] the authors presented an approach based on the decomposition of the parametric space into behavioural tiles, i.e., sets of parameter valuations for which the behaviour of the system is uniform. The method is also extended for probabilistic systems. In [22], a technique is proposed to approximate parametric rate values for continuous-time Markov chains for bounded reachability probabilities, implemented in a GPU-based tool PRISM-PSY [23]. In [24], the authors apply their *Inverse* method for parameter synthesis for TA to PTA. The method starts from reference parameter values of a TA, and derives the constraints on parameters such that the obtained models are time-abstract equivalent. Time-abstract equivalence preserves untimed properties, and thus the parameter values derived on the non-probabilistic version of the model preserve reachability probabilities. Termination is not guaranteed and the derived constraints are not weakest in general. In [25], the authors consider a fully deterministic parametric model, where the remaining time in a node is unique and given as a parameter, and provide a method to compute the expected time to reach some node as a function of model's parameters. Our approach is the first zone-based method that can handle fully nondeterministic PTA for general probabilistic reachability properties.

Probabilistic timed automata are supported by the PRISM model checker [6], where models are specified in a guarded command language and properties in PCTL, a probabilistic extension of CTL. PTA model checking is supported by several engines in PRISM. The zone-based *stochastic games* engine [5] and the *backwards reachability* engine [26] allow time-bounded or unbounded probabilistic reachability properties, where the former currently only handles maximum probabilities. In order to use these engines, the model must satisfy the following restrictions. It should be well-formed, non-Zeno and should not exhibit time-locks, i.e., the possibility of reaching a state where no transitions are possible and time cannot elapse beyond a certain point [3]. The model must also have a single initial state. The *digital clocks* engine [27] implements a time-abstract approach which allows unbounded properties and clock variables, but does not yet support time-bounded reachability properties. For the digital clocks engine, clock constraints cannot use strict comparison operators, and diagonal clock constraints are not allowed, i.e. those containing references to two clocks. We have enabled parameter synthesis for PTA for the *stochastic games* engine, and our implementation is described in Section 5.

2. Preliminaries

A discrete probability distribution over a set S is a function $\mu : S \mapsto [0, 1]$, such that $\sum_{s \in S} \mu(s) = 1$ and the set $\{s \mid s \in S \wedge \mu(s) > 0\}$ is finite. By $\text{Dist}(S)$ we denote the set of such distributions. μ_p is a point distribution if $\mu_p(s) = 1$ for some $s \in S$. We now define *Markov decision processes*, a formalism for modelling systems which exhibit both nondeterministic and probabilistic behaviour.

Definition 1 (Markov decision processes). An MDP is a tuple $\mathcal{M} = (S, s_0, \Sigma, \text{Steps}_{\mathcal{M}})$, where S is a set of states, s_0 is a set of initial states, Σ is a set of actions and $\text{Steps}_{\mathcal{M}} : S \times \Sigma \mapsto \text{Dist}(S)$ is a probabilistic transition function.

A transition in \mathcal{M} from state s is first made by nondeterministically selecting an action $\delta \in \Sigma$ and then the successor state s' is chosen randomly according to the probability distribution $\text{Steps}_{\mathcal{M}}(s, \delta)$. A *path* is a sequence of such transitions and represents a particular resolution of both nondeterminism and probability. A state s is *reachable* in \mathcal{M} if there exists a path from the initial state of \mathcal{M} to s . A strategy \mathcal{A} is a function from finite paths to distributions which resolves nondeterminism in an MDP. For a fixed strategy \mathcal{A} , the behaviour of an MDP is purely probabilistic, and we can define the probability $p_s^{\mathcal{A}}(F)$ of reaching a target set $F \subseteq S$ from s under \mathcal{A} . By quantifying over all strategies in \mathcal{M} , we can define the minimum and maximum probability of reaching F :

$$p_{\mathcal{M}}^{\min}(F) = \inf_{s \in s_0} \inf_{\mathcal{A}} p_s^{\mathcal{A}}(F) \text{ and } p_{\mathcal{M}}^{\max}(F) = \sup_{s \in s_0} \sup_{\mathcal{A}} p_s^{\mathcal{A}}(F)$$

These values can be computed efficiently together with the corresponding strategies using, e.g., *value iteration* method, which approximates the probability value.

Stochastic 2-player games [28] are turn-based games involving two players and probability. They generalise MDPs by allowing two types of nondeterministic choice, each controlled by a separate player.

Definition 2 (Stochastic games). A stochastic game is a tuple $\mathcal{G} = (S, (S_1, S_2), s_0, \Sigma, \text{Steps}_{\mathcal{G}})$, where S is a set of states partitioned into sets S_1 and S_2 , s_0 is a set of initial states, Σ is a set of actions and $\text{Steps}_{\mathcal{G}} : S_1 \times \Sigma \times S_2 \mapsto 2^{\text{Dist}(S)}$ is a probabilistic transition function.

S_1 and S_2 represent the sets of states controlled by player 1 and player 2, respectively. The behaviour of a game is as follows. First, player 1, in state $s \in S_1$, selects an available action $\delta \in \Sigma$, which takes the game into a state $s' \in S_2$. Player 2 then selects a probability distribution μ from the set $\text{Steps}_{\mathcal{G}}(s, \delta, s')$. Finally, the successor state s'' is chosen according to μ . A resolution of nondeterminism in \mathcal{G} is a pair of strategies σ_1, σ_2 for player 1 and player 2, respectively, under which we can define the probability $p_s^{\sigma_1, \sigma_2}(F)$ of reaching a subset $F \subseteq S$ from state s . A pair of strategies induces a probability measure over the set of paths [4].

Let \mathbb{R} , $\mathbb{R}_{\geq 0}$ and \mathbb{Z} be the sets of reals, non-negative reals and integers, respectively. Let X be a finite set. A linear expression on X is an expression of the form $\lambda := k \mid k \cdot x \mid \lambda + \lambda$, where $k \in \mathbb{Z}$ and $x \in X$.

Now let $X = \{x_1, \dots, x_n\}$ be a finite set of clock variables. A clock valuation $u : X \mapsto \mathbb{R}_{\geq 0}$ is a function assigning a non-negative real number to each $x \in X$. Let $\vec{0}$ be a valuation that assigns 0 to all clocks in X . For any $R \subseteq X$ and any valuation u on X , we write $u[R]$ for the valuation on X such that $u[R](x) = 0$ if $x \in R$ and $u[R](x) = u(x)$ otherwise. For $t \geq 0$, $u + t$ denotes the valuation

which assigns $(u + t)(x) = u(x) + t$ to all $x \in X$. Let $P = \{p_1, \dots, p_m\}$ be a finite set of parameters. A (linear parametric) constraint on $X \cup P$ is an expression of the form $\gamma := x_i \sim c \mid x_i - x_j \sim c \mid \gamma \wedge \gamma'$ where $1 \leq i \neq j \leq n$, $x_i, x_j \in X$, $\sim \in \{<, \leq\}$ and c is a linear expression on P . By $\mathcal{C}(X, P)$ we denote the set of such parametric constraints and by $\mathcal{C}'(X, P)$ we denote the set of (non-diagonal) constraints of the form: $\gamma' := x_i \sim c \mid \gamma' \wedge \gamma'$. For any valuation v on P and any linear constraint γ on $X \cup P$, $v(\gamma)$ is the linear constraint on X obtained by replacing each parameter $p \in P$ by the (concrete) value $v(p)$. Given some arbitrary order on $X \cup P$, a valuation can be viewed as a real-valued vector of size $|X \cup P|$. The set of valuations satisfying some linear constraints is then a convex polyhedron of $\mathbb{R}^{|X \cup P|}$. A zone is a polyhedron defined only by conjunctions of the constraints of the form $x - y \sim c$ or $x \sim c$ with $x, y \in X$, $c \in \mathbb{Z}$ and $\sim \in \{<, \leq\}$. If v is a valuation on both clocks and parameters $X \cup P$ (as v is used throughout the paper, unless specified otherwise) then by $v|_P$ (resp. $v|_X$) we denote the projection of v onto P (resp. X). We now give a formal definition of *parametric probabilistic timed automata* (PPTA), which are PTA extended with timing parameters.

Definition 3 (PPTA). A PPTA is a tuple $\mathcal{P} = (L, l_0, X, P, \Sigma, \text{prob}, \text{Inv})$ where L is a finite set of locations, $l_0 \in L$ is the initial location, X is a finite set of clocks, P is a finite set of parameters, Σ is a finite set of actions, $\text{prob} : L \times \Sigma \times \mathcal{C}(X, P) \mapsto \text{Dist}(2^X \times L)$ is a probabilistic transition function, and $\text{Inv} : L \mapsto \mathcal{C}'(X, P)$ is a function that assigns an invariant to each location.

For any rational valuation v on P , the structure $v(\mathcal{P})$ obtained from \mathcal{P} by replacing every constraint γ by $v(\gamma)$ is a PTA. The behaviour of a PPTA \mathcal{P} is described by the behaviour of all PTA $v(\mathcal{P})$ obtained by considering all possible parameter valuations. A (concrete) state of $v(\mathcal{P})$ is a pair $(l, u) \in L \times \mathbb{R}_{\geq 0}^X$ such that the clock valuation u satisfies the invariant (notation $u \models v(\text{Inv}(l))$). A transition in the semantics of $v(\mathcal{P})$ is a timed-action pair (t, δ) . In a state certain amount of time $t \in \mathbb{R}_{\geq 0}$ can elapse as long as $u + t \models v(\text{Inv}(l))$. Time elapse is followed by the choice of an action $\delta \in \Sigma$, for which the set of clocks R to reset and successor locations l' are selected randomly according to the probability distribution $\text{prob}(l, \delta, \gamma)$. The action δ can only be taken once its constraint $v(\gamma)$ (called guard) is satisfied by the current clock valuation. Each element $(R, l') \in 2^X \times L$, such that $\text{prob}(l, \delta, \gamma)(R, l') > 0$, is called an edge and the set of all such edges, denoted $\text{edges}(l, \delta, \gamma)$, is assumed to be an ordered list $\langle e_1, \dots, e_n \rangle$. We now formally define the semantics of a PPTA under a parameter valuation v .

Definition 4 (Semantics of a PPTA). Let $\mathcal{P} = (L, l_0, X, P, \Sigma, \text{prob}, \text{Inv})$ be a PPTA and v be a \mathbb{R} -valuation on P ($v : P \mapsto \mathbb{R}$). The semantics of $v(\mathcal{P})$ is given by the infinite-state MDP $\mathcal{M}_{v(\mathcal{P})} = (Q, q_0, \mathbb{R}_{\geq 0} \times \Sigma, \text{Steps}_{\mathcal{M}_{v(\mathcal{P})}})$ where:

- $Q = \{(l, u) \in L \times \mathbb{R}_{\geq 0}^X \mid u \models v(\text{Inv}(l))\}$, $q_0 = (l_0, \vec{0})$
- $\text{Steps}_{\mathcal{M}_{v(\mathcal{P})}}((l, u), (t, \delta)) = \mu$ iff $\exists (R, l') \in \text{edges}(l, \delta, \gamma)$ such that $u + t \models v(\gamma) \wedge u + t' \models \text{Inv}(l')$ for all $0 \leq t' \leq t$, and for any $(l', u') \in Q$:

$$\mu(l', u') = \sum \{ |\text{prob}(l, \delta, \gamma)(R, l') \mid R \in 2^X \wedge u' = (u + t)[R] | \}$$

Note that the definition of μ involves summation over the cases in which multiple clock resets result in the same target state (l', u') , expressed as a multiset, since some of the probabilities might be the same.

We study the *optimal timing parameter synthesis* problem for PPTA, i.e., automatically finding values of parameters such that the probability (either maximum or minimum) of reaching a certain set of locations is *optimised*. For example, in the case of property “the maximum probability of an airbag failing to deploy”, we would want to choose the timing parameters that minimise this probability value. On the other hand, we would want to maximise “the minimum probability that the protocol successfully terminates”.

3. Synthesis with Forward Reachability

A naive approach to parameter synthesis for PTA is to restrict parameter values to bounded intervals of integers (or rationals that can be scaled to integers) and perform verification for each such (non-parametric) model - basically swapping the parameters for each verification, using a probabilistic model checker, e.g. PRISM [6]. However, this approach is already experimentally shown to be inefficient for (non-probabilistic) TA compared to symbolic techniques, especially when the sets of possible parameter values are large, in [14]. This is why we aim to formulate a symbolic algorithm for deriving constraints on parameters that ensure the optimisation of some reachability probability in the model. For the symbolic exploration of the state-space, we use the notion of a *parametric symbolic state* and symbolic operations on valuation sets given below, defined in [14].

Definition 5 (Parametric symbolic state). A (parametric) symbolic state of a PPTA \mathcal{P} , with the set of clocks X and the set of parameters P , is a pair $S = (l, \zeta)$ where l is a location of \mathcal{P} and ζ is a set of valuations v on $X \cup P$. Sometimes, when it is clear from the context, we refer to ζ as a symbolic state.

For the computation of the set of symbolic states of a PPTA we need several operations defined below.

- *future* (time successors): $\zeta^\nearrow = \{v' \mid v \in \zeta \wedge v'(x) = v(x) + d, d \geq 0 \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$
- *reset of clocks* in $R \subseteq X$: $\zeta[R] = \{v[R] \mid v \in \zeta\}$
- *successor by edge* $e = (R, l')$ in the distribution $\text{prob}(l, \delta, \gamma)$: $\text{Succ}((l, \zeta), e) = (l', (\zeta \cap \gamma)[R]^\nearrow \cap \text{Inv}(l'))$
- *initial symbolic state*: $\text{Init}(\mathcal{P}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v|_X \in \{\vec{0}_X\}^\nearrow \wedge v(\text{Inv}(l_0))\})$.

The sets of valuations ζ of all reachable symbolic states of a PPTA are convex polyhedra [12], since the set of valuations of the initial symbolic state is a convex polyhedron and all the operations preserve convexity.

3.1. Forward reachability exploration

The forward exploration, which builds an MDP-based abstraction of a given PTA [1], is an extension of the well-known zone-based forward reachability algorithm, ubiquitous for model-checking TA and implemented in tools such as UPPAAL [29] and KRONOS [30]. This algorithm performs the exploration of the state-space by successively computing symbolic states using **Succ**, starting from the initial symbolic state. For probabilistic models, this algorithm exhaustively explores the reachability graph and on-the-fly techniques are not used, as the goal is to compute the probability of reaching a state, instead of just checking the existence of a path.

In Figure 1 we present our extension of the forward reachability algorithm from [1] to parametric probabilistic timed automata. It takes a PPTA \mathcal{P} and some target subset of its locations F as input, and returns the *reachability graph* $(Sym, Trans)$. Sym is the set of all reachable parametric symbolic states S of the model and $Trans$ is the set of symbolic transitions. *Waiting* is the set of those symbolic states which have not yet been explored. As long as there are symbolic states unexplored ($Waiting \neq \emptyset$), successor states are computed for each possible edge using **Succ** operator. Each symbolic transition $T \in Trans$ is of the form $T = ((l, \zeta), \delta, \langle (l_1, \zeta_1), \dots, (l_n, \zeta_n) \rangle)$, where $n = |\text{edges}(l, \delta, \gamma)|$. A symbolic transition T induces probability distribution μ_T over symbolic states Sym . For any $(l', \zeta') \in Sym$: $\mu_T(l', \zeta') = \sum \{ |\text{prob}(l, \delta, \gamma)e_i| \mid e_i \in \text{edges}(l, \delta, \gamma) \wedge \zeta_i = \zeta' \}$.

Using these distributions, the algorithm $\text{BUILDMDP}(Sym, Trans)$ constructs an MDP similarly to that of [1] for PTA, which can then be analysed to compute the reachability probabilities. For PTA, and therefore for PPTA, this approach only gives upper (resp. lower) bounds on maximum (resp. minimum) reachability probability in the model. This is because the reachability graph is too coarse to preserve the precise time when the actions can be taken, and thus constructs an over-approximation of the possible strategies.

Let us highlight the differences between our algorithm and its non-parametric counterpart from [1]. In the non-parametric case, all symbolic states (l, ζ) containing some location $l \in F$ are collected into a set *Reached*. Then, in the constructed MDP, the maximum (or minimum) probability of ending up in *Reached* is calculated. In our setting, we are interested in finding the *optimal* parameter valuations (that maximise or minimise the reachability probability of interest). We thus need to keep separate those symbolic states containing different parameter valuations and calculate the maximum/minimum reachability probability for each one. We divide the set *Reached* into subsets $Reached_i$, each of which contains the symbolic states (l_i, ζ_i) with equivalent constraints on parameters (obtained by projection of the set of valuations ζ_i onto parameters: $\zeta_i|_P$). Another difference arises when building symbolic transitions *Trans*. This follows

PARREACH(\mathcal{P}, F)

```

Sym := ∅; Trans := ∅; Reached := ∅; Waiting := {Init( $\mathcal{P}$ )}; n := 0; Reached0 := ∅
while Waiting ≠ ∅
  choose and remove (l, ζ) from Waiting
  Sym := Sym ∪ {(l, ζ)}
  for δ ∈ Σ such that edges(l, δ, γ) ≠ ∅
    for each ei ∈ edges(l, δ, γ) = ⟨e1, ..., en⟩
      (l'i, ζ'i) := Succ((l, ζ), ei)
      if (l'i, ζ'i) ∉ Sym ∧ ζ'i ≠ ∅ ∧ l'i ∉ F then Waiting := Waiting ∪ {(l'i, ζ'i)}
      else if (l'i, ζ'i) ∉ Sym ∧ ζ'i ≠ ∅ then Reached := Reached ∪ {(l'i, ζ'i)}
    Trans := Trans ∪ {((l, ζ), δ, ⟨(l1, ζ1), ..., (ln, ζn)⟩)}
  // Additional transitions from a state to its subsets
for each (l, ζ) ∈ Sym
  if ∃(l', ζ') ∈ Sym such that l = l' ∧ ζ|X ⊆ ζ'|X ∧ ζ|P ⊆ ζ'|P then
    Trans := Trans ∪ {(l', ζ'), ∅, ⟨(l, ζ)⟩}
  // Divide Reached into subsets Reachedi according to different parameter valuations
for each (l, ζ) ∈ Reached
  if (ζ|P = {Reachedi}|P for some Reachedi where i ∈ [0..n]) then
    Reachedi := Reachedi ∪ {(l, ζ)}
  else Reachedn := Reachedn ∪ {(l, ζ)}; n ++;
return (Sym, Trans)

```

BUILDMDP(Sym, Trans)

```

sym0 = {(l, ζ) ∈ Sym | l = l0}
for (l, ζ) ∈ Sym and T ∈ Trans(l, ζ)
  StepsM((l, ζ), T) := μT
return M = (Sym, sym0, Trans, StepsM)

```

Figure 1: Parametric forward reachability and construction of the corresponding MDP

from the property of TA (and therefore PTA) proven in [12], which states that weakening (resp. strengthening) the guards in any TA \mathcal{T} , e.g decreasing lower and increasing upper (resp. increasing lower and decreasing upper) bounds on clocks, yields an automaton whose reachable states include (resp. are subset of) those of \mathcal{T} . We therefore add, for any two symbolic states $(l_i, \zeta_i), (l_j, \zeta_j) \in \text{Sym}$ which satisfy $\zeta_{i|X} \subseteq \zeta_{j|X} \wedge \zeta_{i|P} \subseteq \zeta_{j|P} \wedge l_i = l_j$, a transition (point distribution) from (l_j, ζ_j) to (l_i, ζ_i) , in order to obtain the correct probabilities in the MDP. By $\{\text{Reached}_i\}_{|P}$ in Figure 1 we denote the parameter values contained in the set Reached_i .

Example 1. Let us consider a PPTA shown in Figure 2, which represents a communication protocol operating over a lossy channel. We have simplified the model from [1] and added a parameter a to the guard of the first transition. The protocol starts with the delivery of new data which resets both clocks x and y (represented as the initial location l_0). The data are retained for at least a time

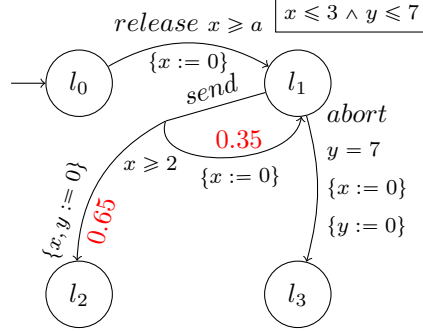


Figure 2: Example of a parametric probabilistic timed automaton

units before being sent to the medium, which resets clock x to zero (represented as a transition between l_0 and l_1 , labelled with *release*). The medium then sends the data to the receiver, and the attempt is successful (location l_2 is reached) with probability 0.65. Otherwise, the medium re-sends the data after a delay varying between 3 and 7 time units. If exactly 7 time units have elapsed since the delivery of the data, the system aborts. We are interested in the values of the parameter a which maximise the probability of the medium successfully *send*-ing the data (which is equivalent to reaching location l_2).

The MDP constructed from the reachability graph is shown in Figure 3. Simplified constraints are shown in the figure, while full constraints, obtained using our prototype implementation, are listed in Appendix A.

There are three symbolic states with *goal* location l_2 and different parameter valuations: $Reached_1 = \{(l_2, x = y \wedge b \leq 5)\}$, $Reached_2 = \{(l_2, x = y \wedge b \leq 3)\}$ and $Reached_3 = \{(l_2, x = y \wedge b \leq 1)\}$. Using the MDP model checking engine in PRISM, we calculated the maximum probabilities of reaching the following states: $p^{max}(\Diamond Reached_1) = 0.65$, $p^{max}(\Diamond Reached_2) = 0.8775$, and $p^{max}(\Diamond Reached_3) = 0.957125$, where $\Diamond\phi$ means that ϕ must hold eventually. If we want to *maximise* the probability of reaching l_2 , it is clear that we should choose parameter values corresponding to s_8 , which is $a \leq 1$. ■

The forward reachability algorithm provides only upper (resp. lower) bounds on the maximum (resp. minimum) reachability probability. In Example 1, this method actually gives the correct values, but consider now the automaton of Figure 4, inspired by [1]. The probability of reaching l_3 obtained using the forward approach (the resulting MDP is shown in Figure 5) is 1, regardless of the value of a . By careful inspection, we observe that the maximum probability is 1 only if $a = 0$ (which results from taking a transition from l_0 at $x = y = 0$, and then, from either l_1 or l_2 , immediately proceeding to l_3). Otherwise this probability is at most 0.5.

The following theorem, a modification from [5] for the parametric setting, establishes that, for PPTA, the reachability graph yields lower bounds on min-

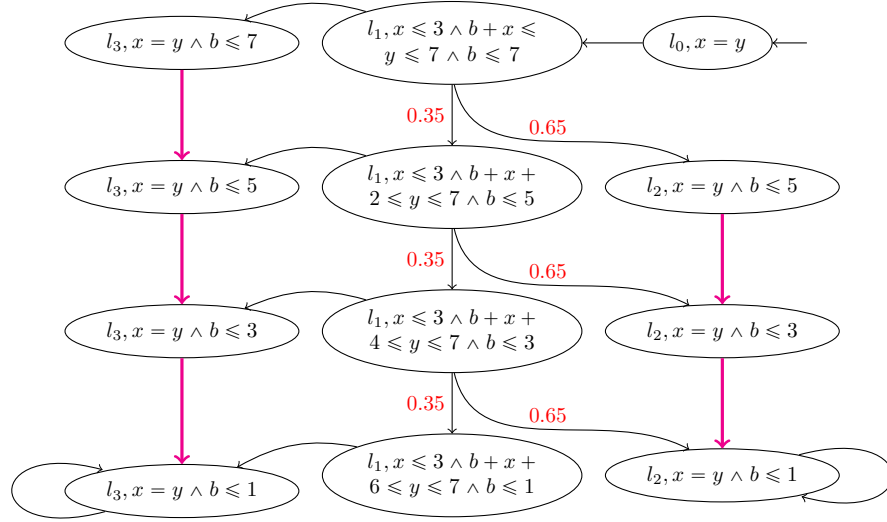


Figure 3: Markov decision process for the PPTA of Figure 2

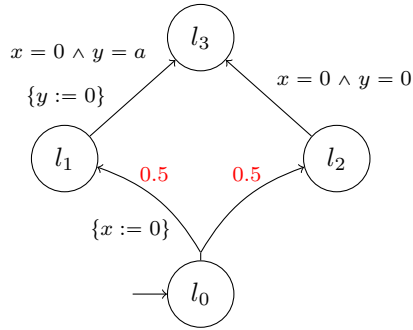


Figure 4: A PPTA

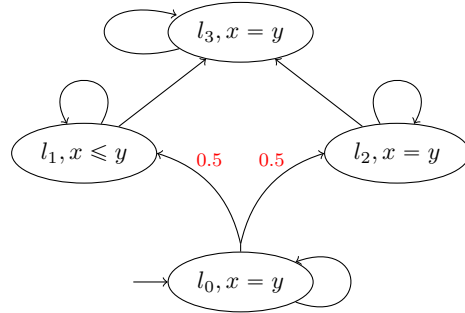


Figure 5: MDP for the PPTA of Figure 4

imum reachability probabilities and upper bounds on maximum reachability probabilities and that the corresponding parameter valuations can be obtained from the symbolic states. In order to prove this theorem, we need to present Lemma 1 and Lemma 2, proven in [5], and Lemma 3 with a proof given in AppendixA.

Let us first introduce the notation needed for proofs. We fix PPTA $\mathcal{P} = (L, l_0, X, P, \Sigma, \text{prob}, \text{Inv})$. $(\text{Sym}, \text{Trans})$ is the reachability graph of \mathcal{P} and $\mathcal{M} = (\text{Sym}, \text{sym}_0, \text{Trans}, \text{Steps}_{\mathcal{M}})$ the MDP obtained with $\text{BUILDMDP}(\text{Sym}, \text{Trans})$. Reached is the set of all symbolic states (l, ζ) from Sym such that $l \in F$. For any parameter valuation v , we obtain a (non-parametric) PTA which we denote $v(\mathcal{P})$. The semantics of $v(\mathcal{P})$ is given by an (infinite state) MDP $\mathcal{M}_{v(\mathcal{P})} = (Q, q_0, \mathbb{R}_{\geq 0} \times \Sigma, \text{Steps}_{\mathcal{M}_{v(\mathcal{P})}})$.

By $v(\text{Sym})$ we denote the set of all reachable (non-parametric) symbolic states obtained using the forward reachability algorithm applied to $v(\mathcal{P})$, and by $v(\mathcal{M})$ the corresponding MDP. In a similar way, we define $v(S) \in v(\text{Sym})$ as $(l, v(\zeta))$, a (non-parametric) symbolic state, and $v(\text{Reached})$ is the set of all symbolic states $(l, v(\zeta))$ such that $l \in F$. For the sake of simplicity, we redefine an edge $e = (R, l')$ in the distribution $\text{prob}(l, \delta, \gamma)$ as $e = (l, \delta, \gamma, R, l')$ and $v(e) = (l, \delta, v(\gamma), R, l')$.

Lemma 1. *For any strategy \mathcal{A} of $\mathcal{M}_{v(\mathcal{P})}$ and $q \in Q$, there exists a strategy $\mathcal{B}_{\mathcal{A}}$ of $v(\mathcal{M})$ where $p_q^{\mathcal{A}}(Q^F) = p_{v(S)}^{\mathcal{B}_{\mathcal{A}}}(v(\text{Reached}))$, for all $v(S) \in v(\text{Sym})$ such that $q \in v(S)$, where Q^F is a set of states $(l, u) \in Q$ such that $l \in F$.*

We also define a run as a finite sequence $\rho = q_1 a_1 q_2 a_2 \dots a_{n-1} q_n$ such that for all $i, q_i \in Q, a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$ and $q_i \xrightarrow{a_i} q_{i+1}$. A sequence of symbolic successor by edges e_1, \dots, e_n : $\text{Succ}(S, e_1, \dots, e_n)$ is defined as $\text{Succ}(\dots \text{Succ}(\text{Succ}((l, \zeta), e_1), e_2) \dots, e_n)$.

Lemma 2. *For any symbolic state (l, ζ) , for all edges e and valuations v , $v(\text{Succ}((l, \zeta), e)) = \text{Succ}((l, v(\zeta)), v(e))$.*

We can safely use Lemma 2 as the Succ operator is not impacted by probabilities. The corollary of Lemma 2 is that for any sequence of edges e_1, \dots, e_n : $v(\text{Succ}(S, e_1, \dots, e_n)) = \text{Succ}(v(S), v(e_1), \dots, v(e_n))$. Let us assume that the set Reached is divided into subsets Reached_i for $i \in [1..n]$. According to our forward reachability algorithm, $\forall (l, \zeta), (l', \zeta') \in \text{Reached}_i$, for any $i \in [1..n]$, $\zeta|_P = \zeta'|_P$. Let us also assume that some $\text{Reached}_k = \{(l_k, \zeta_k)\}, l_k \in F$ has the optimum reachability probability among all $\text{Reached}_i \in \text{Reached}$ in the MDP \mathcal{M} .

Lemma 3. *Let n be a node of T , labeled by some symbolic state S , and such that the subtree rooted at n has depth N . We have that a parameter valuation $v \in \zeta_k|_P$ is the solution to the reachability synthesis problem, starting the forward reachability algorithm from S , iff there exists a state q in $v(S)$ and a run ρ in $v(\mathcal{P})$, with fewer than N discrete steps, that starts in q and reaches l_k .*

Theorem 1. *For a PPTA \mathcal{P} and a subset of its locations F , if $(\text{Sym}, \text{Trans}) = \text{PARREACH}(\mathcal{P}, F)$ and $\mathcal{M} = \text{BUILDMDP}(\text{Sym}, \text{Trans})$, then:*

1. $p_{\mathcal{M}}^{\min}(\text{Reached}) \leq p_{\mathcal{P}}^{\min}(F)$ and $p_{\mathcal{M}}^{\max}(\text{Reached}) \geq p_{\mathcal{P}}^{\max}(F)$;
2. if \mathcal{M} gives the precise reachability probabilities in \mathcal{P} and if some $(l_k, \zeta_k) \in \text{Reached}$ has the optimum (maximum or minimum) reachability probability, among all $(l_j, \zeta_j) \in \text{Reached}$, then $\{\zeta_{k|P}\}$ is the solution to the optimal parameter synthesis problem.

PROOF. We consider the two parts of the theorem separately.

- (1) For $v(\mathcal{P})$, obtained from PPTA \mathcal{P} and any parameter valuation v , the statement is a direct consequence of Lemma 1.
- (2) First we need to prove that $\zeta_{k|P}$ is a solution to the *reachability synthesis problem*, i.e., for any parameter valuation $v \in \zeta_{k|P}$, l_k is reachable in $v(\mathcal{P})$. Let us consider the possibly infinite directed labeled tree, whose root is labeled by $\text{Init}(\mathcal{P})$ and, for every node n , if n is labeled by a symbolic state S , then, for all edges e of the PPTA, there exists a child n' labeled by $\text{Succ}(S, e)$ iff $\text{Succ}(S, e)$ is not empty. For easier reference, we also label the arc from n to n' by e .

Now, consider that the forward reachability algorithm has terminated. Then only a finite prefix (a subset closed under the parent relation) T of the infinite tree has been visited and each leaf must correspond to one of the leaf conditions of the algorithm or to the absence of children. This means that all leaves n of the tree are labeled by symbolic states S such that:

- either $S = (l, \zeta)$ and $l = l_k$;
- or $S \in \text{Sym}$;
- or S has no successor.

With Lemma 3, we immediately have that if $\zeta_{k|P}$ is a solution to the reachability synthesis problem, then there exists a run in $v(\mathcal{P})$ that starts in the initial state and reaches l_k .

In the other direction, suppose there exists such a run ρ . Then ρ is finite and its last state has a location belonging to l_k . Let e_1, \dots, e_p be the edges taken in ρ and consider the branch in the tree T obtained by following this edge sequence on the labels of the arcs in the tree as long as possible. If the whole edge sequence is feasible in T , then the tree T has depth greater or equal to the size of the sequence and we can apply Lemma 1 to obtain that v is the solution to the reachability synthesis problem.

Otherwise, let $S = (l, \zeta)$ be the symbolic state labelling the last node of the branch, e_k be the first edge in e_1, \dots, e_p that is not present in the branch and q be the state of ρ just before taking e_k . Using Lemma 2, $v(\text{Succ}(S, e_k))$ is not empty so $\text{Succ}(S, e_k)$ is not empty. Since the node labeled by S has no children in T , it follows that either $l = l_k$ or there exists another node on the branch that is labeled by S . In the former case, we can apply Lemma 3 to the prefix of ρ ending in q and we obtain that v is the solution to the reachability synthesis problem.

In the latter case, by the corollary of Lemma 2, there exists a run along edges e_1, \dots, e_m , with $m < k$, that reaches q in $v(\mathcal{P})$. From that run we

can construct another run ρ' by merging with the suffix of ρ that starts from q . ρ' has strictly fewer discrete actions than ρ and also reaches l_k and we can repeat the same reasoning as we have just done. We can do this only a finite number of times (because the length of the considered run is strictly decreasing) so at some point we have to be in some of the other cases and we obtain the expected result.

We have proven that $\zeta_{k|P}$ is a solution for the reachability synthesis problem. The same stands for any $\zeta_{j|P}$, such that $(l_j, \zeta_j) \in \text{Reached}$. If we were interested in the reachability synthesis problem, we would take $\bigcup_j \zeta_{j|P}$, for all $(l_j, \zeta_j) \in \text{Reached}$, as a solution. As we are interested in the optimum parameter synthesis problem we must exclude the parameter valuations contained in the other symbolic states of Reached . We therefore take $\{\zeta_{k|P} \setminus (\bigcup_{j \neq k, l_j \in F} \zeta_{j|P})\}$.

□

To resolve the limitation of the forward approach, namely, that it can only compute bounds on the reachability probabilities, in Section 4 we adapt the *game-based abstraction refinement* method from [5] to synthesise the optimal timing parameter values for PPTA. We choose this approach as it can compute precise minimum and maximum probabilities and is shown to perform better than the *digital clocks* approach [27], an alternative model checking technique for PTA.

3.2. Decidability and termination

The *reachability-emptiness* problem for parametric timed automata is undecidable in general [10]. The subclass for which this problem is decidable is L/U automata [12]. For both of those classes, however, the forward reachability exploration is not guaranteed to terminate. This is due to the fact that the classic *k-normalisation* operator [31], which is used to ensure the termination for (non-parametric) timed automata, based on the greatest constant appearing in the model, cannot be used in the parametric case. Since our algorithm for PPTA can be viewed as an extension of forward reachability, termination cannot be guaranteed either. Nevertheless, in order to overcome this problem and ensure both decidability and termination, we could restrict parameter values to bounded integers, as done in [14] for timed automata. This is not a severe restriction, as most tools actually only accept integers in guards and invariants. In the rest of the paper we do not assume this restriction, but, as it is a special case, all the results apply.

4. Synthesis with Game-based Abstraction Refinement

In [32], stochastic two-player games are used as abstractions for MDPs. In such games, the two players represent nondeterminism introduced by the abstraction (player 1) and nondeterminism from the original model (player 2). By

quantifying over all possible strategies for players 1 and 2, we can obtain both the lower bound (*lb*) and the upper bound (*ub*) on either the maximum or minimum reachability probability in the original MDP. If a game \mathcal{G} is constructed from an MDP \mathcal{M} using the approach from [32], where F is a subset of states of \mathcal{M} , we have:

$$p_{\mathcal{G}}^{lb,min}(F) \leq p_{\mathcal{M}}^{min}(F) \leq p_{\mathcal{G}}^{ub,min}(F) \text{ and } p_{\mathcal{G}}^{lb,max}(F) \leq p_{\mathcal{M}}^{max}(F) \leq p_{\mathcal{G}}^{ub,max}(F)$$

where for maximum probabilities:

$$p_{\mathcal{G}}^{lb,max}(F) \stackrel{\text{def}}{=} \sup_{s \in s_0} \inf_{\sigma_1} \sup_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

$$p_{\mathcal{G}}^{ub,max}(F) \stackrel{\text{def}}{=} \sup_{s \in s_0} \sup_{\sigma_1} \sup_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

Using similar techniques to value iteration for MDPs [28], these probabilities can be efficiently approximated, together with the corresponding strategy pairs which achieve them.

In [5], the concept of game-based abstractions is used for PTA in order to compute the maximum and minimum reachability probabilities. The method starts from the MDP obtained via forward reachability algorithm, and subsequently builds and refines the stochastic game abstraction. In this section, we generalise this method by taking into account timing parameters.

4.1. Game-based abstraction for PPTA

The game-based abstraction is constructed by analysing transitions outgoing from each location in a PPTA. The transitions are divided into subsets according to the part of the symbolic state in which they are enabled (sets of transitions with the same source, meaning the same location and subset of ζ). This partition analysis is based on the *validity* operator introduced in [5].

In the non-parametric case, this operator takes the symbolic transition $T = \left((l, \zeta), \delta, \langle (l_1, \zeta_1), \dots, (l_n, \zeta_n) \rangle \right)$ and checks whether the part of symbolic state ζ , from which it is possible to let time pass and then perform action δ , such that taking the i th edge (R_i, l_i) results in reaching some state $(l_i, v) \in (l_i, \zeta_i)$, is empty. Such analysis requires several backward operators, defined for the parametric domain in [33]:

- *past* (time predecessors): $\zeta^{\prec} = \{v' \mid v \in \zeta \wedge v'(x) \geq 0, v'(x) + d = v(x), d \geq 0 \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$
- *inverse reset of clocks* in set $R \subseteq X$: $\zeta[R]^{-1} = \{v' \mid \exists v \in \zeta \text{ s.t. } v'(x) = 0 \text{ if } x \in R \wedge v'(x) = v(x) \text{ otherwise}\}$

We extended the validity operator to the parametric domain. In the non-parametric case, it suffices to know whether it is possible to perform a transition or not. In the parametric setting, however, we want to obtain the valuations on $X \cup P$ from which it is possible to perform the transition T . The operator *valid*(T) gives precisely the set of clock and parameter valuations satisfying ζ

from which it is possible to let time pass and perform an action, such that taking the i th edge gives a state in (l_i, ζ_i) :

$$valid(T) = \zeta \cap \left(\left(\gamma \cap \left(\bigcap_{i=1}^n (\zeta_i[R]^{-1}) \right) \right) \right)^{\swarrow}$$

The transition T is therefore *valid* if the set of valuations (a polyhedron) $valid(T)$ is non-empty. The projection of these valuations onto parameters gives the corresponding values of parameters. In order to construct a stochastic game, the notion of validity is extended to sets of symbolic transitions with the same source:

$$valid(\mathbb{T}) = \left(\bigcap_{T \in \mathbb{T}} valid(T) \right) \cap \left(\bigcap_{T \in Trans(l, \zeta) \setminus \mathbb{T}} \neg valid(T) \right)$$

Here $valid(\mathbb{T})$ defines the set of valuations $v \models \zeta$ on $X \cup P$, such that from (l, v) it is possible to perform any symbolic transition $T \in \mathbb{T}$, but it is not possible to perform any other transition of $Trans(l, \zeta)$. In a symbolic state (l, ζ) of a stochastic game abstraction of a PPTA, player 1 first picks a subset \mathbb{T} of symbolic transitions (in other words, a part of the symbolic state in which these transitions are enabled), and player 2 then picks a transition $T \in \mathbb{T}$. Figure 6 shows the algorithm for the construction of a stochastic game from a given reachability graph. This game, as the following theorem states, yields (by quantifying over all possible strategies for player 1 and player 2) upper and lower bounds on the maximum/minimum reachability probabilities in a PPTA.

Before presenting Theorem 2, we first recall the following lemmas, whose proofs can be found in [5]. $v(S)$, $v(Reached)$ and $v(Sym)$ are defined as in the proof of Theorem 1, and $v(\mathcal{G})$ is the stochastic game obtained from $v(\mathcal{P})$.

Lemma 4. *For any strategy \mathcal{A} of $\mathcal{M}_{v(\mathcal{P})}$ and $q \in Q$, there exists a strategy pair (σ_1, σ_2) of $v(\mathcal{G})$ where $p_q^{\mathcal{A}}(Q^F) = p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached))$ for all $v(S) \in v(Sym)$ such that $q \in v(S)$.*

Lemma 5. *For any symbolic state $v(S) \in v(Sym)$ and player 2 strategy σ_2 of $v(\mathcal{G})$, there exists a strategy \mathcal{A} of $\mathcal{M}_{v(\mathcal{P})}$ where:*

$$\inf_{\sigma_1} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \leq p_q^{\mathcal{A}}(Q^F) \text{ and } \sup_{\sigma_1} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \geq p_q^{\mathcal{A}}(Q^F).$$

Theorem 2. *If $(Sym, Trans) = \text{PARREACH}(\mathcal{P}, F)$, $\mathcal{G} = \text{BUILDGAME}(Sym, Trans)$ and $\ast \in \{min, max\}$ then: $p_{\mathcal{G}}^{lb, \ast}(Reached) \leq p_{\mathcal{P}}^{\ast}(F) \leq p_{\mathcal{G}}^{ub, \ast}(Reached)$.*

PROOF. From Lemma 4 we have:

$$\inf_{\sigma_1, \sigma_2} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \leq \inf_{\mathcal{A}} p_q^{\mathcal{A}}(Q^F)$$

$$\sup_{\sigma_1, \sigma_2} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \geq \sup_{\mathcal{A}} p_q^{\mathcal{A}}(Q^F)$$

for all $v(S) \in v(Sym)$ such that $q \in v(S)$, and therefore:

$$p_{v(\mathcal{P})}^{min}(F) \geq p_{v(\mathcal{G})}^{lb, min}(v(Reached)) \text{ and } p_{v(\mathcal{P})}^{max}(F) \leq p_{v(\mathcal{G})}^{ub, max}(v(Reached))$$

BUILDGAME($Sym, Trans$)

```

 $sym_0 = \{(l, \zeta) \in S \mid l = l_0\}$ 
for  $(l, \zeta) \in S$ 
  for  $\mathbb{T} \subseteq Trans(l, \zeta)$  s.t.  $\mathbb{T} \neq \emptyset$  and  $valid(\mathbb{T}) \neq \emptyset$ 
     $Steps_{\mathcal{G}}((l, \zeta), \mathbb{T}) := \{\mu_T \mid T \in \mathbb{T}\}$ 
return  $\mathcal{G} = (Sym, sym_0, 2^{Trans}, Steps_{\mathcal{G}})$ 

```

Figure 6: Algorithm for stochastic game abstraction

REFINE($Sym, Trans, (l, \zeta), \mathbb{T}_{lb}, \mathbb{T}_{ub}$)

```

 $\zeta_{lb} := valid(\mathbb{T}_{lb}); \zeta_{ub} := valid(\mathbb{T}_{ub})$ 
 $Sym^{new} := \{(l, \zeta_{lb}), (l, \zeta_{ub}), (l, \zeta \wedge \neg(\zeta_{lb} \vee \zeta_{ub}))\} \setminus \{\emptyset\}$ 
 $Sym^{ref} := (Sym \setminus \{(l, \zeta)\}) \uplus S^{new}; Trans^{ref} := \emptyset$ 
for each  $T = (S_0, \delta, \langle S_1, \dots, S_n \rangle) \in Trans$ 
  if  $(l, \zeta) \notin \{S_0, S_1, \dots, S_n\}$  then
     $Trans^{ref} := Trans^{ref} \cup \{T\}$ 
  else  $\mathbb{T}^{new} := \{(S'_0, \delta, \langle S'_1, \dots, S'_n \rangle) \mid S'_i \in Sym^{new} \text{ if } S_i = (l, \zeta) \wedge S'_i = S_i \text{ otherwise}\}$ 
    for  $T^{new} \in \mathbb{T}^{new}$  such that  $valid(T^{new}) \neq \emptyset$ 
       $Trans^{ref} := Trans^{ref} \cup \{T^{new}\}$ 
return  $(Sym^{ref}, Trans^{ref})$ 

```

Figure 7: Algorithm for parametric abstraction refinement

Using Lemma 5, we have that for any $q \in Q$ and $v(S) \in v(Sym)$ such that $q \in v(S)$:

$$\inf_{\mathcal{A}} p_q^A(Q^F) \leq \inf_{\sigma_2} \sup_{\sigma_1} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) = \sup_{\sigma_1} \inf_{\sigma_2} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached))$$

The last equality follows from properties of stochastic games [28]. In a similar way, we have:

$$\sup_{\mathcal{A}} p_q^A(Q^F) \geq \inf_{\sigma_1} \sup_{\sigma_2} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached))$$

It follows that:

$$p_{v(\mathcal{P})}^{min}(F) \leq p_{v(\mathcal{G})}^{ub, min}(v(Reached)) \text{ and } p_{v(\mathcal{P})}^{max}(F) \geq p_{v(\mathcal{G})}^{lb, max}(v(Reached))$$

As the behaviour of \mathcal{P} is described by the behaviour of all $v(\mathcal{P})$, obtained by considering all possible parameter valuations v , we can complete the proof with:

$$p_{\mathcal{P}}^{min}(F) \leq p_{\mathcal{G}}^{ub, min}(Reached) \text{ and } p_{\mathcal{P}}^{max}(F) \geq p_{\mathcal{G}}^{lb, max}(Reached)$$

$$p_{\mathcal{P}}^{min}(F) \geq p_{\mathcal{G}}^{lb, min}(Reached) \text{ and } p_{\mathcal{P}}^{max}(F) \leq p_{\mathcal{G}}^{ub, max}(Reached)$$

□

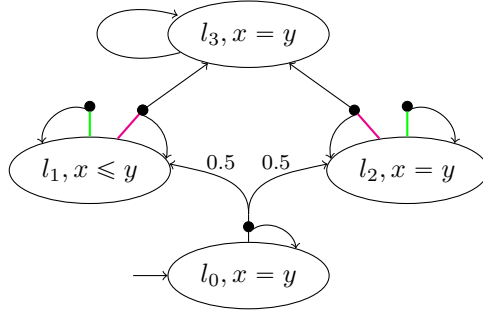


Figure 8: Game-based abstraction

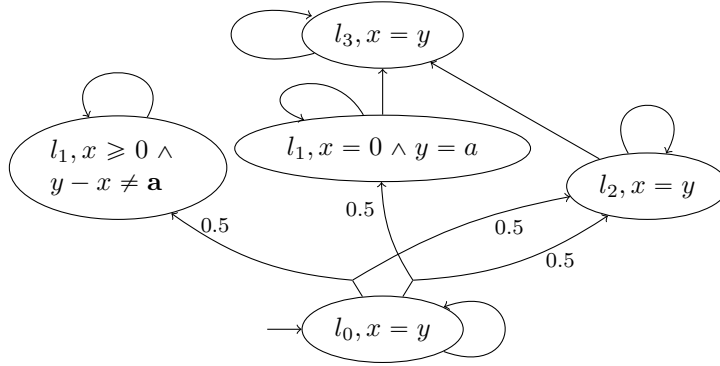


Figure 9: Refinement of a symbolic state ($l_1, x \leq y$)

Example 2. A game constructed from the forward reachability graph of the PPTA in Figure 2 is shown in Figure 8. We represent player 1 states by ellipses containing symbolic states (l, ζ) , and player 2 states by black dots. In two of its states $((l_1, x \leq y)$ and $(l_2, x = y))$, player 1 can choose between the part of the state where both transitions are valid and the part where only one transition is valid (a self-loop). The analysis of this game, however, gives values 0 and 1 for lower and upper bound, respectively, on the maximum probability of reaching l_3 . We address this issue below by applying a method to refine the abstraction. ■

4.2. Parametric abstraction refinement

Stochastic game abstractions might be too imprecise for reachability probabilities, as shown in Example 2. The abstraction refinement method proceeds by iteratively computing refined abstractions of the reachability graph (MDP) until suitable precision is obtained. The game-based abstraction refinement for MDPs from [4] uses the difference between lower and upper bounds on the maximum/minimum reachability probability computed thus far as a quantitative measure of precision. This method has been subsequently generalised in [5] to enable abstraction refinement for PTA. We now explain our extension for the parametric case, which will allow the identification of parameter values corresponding to precise probabilities in the model.

After the construction and analysis of a stochastic game, the refinement algorithm, presented in Figure 7, takes the reachability graph $(Sym, Trans)$, splits one symbolic state per iteration and modifies symbolic transitions accordingly. The split of a symbolic state (l, ζ) is done with respect to player 1 strategy choices in (l, ζ) , T_{ub} and T_{lb} , which achieve lower and upper bounds respectively (such choices must exist in a state where these bounds differ, [32]). The symbolic state (l, ζ) is therefore split into $(l, valid(T_{lb}))$, $(l, valid(T_{ub}))$, and $(l, \zeta \wedge \neg(valid(T_{lb}) \vee valid(T_{ub})))$. By construction, both $valid(T_{lb})$ and $valid(T_{ub})$ are non-empty and $valid(T_{lb}) \neq valid(T_{ub})$, and thus the split produces strict refinement. The MDP of Figure 5, after a refinement of one symbolic state, is shown in Figure 9.

The complete game-based abstraction refinement scheme, shown in Figure 10, provides a method to compute the precise values for maximum/minimum reachability probabilities (unlike the forward approach, which computes only upper/lower bounds), each corresponding to a particular parameter valuation. We can then choose those valuations that optimise (either maximise or minimise) these probabilities. Algorithm SYNTH uses function ANALYZEGAME of [28] to compute bounds on the maximum/minimum probability of reaching some set of locations in a stochastic game and the corresponding strategies. The choice T_i of player 1, in some (l, ζ) , is a set of symbolic transitions T , and also represents the part of ζ in which these transitions are valid (enabled).

To find the parameter valuations corresponding to some $(l_k, \zeta_k) \in Reached$, which has the optimal precise reachability probability in the final MDP, it suf-

fices to compute the projection on the parameters from the target symbolic state $\{\zeta_k|_P\}$.

Theorem 3 states that the refinement algorithm yields a new reachability graph, for which the corresponding stochastic game is a refined abstraction of the PPTA, satisfying the following properties. We need the following lemmas, from [5], modified for the parametric case.

Lemma 6. *If $S^{ref} \in Sym^{ref}$, $(S^{ref}, \delta, \langle S_1^{ref}, \dots, S_n^{ref} \rangle) \in Trans(S^{ref})$ and $S \in Sym$ such that $S^{ref} \subseteq S$, then there exists $(S, \delta, \langle S_1, \dots, S_n \rangle) \in Trans$ such that $S_i^{ref} \subseteq S_i$ for all $1 \leq i \leq n$.*

Lemma 7. *For any strategy pair $(\sigma_1^{ref}, \sigma_2^{ref})$ of $v(\mathcal{G}^{ref})$ and $v(S^{ref}) \in v(Sym^{ref})$ there exists a strategy pair (σ_1, σ_2) of $v(\mathcal{G})$ where:*

$$p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) = p_{v(S^{ref})}^{\sigma_1^{ref}, \sigma_2^{ref}}(v(Reached))$$

for all $v(S) \in v(Sym)$ such that $v(S^{ref}) \subseteq v(S)$.

Lemma 8. *For any $v(S) \in v(Sym)$ and player 2 strategy σ_2 of $v(\mathcal{G})$ there exist a strategy pair $(\sigma_1^{ref}, \sigma_2^{ref})$ of $v(\mathcal{G}^{ref})$ where:*

$$\inf_{\sigma_1} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \leq p_{v(S^{ref})}^{\sigma_1^{ref}, \sigma_2^{ref}}(v(Reached))$$

$$\sup_{\sigma_1} p_{v(S)}^{\sigma_1, \sigma_2}(v(Reached)) \geq p_{v(S^{ref})}^{\sigma_1^{ref}, \sigma_2^{ref}}(v(Reached))$$

for all $v(S^{ref})$ such that $v(S^{ref}) \subseteq v(S)$.

Theorem 3. *For a PPTA \mathcal{P} , a subset of its location F and $\ast \in \{\min, \max\}$, let $(Sym, Trans) = \text{PARREACH}(\mathcal{P}, F)$. If $(Sym^{ref}, Trans^{ref})$ is the result returned by applying REFINE to $(Sym, Trans)$, \mathcal{G} by BUILDGAME($Sym, Trans$) and \mathcal{G}^{ref} by BUILDGAME($Sym^{ref}, Trans^{ref}$) then:*

1. $(Sym^{ref}, Trans^{ref})$ is a reachability graph for (\mathcal{P}, F) ;
2. $p_{\mathcal{G}}^{lb, \ast}(Reached) \leq p_{\mathcal{G}^{ref}}^{lb, \ast}(Reached)$ and $p_{\mathcal{G}}^{ub, \ast}(Reached) \geq p_{\mathcal{G}^{ref}}^{ub, \ast}(Reached)$;
3. If $p^{\ast} = p_{\mathcal{G}^{ref}}^{lb, \ast}(l_k, \zeta_k) = p_{\mathcal{G}^{ref}}^{ub, \ast}(l_k, \zeta_k)$, for some $(l_k, \zeta_k) \in Reached$, is the optimum \ast reachability probability, among all $(l_j, \zeta_j) \in Reached$, then the solution to the optimal parameter synthesis can be extracted from the strategy σ_1 of Player 1, (that corresponds to p^{\ast}), and ζ_k .

PROOF. We consider the three parts of the theorem separately.

- (1) Let us consider any $S^{ref} \in Sym^{ref}$, $(S^{ref}, \delta, \langle S_1^{ref}, \dots, S_n^{ref} \rangle) \in Trans(S^{ref})$ and $S \in Sym$ such that $S^{ref} \subseteq S$. The proof is split into two cases:

- If $S^{ref} \in Sym$, then by construction $S^{ref} = S$, and therefore we have that either $(S^{ref}, \delta, \langle S_1^{ref}, \dots, S_n^{ref} \rangle) \in Trans(S)$, in which case Lemma 6 holds, or there exists $(S, \delta, \langle S_1, \dots, S_n \rangle) \in Trans(S)$ from which $(S^{ref}, \delta, \langle S_1^{ref}, \dots, S_n^{ref} \rangle)$ was constructed. In the second case, it follows from REFINE in Figure 7, that $S_i^{ref} \in S_i$, for all $1 \leq i \leq n$, as required.
- If $S^{ref} \notin Sym$, then for $S^{ref} \subseteq S$ it follows that S^{ref} was formed by splitting S . Thus, there exists a symbolic transition $(S, \delta, \langle S_1, \dots, S_n \rangle) \in Trans$ which was used to construct $(S^{ref}, \delta, \langle S_1^{ref}, \dots, S_n^{ref} \rangle)$. It follows from this construction that $S_i^{ref} \in S_i$ for all $1 \leq i \leq n$, as required.

The fact that $(Sym^{ref}, Trans^{ref})$ is the reachability graph for \mathcal{P} follows from the fact that we split only symbolic states and remove transitions which are not valid.

- (2) We consider $v(\mathcal{P})$ for any parameter valuation v . $v(Sym)$, $v(S)$, $v(Reached)$ and $v(\mathcal{G})$ are defined as in proofs of Theorem 1 and Theorem 2. The proof of this statement then follows similarly to the proof of Theorem 2, using Lemma 7 and Lemma 8 instead of Lemma 4 and Lemma 5. We refer to [5] for the complete proof.
- (3) Combining Theorem 1 and Theorem 3(1), we obtain that $\zeta_{k|P}$ is a solution to the reachability synthesis problem.

Upon the termination of the iterative procedure of the algorithm SYNTH, we are sure that no state $(l, \zeta) \in Reached$ has been divided, therefore parameter valuations $\zeta|_P$ of any $(l, \zeta) \in Reached$ stay unchanged. On the other hand, some of the symbolic states from Sym are divided and therefore their parameter valuations are refined. Optimum strategy σ_1 of player 1 (a strategy to reach (l_k, ζ_k) which has the optimum reachability probability) consists of choices \mathbb{T} in symbolic states of the final set of symbolic states Sym^{ref} .

For each such \mathbb{T} , $valid(\mathbb{T})$ gives the set of valuations on $X \cup P$, in which \mathbb{T} is valid. $valid(\mathbb{T})|_P$ then gives the corresponding parameter values. To allow all choices \mathbb{T}_i in the optimum strategy σ_1 , we need to take $\bigcap_i valid(\mathbb{T}_i)|_P$. Finally, the optimal solution is obtained as $\{\bigcap_i valid(\mathbb{T}_i)|_P \cap (\zeta_{k|P} \setminus (\bigcup_{\forall j \neq k, l_j \in F} \zeta_{j|P}))\}$.

□

The algorithm is designed to terminate when the difference between the upper and lower bound falls below some threshold ϵ for reasons of computational efficiency. We show that this is, however, not necessary. If the initial forward reachability exploration terminates (PARREACH), then our algorithm, similarly to its non-parametric counterpart from [5], is guaranteed to terminate in a finite number of steps with a precise answer.

Theorem 4 (Termination). *Let $\ast \in \{min, max\}$. If the forward reachability algorithm (PARREACH) terminates, then the algorithm for parameter synthesis SYNTH terminates after a finite number of steps and returns $p^\ast = p^{lb, \ast} = p^{ub, \ast}$.*

SYNTH($\mathcal{P}, F, *, \varepsilon, \star$)

```

(Sym, Trans) = PARREACH( $\mathcal{P}, F$ );  $\mathcal{G}$  = BUILDGAME( $Sym, Trans$ );  $p^* := 0$ ;  $\sigma_{p^*} := \emptyset$ 
for each  $Reached_i \in Reached$ 
   $(p_G^{lb,*}, p_G^{ub,*}, \sigma_1^{lb}, \sigma_1^{ub}) := \text{ANALYSEGAME}(\mathcal{G}, Reached_i, *)$ 
  while  $p_G^{ub,*} - p_G^{lb,*} > \varepsilon$ 
    choose  $(l, \zeta) \in Sym$ 
     $(Sym^{ref}, Trans^{ref}) = \text{REFINE}(Sym, Trans, (l, \zeta), \sigma_1^{lb}(l, \zeta), \sigma_1^{ub}(l, \zeta))$ 
     $\mathcal{G} = \text{BUILDGAME}(Sym^{ref}, Trans^{ref})$ 
     $(p_G^{lb,*}, p_G^{ub,*}, \sigma_1^{lb}, \sigma_1^{ub}) := \text{ANALYSEGAME}(\mathcal{G}, Reached_i, *)$ 
  if  $p^* \sim p_G^{lb,*}$  then // put  $<$  (resp.  $>$ ) instead of  $\sim$  when  $\star$  is maximisation
     $p^* := p_G^{lb,*}$ ;  $\sigma_{p^*} := \sigma_1^{lb}$  (resp. minimisation)
return  $[p^*, \sigma_{p^*}]$ 

```

Figure 10: Parameter synthesis using game-based abstraction refinement loop

PROOF. As stated in Section 3, for stochastic game $\mathcal{G} = (S, (S_1, S_2), s_0, \Sigma, Steps_{\mathcal{G}})$ and maximum reachability probabilities, for some $F \subseteq S$ we have:

$$p_G^{lb,max}(F) \stackrel{\text{def}}{=} \sup_{s \in s_0} \inf_{\sigma_1} \sup_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

$$p_G^{ub,max}(F) \stackrel{\text{def}}{=} \sup_{s \in s_0} \sup_{\sigma_1} \sup_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

For the minimum reachability probabilities, similar equations hold:

$$p_G^{lb,min}(F) \stackrel{\text{def}}{=} \inf_{s \in s_0} \inf_{\sigma_1} \inf_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

$$p_G^{ub,min}(F) \stackrel{\text{def}}{=} \inf_{s \in s_0} \sup_{\sigma_1} \inf_{\sigma_2} p_s^{\sigma_1, \sigma_2}(F)$$

At each iteration, a split of a symbolic state provides a strict refinement. At the point where each symbolic states (l, ζ) in a game \mathcal{G}^{ref} has only one outgoing subset of edges \mathbb{T} (thus, only one possible choice for player 1), we have $p_{\mathcal{G}^{ref}}^{lb,*}(Reached) = p_{\mathcal{G}^{ref}}^{ub,*}(Reached)$, because the upper and the lower bounds, for either maximum or minimum reachability probability, are obtained for the different strategy choice of player 1 (player 2 plays either its best or its worst choice, depending on whether we are considering maximum or minimum reachability probabilities, respectively). The algorithm then terminates with the precise answer, as, according to Theorem 3, $p_{\mathcal{G}^{ref}}^{lb,*}(Reached) \leq p_{\mathcal{P}}^*(F) \leq p_{\mathcal{G}^{ref}}^{ub,*}(Reached)$. \square

Example 3. We return to the PPTA from Figure 4. The final stochastic game, shown in Figure 11, after two refinement iterations contains six symbolic states. The validity of each new symbolic transition T_i , obtained in the refinement process, gives the following parameter valuations:

- $T_1 = ((l_0, x = y), \emptyset, \langle (l_1, x = 0 \wedge y = a), (l_2, x = y = 0) \rangle) \neq \emptyset$ if $a = 0$

- $T_2 = ((l_0, x = y), \emptyset, \langle (l_1, x = 0 \wedge y = a), (l_2, x = y > 0) \rangle) \neq \emptyset$ if $a \neq 0$
- $T_3 = ((l_0, x = y), \emptyset, \langle (l_1, x \geq 0 \wedge y \neq a), (l_2, x = y = 0) \rangle) \neq \emptyset$ if $a \neq 0$
- $T_4 = ((l_0, x = y), \emptyset, \langle (l_1, x \geq 0 \wedge y \neq a), (l_2, x = y > 0) \rangle) \neq \emptyset$ for $a \in \mathbb{R}_{\geq 0}$.

The set of transitions $\mathbb{T}_1 = \{T_2, T_3, T_4\}$ is valid if $a \neq 0$, in which case the maximum probability of reaching l_3 is 0.5, and $\mathbb{T}_2 = \{T_1, T_4\}$ is valid if $a = 0$, in which case the maximum probability of reaching l_3 is 1. If we wish to, for example, maximise this probability, the algorithm obtains the constraint $a = 0$.

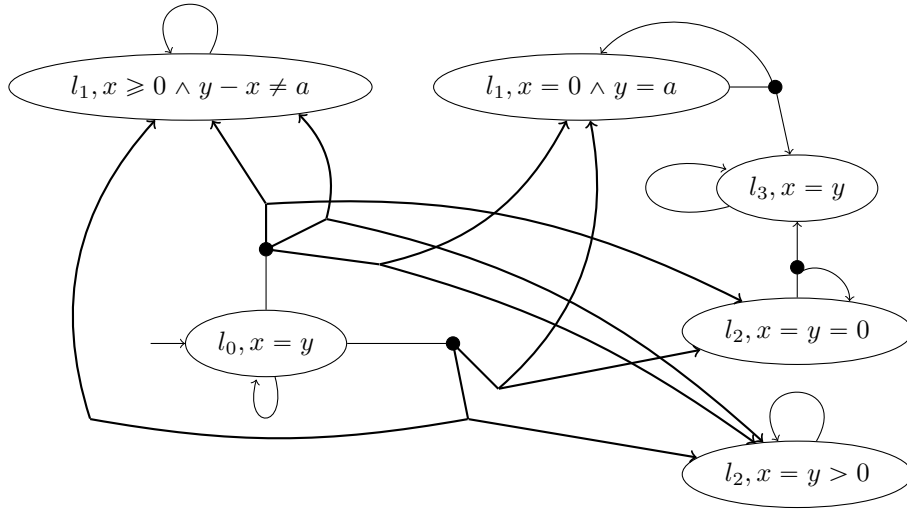


Figure 11: Final stochastic game for PPTA of Figure 4 (probability 0.5 is left out for each edge of probabilistic (thick line) transitions outgoing from l_0)

■

5. Implementation

In this section we report on our prototype implementation of a parametric extension for PTA model checking in PRISM, which is invoked through the command line interface. PTA models can be specified in PRISM's modelling language, where we allow the use of parameters (with possible initial conditions on their values) in guards and invariants in the definition of a PTA. The output is provided in the form of constraints on parameters such that the precise probability of reaching a certain location in the model is maximised or minimised.

A PTA model in PRISM's language is supplied as a composition of reactive modules. For the PPTA in Fig. 2, the model in PRISM is shown below, where the model type is indicated by keyword **pta**:

```

pta
const int a;
module M

s : [0..3] init 0;
x : clock;
y : clock;

invariant
(s=1 => x<=3 & y<=7)
endinvariant

[release] s=0 & x>=a->(s'=1)&(x'=0);
[send] s=1 & x>=2->0.65:(s'=2)&(x'=0)&(y'=0)+0.35:(s'=1)&(x'=0);
[abort] s=1 & y>=7->(s'=3)&(x'=0)&(y'=0);

endmodule

```

The first three lines of declarations following the keyword **module** specify the variables of the model, where s represents the locations of a PPTA and x, y represent clocks. The following three lines specify the invariants and the last three lines define the transitions. To model PPTA, we allow the possibility that the initial values of constants remain unspecified (**const int a** with no **init**), which serves as a means to specify the parameters. This is indicated by listing the parameters (unspecified constants) as the last argument of the command line. The output is produced as a reachability graph (set of parametric symbolic states and transitions) obtained in the first phase of the algorithm before the refinement, together with the probability value (maximum or minimum) and the corresponding parameter valuations as a set of symbolic constraints on parameters.

Our prototype implementation is integrated within the current support for PTA in PRISM that employs the stochastic games engine. We give below a brief description of the main data structures and algorithms that were extended.

5.1. Data structures and algorithms for parametric constraints

A DBM (difference bound matrix) [31] is a data structure used to store and manipulate zones used in symbolic verification of TA and PTA in PRISM, as well as in tools such as UPPAAL and KRONOS. It enables the necessary operations to be performed efficiently, as they are based on the comparison of entries in the matrices. A DBM is an $(n+1) \times (n+1)$ matrix representing a zone, where n is a number of clocks in the model $\{x_1, \dots, x_n\}$, and x_0 is an additional clock with constant value 0. It encodes the differences between each two clocks in an element of the matrix. The row is used for storing lower bounds on the

difference between the clock and all other clocks (the corresponding column is used for upper bounds). DBM D of zone Z is computed in the following steps:

- $d(i, j) = (c, \sim)$ for bound $x_i - x_j \sim c$ in Z
- $d(i, j) = \infty$ if the clock difference $x_i - x_j$ is unbounded in Z
- $d(i, i) = (0, \leq)$ for each clock
- $d(0, i) = (0, \leq)$ implicit constraint that all the clocks are positive.

A PDBM is a parametric extension of DBM, introduced in [12], which we implemented to provide support for PPTA. A PDBM consists of a matrix, where each entry is a linear expression coming from the corresponding guard and a set of constraints on parameters (a polyhedron over its valuations). The following operations on PDBMs are required for the forwards reachability graph construction, described more in detail in [12], where to manipulate sets of valuations on parameters (polyhedra) we used the Parma Polyhedra Library [34]:

- **adding constraints** The operation of adding a constraint in a DBM is done by taking the pointwise minimum of the entries of a DBM and a guard, which is also viewed as a DBM. In the parametric case, we compare linear expressions to find the minimum. In general, the result might be ambiguous. In this case, a zone split is performed and two PDBMs are returned, each with an updated set of constraints on parameters.
- **canonicalisation** This algorithm compares the difference between two clocks to the difference obtained when an intermediate clock is taken into account. For PDBM, as in the case of adding a constraint, we compare two linear expressions, and therefore the operation generally results in a set of new PDBMs rather than a single PDBM. The cost of canonicalisation might be $\mathcal{O}(2^{n^3})$ and the operation is obtained as a modification of the Floyd-Warshall algorithm for computing all-pairs shortest paths [35].
- **reset** As in the case of DBM, the row and column for the reset clock is replaced with the row and column for the clock x_0 , except for the diagonal element $d(i, i)$ which stays equal to 0. The reset operation preserves the canonical form of a PDBM.
- **future (time successors)** Removing upper bounds on clocks, as for DBM, corresponds to setting the bounds in the first column to $(\infty, <)$ for each $i \neq 0$. This operation preserves the canonical form.

Additional operations required for the abstraction refinement of the reachability graph are as follows:

- **past (time predecessors)** Time predecessors are computed by replacing bounds in the first row by $(0, \leq)$. The PDBM obtained after this operation must be canonicalised.

- **inverse reset of clocks** This operation requires intersection, canonicalisation and, finally, replacing the bounds in the row of inverse reset clock with $(\infty, <)$.
- **intersection** Intersection of two PDBMs D_1 and D_2 computes the minimum between $d_1(i, j)$ and $d_2(i, j)$ for every i, j . For each comparison one or two PDBMs may be obtained. Thus, the result of the intersection will be a set of PDBMs.

5.2. Case study: The Root Contention Protocol (IEEE 1394 FireWire)

We consider the abstract probabilistic timed automaton model for the IEEE 1394 FireWire root contention protocol given in [3], also available with the PRISM distribution. This protocol concerns the election of a leader between two contending nodes of a network. The protocol consists of a number of rounds in which each of the contending nodes flips a coin; given the result of the coin flip, a node may decide to wait for a short (*rc_fast_min* and *rc_fast_max* constants) or a long (*rc_slow_min* and *rc_slow_max* constants) amount of time. After this amount of time has elapsed, a node then checks to see if the other node has already deferred, and declares itself to be the leader if this is the case; otherwise, this node defers. Intuitively, in the case in which the result of the two nodes' coin flips are different, the 'faster' node defers to the 'slower' node, the latter of which then becomes leader, signalling the end of protocol execution. However, if the results of the coin flips are the same, the communication delay (*delay* constant) between the two nodes means that it is possible that both nodes attempt to defer to the other, requiring another round of the protocol. Each decision of a node to wait for a short (resp. long) period of time is modelled as a probabilistic transition between the maximum or minimum short (resp. long) period of time to wait.

We parameterised the model from [3] by replacing constants used in guards and invariants in the initial PTA, respectively representing the communication delay *delay* = 360, the minimum and maximum short period of time to wait *rc_fast_min* = 760 and *rc_fast_max* = 850, and the minimum and maximum long period of time to wait *rc_slow_min* = 1590 and *rc_slow_max* = 1670, by six parameters *a, b, c, d* and *e*. We added the initial constraint that each parameter must be greater than 0, as it would be unrealistic to expect a realisable non-zero delay or time to wait.

We ran the algorithm for the deadline property, i.e., the maximum probability of reaching the location representing a state of the system where the leader is elected ($l = 9$ in the model from [3]), written in PRISM as $P_{max} = ?[F \ l = 9]$. The reachability graph gives 52 parametric symbolic states, while the non-parametric model checking for the initial PTA gives 10 symbolic states using the fixed values for parameters given above. This is due to the splitting of parametric symbolic states as a result of comparing linear expressions during intersection with guards, canonicalisation, inverse reset, etc.

The algorithm returns 1 as the probability under the constraints $c - b \geq 0$ and $e - d \geq 0$, which indicates that the minimum short time to wait must be

less than the maximum short time to wait ($c - b \geq 0$) and that the minimum long time to wait must be less than the maximum long time to wait ($e - d \geq 0$). It is interesting to note that the communication delay can be arbitrarily large and does not influence the probability of the leader election in the protocol.

6. Conclusion

We presented a technique for PPTA which derives constraints on parameters of the model, such that the maximum/minimum probability of reaching some set of locations is optimised. We focused on probabilistic reachability, but we can easily consider more expressive target sets that refer to locations and clocks by syntactically modifying the model as in [1]. Termination of our algorithm depends on whether the forward reachability exploration terminates. Unlike for TA/PTA, where the k -normalisation operator on zones can be used to ensure termination, in the parametric case we need to impose restrictions on parameters, as explained in Section 3.

As future work we plan to optimise our implementation and extend it for more involved properties. This is straightforward for PTCTL based on the reduction of [1] to probabilistic reachability, but expected time/reward properties are more challenging. In [36, 37], algorithms are provided to compute minimum and maximum expected time as value iteration over the backwards zone graph of a PTA. It is also shown that zones are not sufficient and convex polyhedra are required. There is, therefore, little hope that the stochastic games abstraction could be employed for expected time/reward properties for PTA.

Acknowledgments This research is supported by ERC AdG VERIWARE.

References

- [1] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston, Automatic verification of real-time systems with discrete probability distributions, TCS 282 (2002) 101–150.
- [2] R. Alur, D. L. Dill, A theory of timed automata, TCS 126 (1994) 183–235.
- [3] M. Kwiatkowska, G. Norman, J. Sproston, F. Wang, Symbolic model checking for probabilistic timed automata, Information and Computation 205 (7) (2007) 1027–1077.
- [4] M. Kattenbelt, M. Kwiatkowska, G. Norman, D. Parker, A game-based abstraction-refinement framework for Markov decision processes, FMSD 36 (3) (2010) 246–280.
- [5] M. Kwiatkowska, G. Norman, D. Parker, Stochastic games for verification of probabilistic timed automata, in: FORMATS’09, Vol. 5813 of LNCS, Springer, 2009, pp. 212–227.

- [6] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: CAV'11, Vol. 6806 of LNCS, Springer, 2011, pp. 585–591.
- [7] A. Jovanović, M. Kwiatkowska, Parameter synthesis for probabilistic timed automata using stochastic game abstractions, in: RP 2014, LNCS, Springer, 2014.
- [8] E. M. Hahn, H. Hermanns, L. Zhang, Probabilistic reachability for parametric markov models, in: SPIN, 2009, pp. 88–106.
- [9] T. Chen, E. M. Hahn, T. Han, M. Kwiatkowska, H. Qu, L. Zhang, Model repair for markov decision processes, in: Proc. 7th International Symposium on Theoretical Aspects of Software Engineering (TASE), IEEE CS Press, 2013, pp. 85–92.
- [10] R. Alur, T. A. Henzinger, M. Y. Vardi, Parametric real-time reasoning, in: STOC'93, ACM Press, 1993, pp. 592–601.
- [11] L. Doyen, Robust parametric reachability for timed automata., Information Processing Letters 102 (5) (2007) 208–213.
- [12] T. Hune, J. Romijn, M. Stoelinga, F. W. Vaandrager, Linear parametric model checking of timed automata, Journal of Logic and Algebraic Programming 52-53 (2002) 183–220.
- [13] L. Bozzelli, S. L. Torre, Decision problems for lower/upper bound parametric timed automata, FMSD 35 (2) (2009) 121–151.
- [14] A. Jovanović, D. Lime, O. H. Roux, Integer parameter synthesis for timed automata, in: TACAS 2013, Vol. 7795 of LNCS, Springer, 2013, pp. 401–415.
- [15] M. Diciolla, C. H. P. Kim, M. Kwiatkowska, A. Mereacre, Synthesising optimal timing delays for timed i/o automata, in: EMSOFT'14, ACM, 2014.
- [16] M. Ceska, P. Pilar, N. Paoletti, L. Brim, M. Kwiatkowska, Prism-psy: Precise gpu-accelerated parameter synthesis for stochastic systems, in: 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), to appear, LNCS, Springer, 2016.
- [17] F. Wang, Parametric timing analysis for real-time systems, Information and Computation 130 (2) (1996) 131 – 150.
- [18] V. Bruyère, J.-F. Raskin, Real-time model-checking: Parameters everywhere, Logical Methods in Computer Science 3 (1).
- [19] M. Knapik, W. Penczek, Transactions on Petri Nets and Other Models of Concurrency V, Springer Berlin Heidelberg, 2012, Ch. Bounded Model Checking for Parametric Timed Automata, pp. 141–159.

- [20] L.-M. Traonouez, A parametric counterexample refinement approach for robust timed specifications, in: S. S. Bauer, J.-B. Raclet (Eds.), FIT, Vol. 87 of EPTCS, 2012, pp. 17–33.
- [21] É. André, L. Fribourg, Behavioral cartography of timed automata, in: A. Kučera, I. Potapov (Eds.), Proceedings of the 4th Workshop on Reachability Problems in Computational Models (RP’10), Vol. 6227 of Lecture Notes in Computer Science, Springer, 2010, pp. 76–90.
- [22] T. Han, J.-P. Katoen, A. Mereacre, Approximate parameter synthesis for probabilistic time-bounded reachability, in: RTSS, IEEE Computer Society, 2008, pp. 173–182.
- [23] M. Kwiatkowska, A. Mereacre, N. Paoletti, A. Patanè, Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques, in: Proceedings of the 4th International Workshop on Hybrid Systems and Biology (HSB 2015), Vol. 9271 of LNCS/LNBI, Springer, 2015, pp. 119–140.
- [24] É. André, L. Fribourg, J. Sproston, An extension of the inverse method to probabilistic timed automata, FMSD 42 (2013) 119–145.
- [25] N. Chamseddine, M. Duflo, L. Fribourg, C. Picaronny, J. Sproston, Computing expected absorption times for parametric determinate probabilistic timed automata, in: QEST’08, IEEE CS Press, 2008, pp. 254–263.
- [26] M. Kwiatkowska, G. Norman, J. Sproston, Symbolic computation of maximal probabilistic reachability, in: K. Larsen, M. Nielsen (Eds.), Proc. 13th International Conference on Concurrency Theory (CONCUR’01), Vol. 2154 of LNCS, Springer, 2001, pp. 169–183.
- [27] M. Kwiatkowska, G. Norman, D. Parker, J. Sproston, Performance analysis of probabilistic timed automata using digital clocks, FMSD 29 (2006) 33–78.
- [28] A. Condon, The complexity of stochastic games, Information and Computation 96 (1992) 203–224.
- [29] K. G. Larsen, P. Pettersson, W. Yi, UPPAAL in a Nutshell, Int. Journal on Software Tools for Technology Transfer 1 (1997) 134–152.
- [30] C. Daws, A. Olivero, S. Tripakis, S. Yovine, The tool kronos, in: In Proc. of Hybrid Systems III, LNCS 1066, Springer Verlag, 1996, pp. 208–219.
- [31] J. Bengtsson, W. Yi, Timed automata: Semantics, algorithms and tools, in: In Lecture Notes on Concurrency and Petri Nets, Springer-Verlag, 2004, pp. 87–124.
- [32] M. Kwiatkowska, G. Norman, D. Parker, Game-based abstraction for Markov decision processes, in: QEST’06, IEEE CS Press, 2006, pp. 157–166.

- [33] A. Jovanović, D. Lime, O. H. Roux, Synthesis of bounded integer parameters for parametric timed reachability games, in: ATVA 2013, Vol. 8172 of LNCS, Springer, 2013, pp. 87–101.
- [34] R. Bagnara, P. M. Hill, E. Zaffanella, The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems, *Sci. Comput. Program.* 72 (1-2) (2008) 3–21.
- [35] T. H. Cormen, C. Stein, R. L. Rivest, C. E. Leiserson, *Introduction to Algorithms*, 2nd Edition, McGraw-Hill Higher Education, 2001.
- [36] A. Jovanovic, M. Kwiatkowska, G. Norman, Symbolic minimum expected time controller synthesis for probabilistic timed automata, in: *Proc. 13th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, to appear, LNCS, Springer, 2015.
- [37] A. Jovanovic, M. Kwiatkowska, G. Norman, Q. Peyras, Symbolic optimal expected time reachability computation and controller synthesis for probabilistic timed automata, *Theor. Comput. Sci.* 669 (2017) 1–21.

Appendix A.

Example 1.

We give here the symbolic states for the MDP of Figure 3. A symbolic state consist of a location and a (list of) PDBM(s). Each PDBM obtained with our prototype implementation (given in curly brackets) consists of a matrix and a polyhedron over parameters. A matrix is given in square brackets and each element of the matrix is represented in the following form $\sim c$, where $\sim \in \{<, \leq\}$ and c is a linear expression over parameters. Different rows are separated by a comma. In this example, matrices have dimension 3×3 . The corresponding constraints on parameters are given after the closing square bracket.

- $S_0 = (l_0, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a > -\infty, a > 0\})$
- $S_1 = (l_1, \{[<= 0 <= 0 <= -a, <= 3 <= 0 <= -a, <= 7 <= 7 <= 0] - a >= -7, a > 4\}, \{[<= 0 <= 0 <= -a, <= 3 <= 0 <= -a, <= 7 <= 7 <= 0] - a >= -4, a > 0\})$
- $S_2 = (l_2, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -5, a > 4\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -4, a > 0\})$
- $S_3 = (l_1, \{[<= 0 <= 0 <= -a - 2, <= 3 <= 0 <= -a - 2, <= 7 <= 7 <= 0] - a >= -5, a > 4\}, \{[<= 0 <= 0 <= -a - 2, <= 3 <= 0 <= -a - 2, <= 7 <= 7 <= 0] - a >= -4, a > 2\}, \{[<= 0 <= 0 <= -a - 2, <= 3 <= 0 <= -a - 2, <= 7 <= 7 <= 0] - a >= -2, a > 0\})$
- $S_4 = (l_3, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] a = 7\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a > -4, a > 0\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] a = 4\})$
- $S_5 = (l_2, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -3, a > 2\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -2, a > 0\})$
- $S_6 = (l_1, \{[<= 0 <= 0 <= -a - 4, <= 3 <= 0 <= -a - 4, <= 7 <= 7 <= 0] - a >= -3, a > 2\}, \{[<= 0 <= 0 <= -a - 4, <= 3 <= 0 <= -a - 4, <= 7 <= 7 <= 0] - a >= -2, a > 0\})$
- $S_7 = (l_3, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] a = 5\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -4, a > 2\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a > -2, a > 0\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] a = 2\})$
- $S_8 = (l_2, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -1, a > 0\})$
- $S_9 = (l_1, \{[<= 0 <= 0 <= -a - 6, <= 3 <= 0 <= -a - 6, <= 7 <= 7 <= 0] - a >= -1, a > 0\})$
- $S_{10} = (l_3, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] a = 3\}, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0] - a >= -2, a > 0\})$

- $S_{11} = (l_3, \{[<= 0 <= 0 <= 0, < \infty <= 0 <= 0, < \infty <= 0 <= 0]a = 1\})$

■

Lemma 3. *Let n be a node of T , labeled by some symbolic state S , and such that the subtree rooted at n has depth N . We have that a parameter valuation $v \in \zeta_{k|P}$ is the solution to the reachability synthesis problem, starting the forward reachability algorithm from S , iff there exists a state q in $v(S)$ and a run ρ in $v(\mathcal{P})$, with fewer than N discrete steps, that starts in q and reaches l_k .*

PROOF. We prove this by induction on N . Note that the tree T is always non-empty (it contains at least the root which is labeled by $Init(\mathcal{P})$).

- Case of a leaf n labeled by S : the subtree rooted at n has depth 1.
 - If v is the solution to the reachability synthesis problem then the only leaf condition of the algorithm that can be verified is $S = (l_k, \zeta_k)$, so, for all states $q = (l_k, u)$ in $v(S)$, there is a run with no discrete steps that starts in q and reaches l_k .
 - If there exists a state $q \in v(S)$ and a run with no discrete steps that starts in q and reaches l_k , then if l is the location of q , we have $l = l_k$, and therefore v is the solution to the reachability synthesis problem.
- Case of a non-leaf node n labeled by S : suppose the subtree rooted at n has depth $k > 1$ and that, for all nodes n' with subtree rooted at n' of depth $k' < k$, the property holds.
 - If v is the solution to the reachability synthesis problem then, since n is not a leaf, the following condition must be true:
 There exists a successor n' of n , labeled by $S' = \text{Succ}(S, e)$ for some edge e such that v is the solution to the reachability synthesis problem, starting the forward reachability algorithm from S' . Since it is a successor of n , n' has depth less than k . So we can use the induction hypothesis: there exists a run with less than $k - 1$ discrete steps, starting in some state $q' \in v(S')$ and reaching l_k in $v(\mathcal{P})$. By Lemma 2, $q' \in \text{Succ}(v(S), v(e))$ so q' has a predecessor q by e in $v(S)$ and we get the expected result.
 - If there exists a run ρ starting in some state $q \in v(S)$ and reaching l_k , with fewer than k discrete steps, then this run has at least 1 discrete step, as otherwise n would be a leaf of T . So we can write it $q \xrightarrow{d} q_d \xrightarrow{a} \rho'$ where a is the action of some edge e . Then ρ' is a run starting from some state $q' \in \text{Succ}(v(S), v(e))$, reaching l_k and with less than $k - 1$ discrete steps. Moreover $q' \in v(S')$ with $S' = \text{Succ}(S, e)$ (by Lemma 2). So we can apply the induction hypothesis and v is the solution to the reachability synthesis problem, starting the forward reachability algorithm from S' .

