Fortuna: Model Checking Priced Probabilistic Timed Automata

Jasper Berendsen, David N. Jansen, and Frits W. Vaandrager

Institute for Computing and Information Sciences, Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, the Netherlands

Abstract. We introduce FORTUNA, the first tool for model checking priced probabilistic timed automata (PPTAs). FORTUNA can handle the *combination* of real-time, probabilistic and cost features, which is required for addressing key design trade-offs that arise in many practical applications. For example the Zeroconf, Bluetooth, IEEE802.11 and Firewire protocols, protocols for sensor networks, and scheduling problems with failures. PPTAs are an extension of probabilistic timed automata (PTAs) with cost-rates and discrete cost increments on states. FORTUNA is able to compute the maximal probability by which a state can be reached under a certain cost-bound (and time-bound). Although this problem is undecidable in general, there exists a semi-algorithm that produces a non-decreasing sequence of probabilities converging to the maximum. This paper presents a number of crucial optimizations of that algorithm. Since PPTAs are PTAs with trivial cost parameters, we were able to compare the performance of FORTUNA with existing approaches for PTAs. Surprisingly, although PPTAs are more general, our techniques exhibit superior performance.

1 Introduction

Model checking technology has initially been developed for finite state models. Both hardware and communication protocols may be modelled naturally in terms of finite state models, and in these areas model checking has been very successful [13]. In practice, however, finite state models are often not sufficiently rich. A characteristic of embedded and cyber-physical systems, is that they have to meet a multitude of *quantitative* constraints. These constraints involve the resources that a system may use (computation resources, power consumption, memory usage, communication bandwidth, costs, etc.), assumptions about the environment in which it operates (arrival rates, hybrid behaviour), and requirements on the services that the system has to provide (timing constraints, QoS, availability, fault tolerance, etc.). In order to handle quantitative constraints, model checking technology has been extended with features for specifying real-time, probabilistic behaviour, and costs. Efficient tools have been developed such as Uppaal [3], Uppaal Cora [5] and PRISM [14], that have been successfully applied to challenging problems in different areas [3, 5, 14]. Nevertheless, until now

* Supported by NWO/EW project 612.000.103 FRAAI, and the European Community's 7th Framework Programme No. 214755 (QUASIMODO).

no model checking tool was able to handle the *combination* of real-time, probabilistic and cost features. For many practical applications, however, the key design trade-offs can only be addressed by models that incorporate all these features. We give three examples:

- Operation of the Zeroconf protocol [8] depends in an essential way on both timing and probabilities. In order to determine the optimal value for some parameters (like the number of retransmissions) one needs a cost function that combines timing delays and cost of failure [10, 8]. Many other protocols also require formal methods that combine probabilities, costs and timing.
- Timing plays an essential role in communication protocols for sensor networks. In a network with battery-powered devices, the limited energy budget can be modelled using costs. Probabilities are needed to model node failure and message loss.
- In scheduling problems it may be useful to consider the probability that a resource (e.g., a production machine) breaks down or produces imperfect output.

This paper presents FORTUNA, the first model checking tool that is able to deal with the combination of probabilities, costs and timing. FORTUNA is based on the model of priced probabilistic timed automata (PPTAs) introduced in [9, 17]. PPTAs equip timed automata with prices and probabilities on discrete transitions. Cost-*rates* indicate the increase of cost per time unit, whereas prices on discrete transitions indicate instantaneous costs. PPTAs are the orthogonal extension of both probabilistic timed automata (PTAs) [15] and priced timed automata [4, 1], as PTAs extend timed automata with probabilities on discrete transitions and priced timed automata extend timed automata with prices. FORTUNA is able to compute the fundamental problem of cost-bounded maximal probabilistic reachability (CBMR) for PPTA. CBMR determines the maximal probability by which a state can be reached under a certain cost-bound (and time-bound.) Sections 6 gives two examples that show the usefulness of CBMR in practice.

As PTAs are PPTAs with trivial cost parameters, we were able to compare the performance of FORTUNA with existing approaches for PTAs that compute maximal probabilistic reachability. The comparison is made on a number of existing PTA case studies to the best approaches available: the game-based verification of [16], and the backwards reachability approach of [18]. Surprisingly, although FORTUNA is more general, it shows the best performance.

FORTUNA adds a number of crucial optimizations to the algorithm described in [9]. The algorithm of [9] performs symbolic backwards exploration, in the spirit of the backwards reachability approach of [18]. Like that work, FORTUNA only adds intersections of symbolic states to the state space, thereby reducing the number of stored states. To compute the probability, the explored symbolic state graph is transformed into a Markov decision process that is analysed with existing techniques. For PPTA FORTUNA may not terminate, since the problem is known to be undecidable in general [7]. But, for increasing exploration depth, the produced sequence of probabilities is non-decreasing and converges to the maximum probability [9].

The optimizations described in this article increase performance drastically. Three optimizations make modifications to the symbolic state graph that is explored, by generating abstractions that preserve probabilistic reachability. The proofs are done in a rigorous way by the use of (probabilistic) simulation relations. The last optimization employs Hasse diagram data structures to speed up comparisons between symbolic states.

Other approaches for maximal probabilistic reachability on PTA use quite different techniques. Game-based verification [16] uses an abstraction–refinement scheme to iteratively generate tighter lower and upper bounds on the solution. We present a detailed comparison of FORTUNA with the results from [16]. We do not compare with the digital clocks approach of [17] since the same authors have shown game-based verification to be much faster [16].

Organization of the paper Section 2 are the preliminaries, introducing definitions and lemmas from: probability theory, Markov decision processes, and automata theory. Section 3 introduces the model of PPTAs and cost-bounded maximal probabilistic reachability. In Section 4, the algorithm to compute cost-bounded maximal probabilistic reachability on PPTAs is discussed, its correctness theorem, and several optimizations and their correctness theorems. Section 5 discusses some implementation issues of FORTUNA. Section 6 compares FORTUNA with existing approaches on case studies, and shows the usefulness of our optimizations. In addition verification of an example PPTA model is shown. Finally, Section 7 concludes this work.

The FORTUNA tool and case studies discussed in this paper are available from `http://www.cs.ru.nl/J.Berendsen/fortuna/`.

2 Preliminaries

In this section, we provide a summary of basic mathematical notions necessary for our development. In particular, we review the definitions of probability spaces, Markov decision processes, and probabilistic reachability.For a more leisurely introduction we refer to []. Our reader is encouraged to skip (portions of) this section as he sees fit.

2.1 Probability Spaces

Let V be a set. A subset $\mathcal{F}$ of 2^V is said to be a *σ-field* over V if $\mathcal{F}$ satisfies the following properties:

- $V \in \mathcal{F}$
- if $W \in \mathcal{F}$ then $V \backslash W$ is also in $\mathcal{F}$ (closed under complement)
- if $W_1, W_2, \ldots \in \mathcal{F}$ then $W_1 \cup W_2 \cup \ldots$ is also in $\mathcal{F}$ (closed under countable union)

The pair $(V, \mathcal{F})$ is called a measurable space.

A *measure* over $(V, \mathcal{F})$ is a function $\mu : \mathcal{F} \to \mathbb{R}_{\geq 0}$ such that $\mu(\emptyset) = 0$, and for each countable set Γ of pairwise disjoint elements of $\mathcal{F}$, we have that $\mu\left(\bigcup_{G \in \Gamma} G\right) = \sum_{G \in \Gamma} \mu(G)$. If $\mu(V) \leq 1$, then μ is a *sub-probability measure*. In case $\mathcal{F} = 2^V$ a special situation arises: for each set $W \in \mathcal{F}$ we have that $\mu(W) = \sum_{w \in W} \mu(w)$, where $\mu(w) = \mu(\{w\})$. In this case μ is called a *distribution*. The set of distributions over V is denoted $\mathrm{Dist}(V)$. The *support* of a distribution μ is defined as $\mathsf{supp}(\mu) \stackrel{\mathsf{def}}{=} \{v \in V \mid \mu(v) > 0\}$.

2.2 Markov Decision Processes

Markov decision processes (MDPs) are widely used to formally model and analyse systems that exhibit both nondeterministic and probabilistic behaviour.

Definition 1 (MDP). *Let* Act *be a fixed set of actions. An* MDP *is a tuple* $M = (S, s_{\mathsf{init}}, T)$*, where* S *is a set of states,* $s_{\mathsf{init}} \in S$ *is the initial state, and* $T \subseteq S \times \mathsf{Act} \times \mathrm{Dist}(S)$ *is a probabilistic transition relation. We require that* T *is total in the sense that, for each state* s*, there exists an action* a *and a distribution* μ *such that* $(s, a, \mu) \in T$*.*

The restriction of having at least one distribution for each state is imposed to ensure that policies (defined below) always exist.

Intuitively speaking, an MDP describes the following behaviour. Whenever the system is in some state, $s \in S$ say, an action $a \in \mathsf{Act}$ and a distribution μ with $(s, a, \mu) \in T$ are selected nondeterministically. Subsequently, the next state is selected probabilistically according to μ, i.e. the next state r is selected with probability $\mu(r)$. Thus, a transition involves resolving both a nondeterministic and a probabilistic choice.

Note that our definition of MDPs allows sub-distributions for the transition relation. For $(s, a, \mu) \in T$, the remaining probability $1 - \sum_r \mu(r)$ may be interpreted as a deadlock probability. Note that any MDP can easily be transformed into an MDP that uses only complete distributions by adding a trapping state s_{trap} that is equipped with a self-loop with probability 1, such that the 'missing' probabilities of all sub-distributions lead to s_{trap}. For the rest of the paper we implicitly apply this transformation when needed.

A *probabilistic transition* $s \xrightarrow{a,\mu}$ is made by nondeterministically selecting an action a and a distribution $\mu \in \mathrm{Dist}(S)$ such that $(s, a, \mu) \in T$. A *transition* $s \xrightarrow{a,\mu} r$ is made by the probabilistic transition $s \xrightarrow{a,\mu}$ followed by choosing next state $r \in S$ with probability $\mu(r) > 0$. In case μ is a Dirac distribution we also write $s \xrightarrow{a} r$.

An *infinite path* starting in state s_0 is an infinite sequence of transitions: $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \xrightarrow{a_2,\mu_2} \cdots$ such that $(s_i, a_i, \mu_i) \in T$ for all i. Let ω^i denote the ith state in the path ω, i.e. $\omega^i = s_i$.

A *finite path* is a finite prefix of an infinite path. We denote the last state in a finite path ω by $\mathrm{last}(\omega)$. The *length* of a path is the number of transitions that it contains. A path of length 0 consists only of the starting state.

We assume a special action label $\tau \in \mathsf{Act}$ that denotes *internal* transitions. We assume internal transitions to be non-probabilistic, i.e. $s \xrightarrow{\tau,\mu}$ implies $\mu = \{r \mapsto 1\}$, for some r. When we have a finite path of τ-transitions $s_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n$ we write $s_0 \Rightarrow s_n$.

Automata (also called labelled transition systems) can be seen as a specialization of MDPs, where the probabilistic transition relation uses only Dirac distributions. Alternatively we may redefine the transition relation to go to a single next state instead of a distribution, as is done in the following definition.

Definition 2 (Automaton). *An* automaton *is a tuple* $(S, s_{\mathsf{init}}, D)$*, where* S *is a set of states,* $s_{\mathsf{init}} \in S$ *is the initial state, and* $D \subseteq S \times \mathsf{Act} \times S$*.*

Let $\mathsf{Paths}^{\infty}_M$ and Paths^{*}_M denote the infinite and finite paths, respectively, of MDP M. In order to associate a probability space to an MDP, we first need to resolve all nondeterministic choices. To this end, we consider policies (also called strategies, schedulers, or adversaries.) A *(deterministic) policy* is a function that maps every finite path ω in an MDP to an action $a \in \mathsf{Act}$ and a distribution $\mu \in \mathrm{Dist}(S)$ such that $(\mathrm{last}(\omega), a, \mu) \in T$. We use $\mathsf{Pol}(M)$ to denote the set of all deterministic policies of an MDP M. A policy resolves all nondeterminism, and therefore an MDP together with a policy yields a discrete-time Markov chain.

In the literature also a more general notion of policy has been proposed, which maps every finite path to a *distribution* of probabilistic transitions. From [20] we know that deterministic policies are sufficient to obtain maximal probabilistic reachability (defined below) when a bounded number of transitions may be used to reach the goal. In case the number of states and probabilistic transitions in the MDP is finite, this also holds when an unbounded number of transitions may be used to reach the goal.

For a given state s, we now define probability measure $\mathsf{Prob}_A(s)$ on $\mathsf{Paths}^{\infty}_M$. Let $\mathcal{F}$ be the smallest σ-field over $\mathsf{Paths}^{\infty}_M$ such that every *cone* of some finite path $\omega \in \mathsf{Paths}^{*}_M$ is in $\mathcal{F}$. A *cone* of a finite path is the set all infinite continuations of the path. Formally, the cone $\mathcal{C}(\omega)$ of $\omega \in \mathsf{Paths}^{*}_M$ is defined as $\{\omega' \in \mathsf{Paths}^{\infty}_M \mid \omega < \omega'\}$, where $<$ is the standard prefix order on sequences. The function $\mathbf{Q}_A$ assigns probabilities to finite paths according to decisions made by policy A and a given starting state s. Formally, for any $r, s \in S$ and $\omega \in \mathsf{Paths}^{*}_M$, we define $\mathbf{Q}_A$ inductively as follows:

$$\mathbf{Q}_A(s, r) \stackrel{\mathsf{def}}{=} \begin{cases} 1 & \text{if } s = r \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{Q}_A(s, \omega \xrightarrow{a,\mu} r)) \stackrel{\mathsf{def}}{=} \begin{cases} \mathbf{Q}_A(s, \omega) \cdot \mu(r) & \text{if } A(\omega) = (a, \mu) \\ 0 & \text{otherwise} \end{cases}$$

The probability assigned to a cone $\mathcal{C}(\omega)$ equals the probability assigned by $\mathbf{Q}_A$ to the finite path ω: $\mathsf{Prob}_A(s)(\mathcal{C}(\omega)) \stackrel{\mathsf{def}}{=} \mathbf{Q}_A(s, \omega)$. By standard measure theoretical arguments, $\mathsf{Prob}_A(s)$ is a measure over $(\mathsf{Paths}^{\infty}_M, \mathcal{F})$.

2.3 Probabilistic Reachability

The *reach probability* is the likelihood to reach a certain set of goal states in a finite number of transitions under some policy. For a starting state $s \in S$, a set of goal states $G \subseteq S$, and policy A, it is formally defined as

$$\mathsf{ProbReach}_A(s, G) \stackrel{\mathsf{def}}{=} \mathsf{Prob}_A(s)(\{\omega \in \mathsf{Paths}^\infty_M \mid \exists i \in \mathbb{N} \,.\, \omega^i \in G\})$$

The set $\{\omega \in \mathsf{Paths}^\infty_M \mid \exists i \in \mathbb{N} \,.\, \omega^i \in G\}$ is measurable by $\mathsf{Prob}_A(s)$, because it may be written as the countable union $\bigcup\{\mathcal{C}(\omega) \mid \omega \in \mathsf{Paths}^*_M \wedge \mathrm{last}(\omega) \in G\}$. The *reach probability using not more than n transitions* is defined as

$$\mathsf{ProbReach}^{\leq n}_A(s, G) \stackrel{\mathsf{def}}{=} \mathsf{Prob}_A(s)(\{\omega \in \mathsf{Paths}^\infty_M \mid \exists i \in [0, n] \,.\, \omega^i \in G\})$$

Like above $\{\omega \in \mathsf{Paths}^\infty_M \mid \exists i \in [0, n] \,.\, \omega^i \in G\}$ is measurable by $\mathsf{Prob}_A(s)$. When $s = s_{\mathsf{init}}$ we write $\mathsf{ProbReach}_A(G)$ and $\mathsf{ProbReach}^{\leq n}_A(G)$, respectively.

Lemma 1 (Convergence). *Let $M = (S, s_{\mathsf{init}}, T)$ be an MDP, $A \in \mathsf{Pol}(M)$, $s \in S$, and $G \subseteq S$. Then $\langle \mathsf{ProbReach}^{\leq n}_A(s, G) \rangle_{n \in \mathbb{N}}$ is a non-decreasing sequence in $[0, 1]$ converging to $\mathsf{ProbReach}_A(s, G)$.*

Proof. By Lemma 34 of [18]. □

The following definition shows how for any policy A and finite path ω, we can construct a policy $A[\omega]$ that acts like A when path ω has already occurred.

Definition 3. *For a policy A and finite path ω, let $A[\omega]$ be the policy such that for any finite path ω':*

$$A[\omega](\omega') \stackrel{\mathsf{def}}{=} \begin{cases} A(\omega \xrightarrow{a,\mu} \omega'') & \text{if } \omega' \text{ is of the form } \mathrm{last}(\omega) \xrightarrow{a,\mu} \omega'' \\ A(\omega') & \text{otherwise} \end{cases}$$

We have the following known lemma.

Lemma 2. *Given state s, policy A such that $A(s) = (a, \mu)$, and set of paths $\Omega \in \mathsf{Paths}^\infty_M$ measurable by $\mathsf{Prob}_A(s)$, then:*

$$\mathsf{Prob}_A(s)(\Omega) = \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} \omega) \in \Omega\})$$

Proof. See Appendix A.1.

Lemma 3. *Let $A(s) = (a, \mu)$. If $s \notin G$, then:*

1. *for any $n \in \mathbb{N}$: $\mathsf{ProbReach}^{\leq n+1}_A(s, G) = \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{a,\mu} r]}(r, G)$*
2. *$\mathsf{ProbReach}_A(s, G) = \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}_{A[s \xrightarrow{a,\mu} r]}(r, G)$*

Proof. We prove result 1. The proof of result 2 is very similar: the only difference is that there is no upperbound n on the length of the paths.

$$
\begin{aligned}
&\mathsf{ProbReach}_A^{\leq n+1}(s, G) \\
&= \mathsf{Prob}_A(s)(\{\omega \in \mathsf{Paths}_M^\infty \mid \exists i \leq n+1.\omega^i \in G\}) \quad \text{by definition of ProbReach} \\
&= \mathsf{Prob}_A(s)(\{\omega \in \mathsf{Paths}_M^\infty \mid \exists i \in [1, n+1].\omega^i \in G\}) \quad \text{since } s \notin G \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \in \mathsf{Paths}_M^\infty \mid \exists i : [0, n].\omega^i \in G\}) \quad \text{by Lemma 2} \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}_{A[s \xrightarrow{a,\mu} r]}^{\leq n}(r, G) \quad \text{by definition of ProbReach}
\end{aligned}
$$

□

The reach probability depends on the nondeterministic choices made by the policy. A nondeterministic choice can be used to model a branch in system execution that cannot be resolved probabilistically, or for which the probability distribution is not known. The *maximal reach probability* is of interest, i.e. the maximal attainable value if all choices are optimal. The maximal reach probability for $G \subseteq S$ is defined as:

$$\mathsf{SupProbReach}_M(G) \stackrel{\mathsf{def}}{=} \sup_{A \in \mathsf{Pol}(M)} \mathsf{ProbReach}_A(G)$$

And using not more than n transitions:

$$\mathsf{SupProbReach}_M^{\leq n}(G) \stackrel{\mathsf{def}}{=} \sup_{A \in \mathsf{Pol}(M)} \mathsf{ProbReach}_A^{\leq n}(G)$$

From [20] we know that if A has a finite number of states and probabilistic transitions, then we can find the policy that obtains the value of $\mathsf{SupProbReach}_M(G)$. Moreover, we may restrict to policies that always takes the same decision in each state, irrespective of the path or the length of the path by which the state was reached. The value of $\mathsf{SupProbReach}_M(G)$ can be computed by several techniques including: value iteration, (modified) policy iteration and linear programming.

3 Priced Probabilistic Timed Automata

3.1 Time, Clocks and Guards

A *clock* is a real-valued variable that can be used to measure the elapse of time. All clocks run at the same pace. A *clock valuation* assigns a non-negative value to each clock in some set $\mathbb{X}$. Let $\mathbb{R}_{\geq 0}^{\mathbb{X}}$ denote the set of all possible clock valuations. A clock valuation $v \in \mathbb{R}_{\geq 0}^{\mathbb{X}}$ is thus a mapping $\mathbb{X} \to \mathbb{R}_{\geq 0}$. For $d \in \mathbb{R}_{\geq 0}$, let $v+d$ denote the clock valuation that maps each $x \in \mathbb{X}$ to $v(x) + d$. For $R \subseteq \mathbb{X}$, let $v[R := 0]$ denote the clock valuation in which the clocks in R have been reset, i.e. $v[R := 0](x)$ equals 0 if $x \in R$ and $v(x)$ otherwise.

A *guard* is a conjunction of inequalities where the value of a single clock is compared to an integer. Formally, the set $\mathsf{Guards}(\mathbb{X})$ of guards g is defined by the grammar:

$$g ::= x \bowtie b \mid g \wedge g \mid \mathsf{true}, \quad \text{where } x \in \mathbb{X}, b \in \mathbb{N}, \bowtie \in \{<, \leq, \geq, >\}.$$

(In)equalities such as $(x = b)$ and $(2 \leq x - y \leq 3)$ are abbreviations for a conjunction of multiple inequalities. We write $v \models g$ when valuation v satisfies the constraints of guard g.

3.2 PPTAs and their semantics

We are now in a position to define *PPTAs*.

Definition 4 (PPTA). *A* priced probabilistic timed automaton *(PPTA) is a tuple* $\mathcal{A} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{inv}, \mathsf{edges}, \dot{\$})$*, where:*

- L *is a finite set of locations;*
- $l_{\mathsf{init}} \in L$ *is the initial location;*
- $\mathbb{X}$ *is a finite set of clocks;*
- $\mathsf{inv} : L \to \mathsf{Guards}(\mathbb{X})$ *assigns an invariant to each location;*
- $\mathsf{edges} \subseteq L \times \mathsf{Guards}(\mathbb{X}) \times \mathrm{Dist}(2^{\mathbb{X}} \times \mathbb{N} \times L)$ *is a finite set of edges; and*
- $\dot{\$} : L \to \mathbb{N}$ *associates a cost-rate with each location.*

For edge $(l, g, p) \in \mathsf{edges}$, l denotes the source location, g the guard, and p a distribution over *instantaneous effects*, which consist of a set of clocks to be reset, a cost increment, and a destination location. For the rest of the paper we fix a PPTA $\mathcal{A} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{inv}, \mathsf{edges}, \dot{\$})$.

Example 1 Figure 1 shows a PPTA with clock x. Locations are represented by rounded boxes, with branching arrows between them denoting the edges of the PPTA. The initial location is l_0. Invariants and cost-rates are written in the locations. Guards (e. g. $x \leq 0$) are placed next to the source location; probabilities, resets and cost increments are at the branches (e. g. probability 0.3 and $x{:=}0$.) Probabilities 1, guards that always hold, and instantaneous cost increments of 0 are left out.

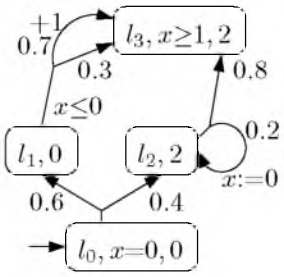

Fig. 1. Example PPTA

Intuitively, a PPTA behaves as follows. It always is in a state consisting of a location l, a clock valuation v, and the cost incurred until now c. A policy makes the nondeterministic choice between delaying or which edge to take. Only edges with guards satisfying the current valuation are available. Delaying will increase each clock by the quantity of delay and the accumulated cost by the quantity of delay times the cost-rate $\dot{\$}(l)$. When taking an edge, a reset set, cost increment, and destination location are chosen probabilistically. They determine which clocks are reset, the cost increment, and the next location.

Definition 5 (PPTA Semantics). *Fix* $\mathsf{Act} = \mathbb{R}_{\geq 0} \cup \{\tau\}$. *The* semantics *of PPTA* $\mathcal{A}$*, denoted* $[\![\mathcal{A}]\!]$*, is given by the tuple* $(S, s_{\mathsf{init}}, T)$*, where* $S = \{(l, v, c) \mid l \in L \wedge v \models \mathsf{inv}(l) \wedge c \in \mathbb{R}\}$*, i.e. states consist of a location, a clock valuation, and the accumulated cost;* $s_{\mathsf{init}} = (l_{\mathsf{init}}, \{x \mapsto 0 \mid x \in \mathbb{X}\}, 0)$*; and* $((l, v, c), d, \mu) \in T$ *iff* $(l, v, c) \in S$*,* $d \in \mathbb{R}_{\geq 0}$*,* $v + d \models \mathsf{inv}(l)$ *and one of the following conditions holds:*

- *either* $\mu(l, v + d, c + \dot{\$}(l)d) = 1$
- *or there exists* $(l, g, p) \in \mathsf{edges}$ *such that* $v + d \models g$ *and for all* $(l', v', c') \in S$*:* $\mu(l', v', c') = \sum_{R \subseteq \mathbb{X}\ s.t. v' = (v+d)[R:=0]} p(R, c' - d \cdot \dot{\$}(l) - c, l')$

If the first condition holds then we call $((l, v, c), d, \mu)$ *a* time transition*, otherwise we call it a* delayed discrete transition.

It is straightforward to prove that $[\![\mathcal{A}]\!]$ is an MDP, where time transitions spending zero time ensure every state has at least one outgoing transition. Time divergence of $[\![\mathcal{A}]\!]$ is not an issue for this paper since we are dealing with reachability properties, see [18].

3.3 Cost-Bounded Maximal Reachability

Cost-bounded maximal reachability (CBMR) is the maximal probability by which a goal location in a PPTA can be reached, without the accumulated cost exceeding some bound. For PPTA $\mathcal{A}$ we fix $l_{\mathsf{goal}} \in L$ to be the goal state, and $c_{\mathsf{bound}} \in \mathbb{N}$ to be the cost-bound.

Definition 6 (CBMR). *CBMR is the probability* $\mathsf{SupProbReach}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}})$*, where* $\sigma_{\mathsf{goal}} = \{l_{\mathsf{goal}}\} \times \mathbb{R}^{\mathbb{X}}_{\geq 0} \times [0, c_{\mathsf{bound}}]$.

Naturally, by comparing CBMR to a probability p, CBPR can be used to answer the question "Is it possible to reach location l_{goal} with probability at least $p \in (0, 1]$ and with cost at most c_{bound}?".

3.4 Predecessor Operations

We now define predecessor operations on sets of states of $[\![\mathcal{A}]\!]$. Predecessor operations are essential to the symbolic backward exploration that is done in our algorithm. The *discrete predecessor* of a set of states Z via edge e and instantaneous effect f, gives all states in $[\![\mathcal{A}]\!]$ that can reach some state in Z via edge e and instantaneous effect f. The *time predecessor* of a set of states Z, gives all states in $[\![\mathcal{A}]\!]$ that can reach some state in Z by letting time elapse.

Definition 7 (Predecessor Operations). *Let* $[\![\mathcal{A}]\!] = (S, s_{\mathsf{init}}, T)$*. For any* $Z \subseteq S$*,* $e = (l, g, p) \in \mathsf{edges}$*. Let* $f = (R, h, l') \in \mathsf{supp}(p)$*. The discrete predecessor and time predecessor operations are respectively:*

$$\mathsf{dpre}_{e,f}(Z) \stackrel{\mathsf{def}}{=} \{(l, v, c) \in S \mid v \models g \wedge (l', v[R := 0], c + h) \in Z\}$$

$$\mathsf{tpre}(Z) \stackrel{\mathsf{def}}{=} \{s \in S \mid \exists d \geq 0. \exists r \in Z. s \xrightarrow{d} r\}$$

Lemma 4 (Properties of Predecessors).

- *If* $S = \mathsf{dpre}_{e,f}(T)$*, then for all* $s \in S$ *there exists* $r \in T$ *such that* $s \xrightarrow{0,\mu} r$.
- *If* $S = \mathsf{tpre}(T)$*, then for all* $s \in S$ *there exists* $r \in T$ *and* $d \in \mathbb{R}_{\geq 0}$ *such that* $s \xrightarrow{d} r$.

Proof. Straightforward from the definition of tpre and dpre. □

4 The Algorithm

Algorithm 1 is used for computing CBMR for paths of length up to maxlength (possibly ∞). It gets as inputs a CBMR problem, i.e. a PPTA $\mathcal{A}$, a goal location l_{goal}, a cost-bound c_{bound}, and the maximal path length maxlength.[1] Algorithm 1 returns a finite automaton $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$, which we will call *reachability graph*. The reachability graph captures symbolically all transitions by which a target state may be reached. Definition 8 defines an MDP M for the reachability graph. Now, from Theorem 1, we see that:

1. The maximal reach probability in M is an upperbound on the CBMR solution.
2. If $\mathsf{maxlength} = \infty$ the upperbound is precise.
3. Increasing maxlength leads to a higher or equal upperbound on CBMR with unbounded lenght.

Zones are sets of states of $[\![\mathcal{A}]\!]$ that share the same location. Since clocks and cost take real values, zones may (and will) contain infinitely many states. In [9] it is shown that the zones generated by our algorithm have a finite representation called multi-priced zones. Multi-priced zones are closed under the operations of the algorithm. They are a subclass of convex polyhedra.

Visited are the zones that are generated by Algorithm 1 in a backward fashion: starting from the zone of goal states (σ_{goal}), more zones are generated by repeatedly computing predecessors of explored zones. The predecessor of a zone is computed by first computing the time predecessor, and from that the discrete predecessor. The operations are combined to avoid storing an intermediate zone. The algorithm proceeds in a breadth-first way, by first calculating the predecessors of all zones that were waiting to be explored, before computing predecessors of the newly generated zones. In addition to predecessors, intersections of explored zones are added to *Visited*, which are zones themselves.

We now discuss the algorithm line by line. An example is given later. Throughout this work, edges a reachability graph may be called *directions* to distinguish them from the edges in a PPTA. An important property of any direction generated by Algorithm 1 is the following:

> A direction (σ, e, f, ρ) means any state in ρ is reachable from some state in σ by a delayed discrete transition with 0 delay using edge e and instantaneous effect f, followed by a time transition.

[1] As a minor condition, $\mathsf{inv}(l_{\mathsf{init}})$ should require all clocks to be zero.

Algorithm 1: The basic backwards reachability algorithm

```
1  Input: PPTA A = (L, l_init, X, inv, edges, $), l_goal, c_bound, maxlength,
   where inv(l_init) = ⋀_{x∈X}(x = 0)
   Output: (Visited, σ_init, D)
2  σ_goal := l_goal × inv(l_goal) × [0, c_bound]
3  σ_init := {(l_init, {x ↦ 0 | x ∈ X}, 0)}
4  if σ_init ⊆ σ_goal then σ_init := σ_goal
5  Visited := {σ_goal, σ_init}
6  D := {(σ_goal, τ, σ_goal), (σ_init, τ, σ_init)}
7  Waiting_0 := {σ_goal}
8  for i := 1 to maxlength
9    if Waiting_{i−1} = ∅ then break
10   Waiting_i := ∅
11   foreach ρ ∈ Waiting_{i−1}
12     foreach e ∈ edges and f is an inst. effect of e going to the location of ρ
13       σ := dpre_{e,f}(tpre(ρ))
14       if σ ≠ ∅
15         D := D ∪ {(σ, e, f, ρ)}
16         if σ ∉ Visited then Waiting_i := Waiting_i ∪ {σ}, Visited := Visited ∪ {σ}
17         foreach (σ', a', ρ') ∈ D
18           if σ ∩ σ' ≠ ∅
19             D := D ∪ {(σ ∩ σ', e, f, ρ), (σ ∩ σ', a', ρ')}
20             if σ ∩ σ' ∉ Visited
21               Waiting_i := Waiting_i ∪ {σ ∩ σ'}, Visited := Visited ∪ {σ ∩ σ'}
22 return (Visited, σ_init, D)
```

Line 2 Zone σ_{goal} is the goal set of states.

Line 3 Zone σ_{init} is a singleton consisting of the initial state.

Line 4 Needed for the special case when $\sigma_{\mathsf{init}} \subset \sigma_{\mathsf{goal}}$.

Line 5 *Visited* maintains the zones that were generated and starts as $\{\sigma_{\mathsf{goal}}, \sigma_{\mathsf{init}}\}$.

Line 6 D maintains the directions of the reachability graph.

Line 7 $Waiting_i$ are zones to be explored after iteration i. Goal zone σ_{goal} is the first zone to be explored.

Line 8 The algorithm takes exactly maxlength steps in the outer loop. All the paths of maximal length maxlength of $[\![\mathcal{A}]\!]$ are symbolically explored, including those having a loop.

Line 9 Quit the loop if there is nothing left to explore.

Line 10 The set of zones waiting to be explored in the next iteration is initially $\emptyset$.

Line 11 Pick an arbitrary zone ρ from the current set of zones waiting to be explored.

Line 12 Explore all incoming edge/effect combinations e, f of the PPTA for ρ.

Line 13 Compute the predecessor σ via e, f.

Line 14 Only proceed if the generated predecessor σ is not empty.

Line 15 Add the direction.

Line 16 In case σ was not visited before, add it to the next set of waiting zones.

Lines 17–21 are concerned with the intersections. Intersections are useful for the following reason. Given zones ρ and ρ', let $\sigma = \mathsf{dpre}_{e,f_1}(\mathsf{tpre}(\rho))$ and $\sigma' = \mathsf{dpre}_{e,f_2}(\mathsf{tpre}(\rho'))$ for some $e \in \mathsf{edges}$ and f_1, f_2 instantaneous effects of e. Now, by Lemma 4, $\sigma \cap \sigma'$ contains the states that can reach ρ as well as ρ': A state in ρ is reached by taking edge e, probabilistically choosing f_1, and delaying some time. A state in ρ' is reached by taking the same edge e, probabilistically choosing f_2, and delaying some (possibly different) time. Whenever $f_1 \neq f_2$ and the goal can be reached via both ρ and ρ', the probabilities of choosing either effect can be added. This explains why we first took the time predecessor and then the discrete predecessor: only after the probabilistic choice has been resolved, the policy gets the information whether f_1 or f_2 was chosen and can decide how long to delay to reach ρ or ρ', respectively. This reasoning can easily be generalized to intersections of more than two zones.

Line 17 Pick each direction in an arbitrary order.

Line 18 Proceed on a non-empty intersection of the source zones.

Line 19 The intersection gets outgoing directions to the target zones of the two directions. These directions capture the possibility of states in the intersection to go in both ways. Note that in subsequent iterations of the loop on lines 16–20 more directions may be added that start from the same intersection. Intersections between multiple zones are generated by pair-wise intersection.

Lines 20, 21 In case the intersection was not visited before, add it to the next set of waiting zones.

Example 2 Figure 2 shows the reachability graph that Algorithm 1 would generate for the PPTA of Figure 1 if $l_{\mathsf{goal}} = l_3$ and $c_{\mathsf{bound}} = 3$. In the figure, zones are labeled $\sigma_{\mathsf{init}}, A, B, \ldots$, and are represented by planes in coordinate systems with x and c at the axes. Furthermore, zones have a location. Labels on directions are left out and probabilities are added in such a way that there is no ambiguity from which edge in the PPTA they were generated. Plain edges are the ones resulting from line 15. Thus, for example B is the predecessor of A. The predecessors of $B, \ldots, E$ are left out for brevity. Dashed edges are the ones resulting from line 19. Note that F is the intersection of D and E, while C is the intersection of C and B. Zone F has only an empty predecessor.

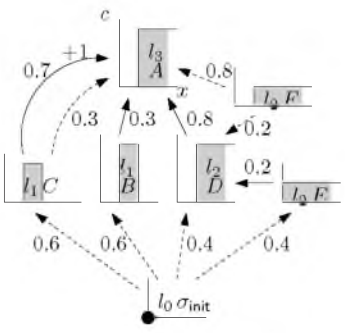

Fig. 2. Reachability graph

The next definition is used to transform any reachability graph that is generated by Algorithm 1 into an MDP. As such, it generates the symbolic semantics for the PPTA under the given CBMR problem.

Example 3 Figure 3 shows the MDP of Definition 8 for the reachability graph of Figure 2. The positions of the states correspond to the positions of the zones in the reachability graph. For brevity probabilities and labels have been left out.

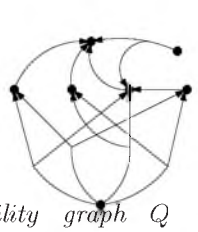

Fig. 3. Constructed MDP without optimizations

Definition 8 (Symbolic semantics). *Given reachability graph* $Q = (S, s_{\mathsf{init}}, D)$*, we define the Markov decision process* $\mathsf{MDP}(Q) = (S, s_{\mathsf{init}}, T)$*, where* $(s, a, \mu) \in T$ *if and only if either*

- $a = \tau$ *and* $\mu = \{r \mapsto 1\}$ *and there exists* $(s, \tau, r) \in D$*; or*
- $a = e$ *and there exists* $D_{e,\mu} \subseteq D$ *such that*
  1. $\forall (s', e', f', r') \in D_{e,\mu}.s' = s \wedge e' = e$
  2. $\forall (s_1, e_1, f_1, r_1), (s_2, e_2, f_2, r_2) \in D_{e,\mu}.r_1 \neq r_2 \implies f_1 \neq f_2$
  3. $D_{e,\mu}$ *is maximal*
  4. $\forall r \in S.\mu(r) = \sum_{(s,(l,g,p),f,r) \in D_{e,\mu}} p(f)$

Theorem 1. *Assume Algorithm 1 is executed with input PPTA* $\mathcal{A}$*, location* l_{goal}*,* $c_{\mathsf{bound}} \in \mathbb{N}$*, and* $\mathsf{maxlength} \in \mathbb{N} \cup \{\infty\}$*. And on the output we define* M *by Definition 8. Now all of the following hold:*

1. $\forall n \leq \mathsf{maxlength}.\ \mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}}) = \mathsf{SupProbReach}^{\leq n}_{M}(\{\sigma_{\mathsf{goal}}\})$
2. $\forall n \leq \mathsf{maxlength}.\ \mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}}) \leq \mathsf{SupProbReach}_{M}(\{\sigma_{\mathsf{goal}}\})$
3. *If* $\mathsf{maxlength} = \infty$ *we have that:* $\mathsf{SupProbReach}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}}) = \mathsf{SupProbReach}_{M}(\{\sigma_{\mathsf{goal}}\})$
4. *Let* $\mathsf{maxlength} = m \neq \infty$*, and construct* M' *from the same input as* M *except that* $\mathsf{maxlength}$ *is set to* $m + 1$*. Then, the probability does not decrease:* $\mathsf{SupProbReach}_{M}(\{\sigma_{\mathsf{goal}}\}) \leq \mathsf{SupProbReach}_{M'}(\{\sigma_{\mathsf{goal}}\})$

Proof. Result 1 follows from Theorem 1 in [9], but is rephrased in Appendix A.2. The proof goes along the same lines as the proof of Proposition 29 in [18]. However, the length of paths match in $[\![\mathcal{A}]\!]$ and M, which is possible due to the delayed discrete transitions in $[\![\mathcal{A}]\!]$. Results 2, 3, and 4 follow directly from result 1, the definition of supremum and Lemma 1. □

Definition 9 (Cover). *Given a set of zones* Σ*, for any* $\rho, \sigma \in \Sigma$*, we say* ρ covers σ*, written* $\rho \supset\!\!- \sigma$*, iff* $\rho \supset \sigma$ *and there exists no* $q \in \Sigma$ *such that* $\rho \supset q \supset \sigma$*.*

Optimization 1 extends Algorithm 1. To understand its basic idea, regard two zones ρ and ρ' with $\rho \supset \rho'$. All states in a zone have the same probabilistic transitions available, as long as these contribute to reaching the goal with maximal probability. If execution of the MDP enters ρ', this corresponds to an execution of the semantics that enters some state $r \in \rho'$. But, execution of the MDP might as well enter ρ, since $r \in \rho$. Therefore, we may add a τ-direction from ρ' to ρ. A τ-direction has probability 1 and does not correspond with any transition

Optimization 1.
Fragment A: inserted after lines 15 and 19 of Algorithm 1

15bis,19bis **foreach** $(\sigma, e, f, \rho), (\sigma', e', f', \rho') \in D$, with $(\sigma', e', f') = (\sigma, e, f)$
15ter,19ter **if** $\rho' \subset \rho$ **then** $D := D \backslash \{(\sigma, e, f, \rho)\}$

Fragment B: replaces line 22 of Algorithm 1

$22'$ **foreach** $\rho, \rho' \in$ *Visited*
$23'$ **if** $\rho \supset\!\!- \rho'$ **then** $D := D \cup \{(\rho', \tau, \rho)\}$
$24'$ **return** $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$

Optimization 2: replaces line 18 of Algorithm 1

$18'$ **if** $\sigma \cap \sigma' \neq \emptyset$ **and** $(\sigma' = \sigma_{\mathsf{init}}$ **or** $\exists f' \neq f.a' = (e, f'))$

in the semantics. Fragment B adds the minimal number of τ-directions to obtain a path of τ-directions from any subzone to a superzone. The optimization comes from Fragment A that removes directions from any zone σ to ρ, when ρ' is reachable from σ with the same label. A policy can simply go to ρ by first going to ρ' and then taking τ-directions. The benefit of Optimization 1 comes from the way Definition 8 works: given an edge, sets $D_{e,\mu}$ are constructed by taking all possible (maximal) combinations of instantaneous effects of that edge. Given some edge e with n instantaneous effects, a zone with m_i outgoing directions that use the i-th instantaneous effect would have $m_1 \cdot m_2 \cdot \ldots \cdot m_n$ possibilities for $D_{e,\mu}$, which gives a blow-up exponential in n.

Example 4 Recall Example 2. With Optimization 1 the reachability graph will differ from the one in Figure 2: the direction from σ_{init} to B will not be present, and there will be τ-directions (C, τ, B), (F, τ, D) and (F, τ, E). The generated MDP is depicted in Figure 4.

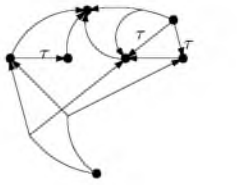

Fig. 4. MDP when using Optimization 1

Theorem 2. *Assume that we change Algorithm 1 with Optimization 1, then Theorem 1 results 2–4 still hold.*

Intersections are only useful if they capture probabilistic branching. Optimization 2 is straightforward, and is also made in [18] to suppress intersections that have only outgoing transitions with the same probability resolution.

Theorem 3. *Both for Algorithm 1 changed with Optimizations 1 and 2, and Algorithm 1 changed with only Optimization 2, theorem 1 results 2–4 still hold.*

We now define a transformation from a reachability graph Q to a reachability graph $\bar{Q}$. The transformation reduces the number of probabilistic transitions in $\mathsf{MDP}(\bar{Q})$ w. r. t. $\mathsf{MDP}(Q)$, but does not affect the maximum reach probability. The zones of Q are maintained by the transformation, but $\bar{Q}$ has extra *intermediate* zones. Two or more directions leaving a zone in Q that have the same

label are replaced in $\bar{Q}$ by a direction with the same label going to a fresh intermediate zone. From the fresh intermediate zone there are τ-directions going to the goal zones of the original directions in Q. The benefit is the same as for Optimization 1: we reduce the number of outgoing directions from a state that have the same label.

Example 5 Recall Example 4. Optimization 3 creates an intermediate state with τ transitions to zone D and zone E. Figure 5 shows the generated MDP.

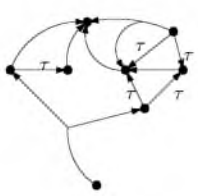

Fig. 5. MDP with all optimizations

In $\bar{Q}$ all directions leaving a zone have a different label, except for τ-directions, that all use the label τ. Because of these properties of directions in $\bar{Q}$, from Definition 8 we can see that for each zone and each direction leaving that zone there is only one possibility to construct a probabilistic transition. Each direction has a corresponding τ-direction from the intermediate zone and there is one direction to the intermediate zone, so in total at most one extra direction is needed per instantaneous effect per zone.

Optimization 3. *Given reachability graph $(S, s_{\mathsf{init}}, D)$. We define the reachability graph $(S \cup I, s_{\mathsf{init}}, \bar{D})$, where $I \subseteq S \times \mathsf{Act}$ and for any $(s, a, r) \in D$*

- *if $\exists (s, a, r') \in D . r' \neq r$ then $(s, a, (s, a)) \in \bar{D}$ and $((s, a), \tau, r) \in \bar{D}$.*
- *otherwise: $(s, a, r) \in \bar{D}$.*

Theorem 4. *Given reachability graph Q. Let $\bar{Q}$ be obtained from Q by applying the transformation of Optimization 3, then for any G:*

$$\mathsf{SupProbReach}_{\mathsf{MDP}(\bar{Q})}(G) = \mathsf{SupProbReach}_{\mathsf{MDP}(Q)}(G)$$

It is not hard to see that also without Optimization 1, Optimization 3 will still make sure each zone has no two outgoing directions with the same label, except for τ-directions. However, Optimization 1 is useful for the following reasons:

- It works during the construction of the reachability graph, i.e. directions from a zone are removed as soon as it gets a new outgoing direction.
- It needs no extra 'intermediate' states.
- The added τ-directions do not depend on the labels of the directions that were removed. As such they can be used for more directions at the same time.

5 Implementation Issues

A straightforward ehancement FORTUNA employs is to consider only locations that are reachable. To this end a standard symbolic *forward* exploration is performed, not taking probabilities into account. Since there are no guards on cost,

lowering the cost in a state by say C will not influence the locations reachable, although they are reachable with C less cost. Thus, we need to compute minimum cost reachability for combinations of clock values and locations, and store the locations that are reachable with a cost below the CBMR cost-bound. This amounts to minimum cost reachability in a priced timed automaton which is a decidable problem [4].

We use convex polyhedra instead of the multi-priced zones of [9] to represent zones, because there is a advanced library for them available: the Parma Polyhedra Library [2]. The library offers operations on convex polyhedra, such as intersection, inclusion checking, and time predecessor. Another advantage is that convex polyhedra will allow for an extension to more general classes of automata, such as the probabilistic linear hybrid automata of [21].

A disadvantage could be reduced performance, although it has yet to be investigated if multi-priced zones allow for a more efficient implementation than general convex polyhedra. Some operations of the Parma Polyhedra Library take considerable computation time, most notably the inclusion and intersection operations on polyhedra. To reduce the number of intersections, we maintain a Hasse diagram structure for zones that share the same location. The Hasse diagrams use the $\supset$– relation to order their zones. The top and bottom elements are added. Now, as a new zone enters the explored state space on line 17, it will be inserted in the Hasse diagram that corresponds to its location. This means more inclusion checks to determine the position of the new element in the Hasse diagram, but the benefit comes at line 18 of Algorithm 1: If one zone includes the other, the smaller is the intersection of the two, and inclusion can now be quickly decided based on the Hasse diagram. Additionally, we reuse the diagram for the τ-directions of Optimization 1: they are no longer stored. However, when the MDP is generated, they are treated as if they were stored.

6 Case Studies and Model Checking Results

We present two simple case studies that illustrate the practical usefulness of PPTAs and CBMR. In addition, we present experimental results for some PTA case studies, taken from [16], which do not include costs. Even though these case studies only exploit part of the functionality of FORTUNA, they allow us to compare the performance of FORTUNA to other tools. We also demonstrate the usefulness of our optimizations by comparing non-optimized versions of our algorithm to optimized ones.

FORTUNA uses a CCS style parallel composition, whereas the approaches that we compare with have a CSP style composition. Due to this, the models used by the different tools are not entirely isomorphic. We tried to stay as close as possible to the original case studies. We did not change the number of locations in the PTAs, but only added intermediate locations on some transitions when needed. Two case studies are concerned with calculating a minimum probability. However, both can be rephrased in maximum probability, as shown in the respective papers.

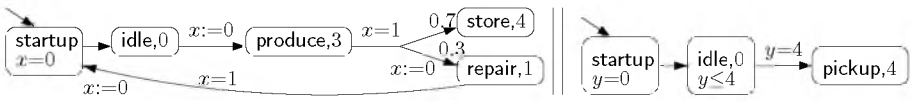

Fig. 6. A Production Plant

All experiments are carried out on a 2GHz PC with 2GB RAM, which is similar to the hardware used in [16]. We do not compare on memory use as these statistics from the other approaches are unavailable to us. However, for the instance of the case study that is the largest in both states and directions (firewire 100k), FORTUNA needs only 70MB.

6.1 A Production Plant

This case study has been inspired by a case study on a lacquer production plant [19]. Although small, it easily scales up to more elaborate plant models. The PPTA on the left of Figure 6 models a production plant. Initially the system is in the startup location, but goes immediately to the idle location. At some point in time, a scheduler may decide to produce the lacquer. Production takes 1 day and costs 3 credits. With probability 0.7 production succeeds and the lacquer is stored, which costs 4 credits per day. With probability 0.3 production fails; the machine needs to be cleaned, after which production can start again. The PPTA on the right of Figure 6 models the customer. After 4 days the customer will try to pick up the product. The two PPTAs work in parallel: they start at the same time and their clocks work at the same speeds. The CBMR is the maximal probability to reach the locations store and pickup with cost at most 9. Note that leaving a customer waiting also costs 4 credits per day. FORTUNA calculates a maximum probability of 0.91. In order to realize this, the plant scheduler should wait for 1.5 days before starting production.

6.2 CSMA with Energy Constraints

IEEE 802.3 CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) is a protocol to avoid data collision on a single channel. The authors of [18] model CSMA with PTAs and are able to compute the maximal probability that both senders have successfully sent their data within a deadline. In certain settings power consumption is important. Sending data consumes power, and typically when a node is listening to receive data, it consumes more power than in other modes. We build a PPTA from the PTA in [18], due to lack of space we refer to their figures and further explanation. We added the following cost to their model: an instantaneous cost of 50 upon a *send*, cost-rate 1 in the *wait* and *done* locations, and cost-rate 3 in the *transmit* location. All other locations use cost-rate 0. With CBMR we are now able to compute the maximal probability by which both nodes are done, but total power consumption is not more than c_{bound}.

6.3 Experiments on Optimizations

We have presented Optimizations 1–3 and the optimization that uses Hasse diagrams. Optimization 2 is already used in the backward reachability algorithm of [18]. We conjecture this optimization is always useful and did not experiment with turning it off: by excluding more intersections one can save on the number of directions that are added at line 19 *and* the number of zones in the state space.

Also, we conjecture Optimization 3 is always useful. Like explained, extra intermediate states are used in the MDP, but the savings in terms of the number of probabilistic transitions in the MDP is huge. Adhoc tests we ran indicate this. Notice that the extra intermediate states are only added to the MDP and are not part of the state space that is explored backwards.

The benefit of the other optimizations becomes apparent from Table 1. The last column shows the situation when using only use Optimizations 2 and 3. The second column puts Optimization 1 into play. Finally, the first column also adds the use of Hasse diagrams.

The implementation gives no guarantees on the order in which zones are explored. The explored zones determine which directions are present. Therefore when trying to add an intersection, Optimization 2 may or may not suppress this. As a result, the number of zones, directions, and τ-directions may vary between different runs of the algorithm. Each experiment is repeated 10 times. We use the format $a \pm b$ to express the calculated mean a and the calculated standard deviation b, where b has the same significance as the least significant digit of a.

For each optimization level the number of generated states is approximately the same. We have displayed the number of directions for each optimization level, and the number of τ-directions when using Hasse diagrams.

The second column shows a strong reduction in the number of directions, for all case studies except "csma". This results mainly in less memory usage. Also the benefit of using Hasse diagrams is clear. There are some great reductions in the number of directions, as well as in the run-time.

6.4 Comparison to other Approaches

Table 2 compares the performance of FORTUNA to the *game-based verification* approach of [16], and the *backwards reachability* approach of [18]. The statistics are take from [16]. The probabilities computed by FORTUNA, as shown in the last column, vary slightly from those results on the larger instances. This is a result of rounding errors.

Uppaal-Pro is another tool for checking maximal reachability on PTA, available from [12]. Since at the time of writing Uppaal-Pro is still in its development phase, we have not included it in our comparison. The *backwards reachability* approach is included in the comparison because it is closest to our approach in its workings. From [16], we see the *digital clocks* approach of [17], performs worse in all the instances.

Table 1. Performance statistics of the optimizations

Case study (parameters) [min /max]		Optimization 1 and Hasse diagrams			Optimization 1		only Optimizations 2 and 3	
		Dirs.	τ-dirs.	Time (s)	Dirs.	Time (s)	Dirs.	Time (s)
csma_cost	8k	947±0	956±0	**1.831±26**	1538±20	2.873±28	2357±29	3.041±35
(c_bound)	9k	1697±1	1795±0	**4.388±32**	2780±27	12.026±63	5283±140	12.98±11
[max]	10k	2754±0	2922±0	**11.08±16**	5277±199	62.684±21	12589±250	67.56±35
	11k	4223±0	4662±0	**26.59±18**	9921±543	271.6±11	28464±1004	296.1±103
csma	2 4	290±0	172±0	0.270±7	360±0	**0.265±7**	374±0	0.267±9
(max_backoff	2 8	742±0	476±0	0.735±8	948±0	**0.726±5**	1002±0	0.752±12
collisions)	4 4	1593±0	812±0	2.418±34	1941±0	**2.366±18**	1999±0	2.406±27
[max]	4 8	3273±0	2092±0	7.241±38	4247±0	**7.223±37**	4617±0	7.901±62
csma_abst	1k 1	362±0	309±0	**0.323±19**	574±3	0.459±22	781±1	0.467±14
(deadline	2k 1	602±0	552±0	**0.627±24**	1036±7	0.811±17	1591±5	0.863±15
bcmax)	3k 1	1499±0	1527±0	**3.445±49**	8375±125	6.879±59	10790±5	7.053±82
[min]	1k 2	1298±1	1061±0	**1.885±54**	2263±6	3.99±13	5236±3	4.401±95
	2k 2	2955±0	2737±0	**7.75±18**	5326±52	13.92±18	13576±27	17.03±11
	3k 2	5298±0	5354±0	**46.14±44**	25492±425	83.68±87	51856±123	102.8±10
firewire_abst	5k	102±0	52±0	**0.024±5**	220±2	0.029±6	290±0	0.034±5
(deadline)	10k	276±0	169±0	**0.081±6**	1284±0	0.192±6	1727±0	0.216±5
[min]	20k	946±0	629±0	**0.587±8**	14864±2	4.135±16	18694±2	4.610±20
	100k	20884±0	14516±0	**221.8±11**	> 1 hour		> 1 hour	
nrp_malicious	5	244±3	168±5	**0.141±9**	312±3	0.192±10	567±15	0.204±12
(deadline)	10	654±12	478±5	**0.711±14**	997±5	1.480±24	2300±25	1.629±29
[max]	20	1436±27	1107±4	**3.135±73**	2569±14	12.043±45	7404±37	13.07±10

As FORTUNA uses the backward reachability approach with new optimizations, it improves on the latter. For all instances FORTUNA is faster than game-based verification, often several orders of magnitude. Why FORTUNA out-performs game-based verification is hard to say, as both approaches are very different in nature. However, we see the following possible reasons:

- Like backward reachability, FORTUNA does not calculate the difference between two zones, but only intersections. As a result, the number of states is much smaller, as can be seen in the table.
- FORTUNA does forward exploration of the reachable state space.
- FORTUNA uses the efficient Parma Polyhedra Library [2] to do operations on zones.
- FORTUNA has been implemented in C++, but we do not know the implementation language for the other tools.

7 Conclusion

We have presented FORTUNA, the first tool for model checking PPTA. FORTUNA is able to compute CBMR. It uses novel optimizations that drastically improve the backward reachability algorithm. Although FORTUNA is more general, it outperforms existing tools for PTAs by several orders of magnitude on a number of case studies in computing maximal probabilistic reachability.

Users of FORTUNA enter models as hard-wired C++ code, using calls to an interface. Although this interface is quite clear, a user-interface is preferable. To increase useablity, the actual policy and the traces it generates should be given as feedback to to the user. Another interesting feature would be to output the

Table 2. Performance statistics and comparisons

Case study (parameters) [min /max]		FORTUNA States	FORTUNA Time (s)	Game-based verification [16] States	Game-based verification [16] Time (s)	Backwards reachability [18] States	Backwards reachability [18] Time (s)	Min/Max reachability probability
csma	2 4	224±0	**0.270±7**	6,476	3.9	243	20.7	0.143555
(max_backoff	2 8	572±0	**0.735±8**	18,196	8.9	575	77.8	0.00525932
collisions)	4 4	1082±0	**2.418±34**	34,826	20.5	303	1443.7	0.0769043
[max]	4 8	2315±0	**7.241±38**	239,298	431.4	> 1 hour		1.65363e-5
csma_abst	1k	254±0	**0.323±19**	6,392	1.9	366	68.2	0.0
(deadline)	2k	437±0	**0.627±24**	24,173	20.7	722	367.8	0.869791
[min]	3k	1178±0	**3.445±49**	79,608	448.0	1,736	1436.3	0.999820099
firewire_abst	5k	64±0	**0.024±5**	205	0.25	63	2.45	0.78125
(deadline)	10k	181±0	**0.081±6**	1,023	1.76	180	3.8	0.9747314
[min]	20k	641±0	**0.587±8**	9,059	26.1	640	26.4	0.999629555
nrp_malicious	5	123±2	**0.141±9**	1,663	1.5	75	2.9	0.100072
(deadline)	10	293±2	**0.711±14**	8,080	11.1	408	117.3	0.105447
[max]	20	632±2	**3.135±73**	49,622	218.1	1,108	1606.5	0.105658

probability at each depth of exploration. This sequence of probabilities is non-decreasing. The algorithm may be stopped when the outcome is large enough compared to some objective. In this iterative approach at each iteration one can benefit from the probabilities calculated in the previous iteration. In case of model checking PTAs, zones can be represented by DBMs (difference bounded matrices), see [6]. These allow faster operations on them than the more general convex polyhedra we use. Thus for this special sub-problem the use of a DBM library may improve performance drastically for PTA models.

References

1. R. Alur, S. L. Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In M. D. D. Benedetto and A. L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.
2. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
3. G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *SFM*, volume 3185 of *LNCS*, pages 200–236, Berlin, 2004. Springer.
4. G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In M. D. D. Benedetto and A. L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.
5. G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Performance Evaluation Review*, 32(4):34–40, 2005.
6. J. Bengtsson and W. Yi. Timed automata: semantics, algorithms and tools. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124, Berlin, 2004. Springer.
7. J. Berendsen, T. Chen, and D. N. Jansen. Undecidability of cost-bounded reachability in priced probabilistic timed automata. In J. Chen and S. B. Cooper, edi-

tors, *TAMC*, volume 5532 of *Lecture Notes in Computer Science*, pages 128–137. Springer, 2009.
8. J. Berendsen, B. Gebremichael, F. Vaandrager, and M. Zhang. Formal specification and analysis of zeroconf using Uppaal. *ACM Transactions on Embedded Computing Systems*, 2010. To appear.
9. J. Berendsen, D. N. Jansen, and J.-P. Katoen. Probably on time and within budget: On reachability in priced probabilistic timed automata. In *QEST*, pages 311–322. IEEE Computer Society, 2006.
10. H. C. Bohnenkamp, P. van der Stok, H. Hermanns, and F. W. Vaandrager. Cost-optimization of the ipv4 zeroconf protocol. In *DSN*, pages 531–540. IEEE Computer Society, 2003.
11. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge mathematical textbooks. Cambridge University Press, 2nd edition, 2002.
12. A. Haugstad. http://www.cs.aau.dk/∼arild/uppaal-probabilistic/.
13. R. P. Kurshan. Verification technology transfer. In O. Grumberg and H. Veith, editors, *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 46–64. Springer, 2008.
14. M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.
15. 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.
16. M. Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic games for verification of probabilistic timed automata. In J. Ouaknine and F. W. Vaandrager, editors, *FORMATS*, volume 5813 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2009.
17. M. Z. 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.
18. M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Information and Computation*, 205(7):1027–1077, 2007.
19. A. Mader, H. Bohnenkamp, Y. S. Usenko, D. N. Jansen, J. Hurink, and H. Hermanns. Synthesis and stochastic assessment of cost-optimal schedules. Technical Report TR-CTIT-06-14, Centre for Telematics and Information Technology, University of Twente, Enschede, 2006. http://eprints.eemcs.utwente.nl/2694/.
20. M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, 2005.
21. J. Sproston. Decidable model checking of probabilistic hybrid automata. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-tolerant Systems: ... FTRTFT*, volume 1926 of *LNCS*, pages 31–45, Berlin, 2000. Springer.

A Proofs

A.1 Proof of Lemma 2

Recall that we defined the probability space $(\mathsf{Paths}^{\infty}_{M}, \mathcal{F}, \mathsf{Prob}_{A}(s))$. The σ-field $\mathcal{F}$ is said to be *generated* by the set of cones $\{\mathcal{C}(\omega) \mid \omega \in \mathsf{Paths}^{*}_{M} \wedge \omega^{0} = s\}$.

The set of cones also generates a (conventional) *field* F, where F satisfies the following properties:

- $\mathsf{Paths}^{\infty}_{M} \in F$
- if $W \in F$ then $\mathsf{Paths}^{\infty}_{M} \backslash W \in F$ (closed under complement)
- if $X, Y \in F$ then $X \cup Y \in F$ (closed under *finite* union)

The difference with a σ-field is that a field does require countable unions of elements to be elements.

Lemma 2 can be viewed as proving that the measure $\mathsf{Prob}_A(s)$ equals the measure $\mu(\omega^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega^0]}(\omega^0)(\mathcal{C}(\omega))$, where both measures are defined over the same measurable space $(\mathsf{Paths}^{\infty}_{M}, \mathcal{F})$. From Carathéodory's extension theorem, we may derive that in order to check this equality, it is enough to check that the measures are equal on the elements of F. We may skip elements in $\mathcal{F}$ that are not in F, see e.g. [?].

We first proof the lemma for cones of the form $\mathcal{C}(s \xrightarrow{a,\mu} \omega)$ for any path $(s \xrightarrow{a,\mu} \omega) \in \mathsf{Paths}^{*}_{M}$. Thus for $A(s) = (a, \mu)$ we want to prove:

$$\mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} \omega)) = \mu(\omega^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega^0]}(\omega^0)(\mathcal{C}(\omega))$$

The proof is by induction on the length of ω. For $n = 0$. Then ω is a single state, say r.

$$\begin{aligned}
&\mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} \omega)) = \mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} r)) \\
&= \mathsf{Prob}_A(s)(\mathcal{C}(s)) \cdot \mu(r) && \text{by definition of } \mathsf{Prob} \\
&= \mu(r) && \text{by definition of } \mathsf{Prob} \\
&= \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\mathcal{C}(r)) && \text{by definition of } \mathsf{Prob} \\
&= \mu(\omega^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega^0]}(\omega^0)(\mathcal{C}(\omega))
\end{aligned}$$

Now assume the lemma holds for n. We will proof it also holds for $n+1$. Let $\omega = \omega' \xrightarrow{a',\bar{\mu}} r$.

$$\begin{aligned}
&\mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} \omega)) = \mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} \omega' \xrightarrow{a',\bar{\mu}} r)) \\
&= \mathsf{Prob}_A(s)(\mathcal{C}(s \xrightarrow{a,\mu} \omega')) \cdot \bar{\mu}(r) && \text{by definition of } \mathsf{Prob} \\
&= \mu(\omega'^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega'^0]}(\omega'^0)(\mathcal{C}(\omega')) \cdot \bar{\mu}(r) && \text{by induction} \\
&= \mu(\omega'^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega'^0]}(\omega'^0)(\mathcal{C}(\omega' \xrightarrow{a',\bar{\mu}} r)) && \text{by definition of } \mathsf{Prob} \\
&= \mu(\omega^0) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} \omega^0]}(\omega^0)(\mathcal{C}(\omega))
\end{aligned}$$

What remains is the case of the element $\mathcal{C}(s)$, and the case of elements that are formed from other elements by complement or union. Assume we have

arbitrary $X, Y \in F$, such that the measures are equal on X and on Y.

$$
\begin{aligned}
&\mathsf{Prob}_A(s)(\mathsf{Paths}^{\infty}_M \backslash X) = \mathsf{Prob}_A(s)(\mathsf{Paths}^{\infty}_M) - \mathsf{Prob}_A(s)(X) \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\mathsf{Paths}^{\infty}_M) \\
&\quad - \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} r) \in X\}) && \text{by induction} \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} r) \in \mathsf{Paths}^{\infty}_M \backslash X\}) && \text{by induction}
\end{aligned}
$$

And we have:

$$
\begin{aligned}
&\mathsf{Prob}_A(s)(X \cup Y) = \mathsf{Prob}_A(s)((X \backslash Y) \cup Y) \\
&= \mathsf{Prob}_A(s)(X \backslash Y) + \mathsf{Prob}_A(s)(Y) && \text{by union of disjoint sets} \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} r) \in X \backslash Y\}) \\
&\quad + \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} r) \in Y\}) && \text{by induction} \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{Prob}_{A[s \xrightarrow{a,\mu} r]}(r)(\{\omega \mid (s \xrightarrow{a,\mu} r) \in X \cup Y\})
\end{aligned}
$$

The final case is when we have the element $\mathcal{C}(s)$. By definition of $\mathcal{C}(\cdot)$ we have that $\mathcal{C}(s) = \bigcup_{r \in \mathsf{supp}(\mu)} \mathcal{C}(s \xrightarrow{a,\mu} r)$. But this union is proven by the previous cases.

A.2 Proof of Theorem 1

Only result 1 remains to be proven. The proof is very similar to the proof of Proposition 29 in [18], however the length of paths match in $[\![\mathcal{A}]\!]$ and M. Let $[\![\mathcal{A}]\!] = (S, s_{\mathsf{init}}, \mathcal{T})$. Let $(\Sigma, s_{\mathsf{init}}, D)$ be the reachability graph generated by Algorithm 1, thus $\mathit{Visited} = \Sigma$. Let $M = (\Sigma, \sigma_{\mathsf{init}}, T)$ be the MDP generated using Definition 8, i.e. $\mathsf{MDP}(\Sigma, s_{\mathsf{init}}, D) = M$.

We introduce two new notations that improve the readability of the proof. Notice that by Definition 5 each state $s \in S$ is a tuple (l, v, c) such that $l \in L$, $v \in \mathsf{inv}(l)$ and $c \in \mathbb{R}_{\geq 0}$. We overload the $+$ operator and let $s{+}d$ denote the state reached from s after a time transition with delay d. Thus, $s{+}d = (l, v + d, c + d \cdot \dot{\$}(l))$. For any $(l, g, p) \in \mathsf{edges}$ and $f \in \mathsf{supp}(p)$ we know that $f = (R, h, l')$ for some $R \subseteq \mathbb{X}$, $h \in \mathbb{N}$ and $l' \in L$. We abuse notation by using f as a function and let $f(s) = (l', v[R := 0], c + h)$.

The proof needs the following properties:

1. If $\mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(s, \sigma_{\mathsf{goal}}) > 0$ then there exists $\sigma \in \Sigma$ such that $s \in \mathsf{tpre}(\sigma)$.
2. For all $(\sigma, a, \mu) \in T$, if $a = \tau$ and $\mu = \{\rho \mapsto 1\}$, then $\sigma \subseteq \rho$.

3. For all $n \in \mathbb{N}, \sigma \in \Sigma, B \in \mathsf{Pol}(M), s \in \mathsf{tpre}(\sigma)$ there exists $A \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$ such that:
$$\mathsf{ProbReach}_A^{\leq n}(s, \sigma_{\mathsf{goal}}) \geq \mathsf{ProbReach}_B^{\leq n}(\sigma, \{\sigma_{\mathsf{goal}}\})$$
4. For all $n \leq \mathsf{maxlength}, s \in S, A \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$, if $\mathsf{SupProbReach}_{\llbracket \mathcal{A} \rrbracket}^{\leq n}(s, \sigma_{\mathsf{goal}}) > 0$, then there exist $i \leq n, \sigma \in \mathit{Waiting}_i, B \in \mathsf{Pol}(M)$ such that $s \in \mathsf{tpre}(\sigma)$ and
$$\mathsf{ProbReach}_B^{\leq n}(\sigma, \{\sigma_{\mathsf{goal}}\}) \geq \mathsf{ProbReach}_A^{\leq n}(s, \sigma_{\mathsf{goal}})$$
5. For all $n \leq \mathsf{maxlength}, A \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$, there exist $B \in \mathsf{Pol}(M)$ such that
$$\mathsf{ProbReach}_B^{\leq n}(\{\sigma_{\mathsf{goal}}\}) \geq \mathsf{ProbReach}_A^{\leq n}(\sigma_{\mathsf{goal}})$$

By Definition 5, $s_{\mathsf{init}} = (l_{\mathsf{init}}, \{x \mapsto 0 \mid x \in \mathbb{X}\}, 0)$. From Algorithm 1 we have that $\sigma_{\mathsf{init}} = \{s_{\mathsf{init}}\}$. Then $\mathsf{tpre}(\sigma_{\mathsf{init}}) = \sigma_{\mathsf{init}}$ by definition of tpre. Thus $s_{\mathsf{init}} \in \mathsf{tpre}(\sigma_{\mathsf{init}})$. Using property 3 and the definition of $\mathsf{SupProbReach}$ we have that:
$$\mathsf{SupProbReach}_{\llbracket \mathcal{A} \rrbracket}^{\leq n}(\sigma_{\mathsf{goal}}) \geq \mathsf{SupProbReach}_M^{\leq n}(\{\sigma_{\mathsf{goal}}\})$$
Using property 5 and the definition of $\mathsf{SupProbReach}$ we have that:
$$\mathsf{SupProbReach}_{\llbracket \mathcal{A} \rrbracket}^{\leq n}(\sigma_{\mathsf{goal}}) \leq \mathsf{SupProbReach}_M^{\leq n}(\{\sigma_{\mathsf{goal}}\})$$
Combining the two inequations above concludes the proof.

Proof of property 1 If $\mathsf{SupProbReach}_{\llbracket \mathcal{A} \rrbracket}^{\leq n}(s, \sigma_{\mathsf{goal}}) > 0$ then there exists a finite path $\omega \in \mathsf{Paths}_{\llbracket \mathcal{A} \rrbracket}^*$ such that $\omega^0 = s$, $|\omega| \leq n$, and $\mathrm{last}(\omega) \in \sigma_{\mathsf{goal}}$. By induction on n and the definition of dpre and tpre, we can conclude there exists a path $\omega \in \mathsf{Paths}_M^*$ such that $|\omega| \leq n$, $s \in \mathsf{tpre}(\omega^0)$, and $\mathrm{last}(\omega) = \sigma_{\mathsf{goal}}$.

Proof of property 2 For any $(\sigma, a, \mu) \in T$ with $a = \tau$ and $\mu = \{\rho \mapsto 1\}$, by Definition 8, we have that $(\sigma, \tau, \rho) \in D$. The proof is by induction on n, which represents the number of directions in D that use action τ. For $n = 2$, from Algorithm 1, we can see all directions are added as a result of line 6, and clearly $\sigma \subseteq \rho$.

Now assume the property holds for some $n \geq 2$. We will proof it also holds for $n + 1$. The last direction that was added to D and uses action τ must have been added on line 19. From lines 17–19 we see that it was added as a result of an existing direction that uses action τ. From line 19 we see that $\sigma \subseteq \rho$ holds.

Proof of property 3 The proof is by induction on n. Now look at the case when $n = 0$. By definition of $\mathsf{ProbReach}$, two cases have to be considered.

- If $\mathsf{ProbReach}_B^{\leq n}(\sigma, \{\sigma_{\mathsf{goal}}\}) = 1$, then $\sigma = \sigma_{\mathsf{goal}}$. From Algorithm 1 we have that $\sigma_{\mathsf{goal}} = \{l_{\mathsf{goal}} \times \mathsf{inv}(l_{\mathsf{goal}}) \times [0, c_{\mathsf{bound}}]\}$. Then $\mathsf{tpre}(\sigma_{\mathsf{goal}}) = \sigma_{\mathsf{goal}}$ by definition of tpre. Since $s \in \mathsf{tpre}(\sigma) = \sigma_{\mathsf{goal}}$ we have $\mathsf{ProbReach}_A^{\leq n}(s, \sigma_{\mathsf{goal}}) = 1$ for any policy $A \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$.

– If $\mathsf{ProbReach}_B^{\leq n}(\sigma, \{\sigma_{\mathsf{goal}}\}) = 0$, then for any policy $A \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$ the inequality holds.

Now suppose that property 3 holds for n. We will proof it also holds for $n+1$. If $\sigma = \sigma_{\mathsf{goal}}$, then the result follows as in the first case for $n = 0$. We are therefore left to consider the case when $\sigma \neq \sigma_{\mathsf{goal}}$. Let $B(\sigma) = (\bar{a}, \bar{\mu})$. By Lemma 3:

$$\mathsf{ProbReach}_B^{\leq n+1}(\sigma, \{\sigma_{\mathsf{goal}}\}) = \sum_{\rho \in \Sigma} \bar{\mu}(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{\bar{a}, \bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \tag{1}$$

When $\bar{a} = \tau$, we have that $\bar{\mu} = \{\rho' \mapsto 1\}$ for some $\rho' \in \Sigma$. Using property 2 we have that $\sigma \subseteq \rho'$. By definition of tpre we have that $s \in \mathsf{tpre}(\rho')$. Thus:

$$\begin{aligned}
\text{RHS of (1)} &= \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{\bar{a}, \bar{\mu}} \rho']}(\rho', \{\sigma_{\mathsf{goal}}\}) \\
&\leq \mathsf{ProbReach}^{\leq n}_{A}(s, \sigma_{\mathsf{goal}}) && \text{for some } A \text{ by induction} \\
&\leq \mathsf{ProbReach}^{\leq n+1}_{A}(s, \sigma_{\mathsf{goal}}) && \text{by Lemma 1}
\end{aligned}$$

Now look at the case when $\bar{a} \neq \tau$. By Definition 8, for some $D_{e,\bar{\mu}}$ with $e = (l, g, p)$ and $\bar{a} = e$ we have that

$$\begin{aligned}
\text{RHS of (1)} &= \sum_{\rho \in \Sigma} \left(\sum_{(\sigma, e, f, \rho) \in D_{e,\bar{\mu}}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e, \bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{(\sigma, e, f, \rho) \in D_{e,\bar{\mu}}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e, \bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \qquad (2)
\end{aligned}$$

Because $s \in \mathsf{tpre}(\sigma)$ there exists a time transition $s \xrightarrow{d} s{+}d$ in $\llbracket \mathcal{A} \rrbracket$, with $s{+}d \in \sigma$. From Algorithm 1 we can see that $(\sigma, e, f, \rho) \in D_{e,\bar{\mu}}$ implies that $\sigma \subseteq \mathsf{dpre}_{e,f}(\mathsf{tpre}(\rho))$. By definition of dpre we have that $f(s{+}d) \in \mathsf{tpre}(\rho)$ and $s{+}d \models v$. Thus, by induction for any $(\sigma, e, f, \rho) \in D_{e,\bar{\mu}}$ there exists a policy $A^f \in \mathsf{Pol}(\llbracket \mathcal{A} \rrbracket)$ such that:

$$\mathsf{ProbReach}^{\leq n}_{A^f}(f(s{+}d), \sigma_{\mathsf{goal}}) \geq \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e, \bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\})$$

Let A be the policy such that:

– $A(s) = (d, \mu)$, where for any $r = (l', v', c') \in S$:

$$\mu(r) = \sum_{\substack{R \subseteq \mathbb{X} \text{ s.t.} \\ v' = (v+d)[R:=0]}} p(R, c' - d \cdot \dot{\$}(l) - c, l') = \sum_{\substack{f \in \mathsf{supp}(p) \text{ s.t.} \\ f(s+d) = r}} p(f)$$

The probabilistic transition $s \xrightarrow{d,\mu}$ exists due to Definition 5.

– For any $(\sigma, e, f, \rho) \in D_{e,\bar{\mu}}$:

$$A[s \xrightarrow{d,\mu} f(s{+}d)] = A^f$$

Now we are able to complete the proof of property 3 as follows:

$$
\begin{aligned}
&\mathsf{ProbReach}^{\leq n+1}_{B}(\sigma, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{(\sigma,e,f,\rho)\in D_{e,\bar{\mu}}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e,\bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) && \text{by (1) and (2)} \\
&\leq \sum_{(\sigma,e,f,\rho)\in D_{e,\bar{\mu}}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A^f}(f(s{+}d), \sigma_{\mathsf{goal}}) && \text{by induction} \\
&\leq \sum_{f\in\mathsf{supp}(p)} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A^f}(f(s{+}d), \sigma_{\mathsf{goal}}) \\
&= \sum_{r\in S} \sum_{\substack{f\in\mathsf{supp}(p) \text{ s.t.} \\ f(s+d)=r}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A^f}(r, \sigma_{\mathsf{goal}}) && \text{by rewriting} \\
&= \sum_{r\in S} \left(\sum_{\substack{f\in\mathsf{supp}(p) \text{ s.t.} \\ f(s+d)=r}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} r]}(r, \sigma_{\mathsf{goal}}) && \text{by construction of } A \\
&= \sum_{r\in S} \mu(r) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} r]}(r, \sigma_{\mathsf{goal}}) && \text{by construction of } A \\
&= \mathsf{ProbReach}^{\leq n+1}_{A}(s, \sigma_{\mathsf{goal}}) && \text{by Lemma 3}
\end{aligned}
$$

Proof of property 4 The proof is by induction on n. Now look at the case when $n = 0$. By definition of ProbReach, two cases have to be considered.

- If $\mathsf{ProbReach}^{\leq n}_{A}(s, \sigma_{\mathsf{goal}}) = 1$, then $s \in \sigma_{\mathsf{goal}} \in \mathit{Waiting}_0$. We have $s \in \mathsf{tpre}(\sigma_{\mathsf{goal}})$ by definition of tpre. Now, for arbitrary B we have that $\mathsf{ProbReach}^{\leq n}_{B}(\sigma_{\mathsf{goal}}, \{\sigma_{\mathsf{goal}}\}) = 1$.
- The case $\mathsf{ProbReach}^{\leq n}_{A}(s, \sigma_{\mathsf{goal}}) = 0$. Since we assumed $\mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(s, \sigma_{\mathsf{goal}}) > 0$, by property 1, we have that $s \in \mathsf{tpre}(\sigma)$ for some $\sigma \in \Sigma$. Clearly from the algorithm we see that then for some $i \leq n$ we have that $\sigma \in \mathit{Waiting}_i$. Now for any policy $B \in \mathsf{Pol}(M)$ the inequality holds.

Now suppose that property 4 holds for n. We will prove it also holds for $n + 1 \leq \mathsf{maxlength}$. If $\mathsf{ProbReach}^{\leq n+1}_{A}(s, \sigma_{\mathsf{goal}}) = 0$, then the result follows as in the second case for $n = 0$. We are therefore left to consider the case when

$$\mathsf{ProbReach}^{\leq n+1}_{A}(s, \sigma_{\mathsf{goal}}) > 0 \tag{3}$$

If $s \in \sigma_{\mathsf{goal}}$, then the result follows as in the first case for $n = 0$. We are therefore left to consider the case when $s \notin \sigma_{\mathsf{goal}}$. Let $A(s) = (d, \mu)$. By Lemma 3:

$$\mathsf{ProbReach}^{\leq n+1}_{A}(s, \sigma_{\mathsf{goal}}) = \sum_{r\in S} \mu(r) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} r]}(r, \sigma_{\mathsf{goal}}) \tag{4}$$

By Definition 5 we can distinguish two cases.

1. A chooses a time transition, so $\mu = \{r \mapsto 1\}$ for some $r \in S$, then

$$\text{RHS of (4)} = \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} r]}(r, \sigma_{\mathsf{goal}}) \tag{5}$$

Using assumption (3) we conclude $\mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(r, \sigma_{\mathsf{goal}}) > 0$. Now, by induction, the exists $i \leq n$, $\sigma \in \mathit{Waiting}_i$, $B \in \mathsf{Pol}(M)$ such that $r \in \mathsf{tpre}(\sigma)$ and

$$\text{RHS of (5)} \leq \mathsf{ProbReach}^{\leq n}_{B}(\sigma, \{\sigma_{\mathsf{goal}}\})$$

Clearly $s \in \mathsf{tpre}(\sigma)$, and using Lemma 1 we are done.

2. A chooses a delayed discrete (probabilistic) transition. By Definition 5:

$$\begin{aligned}
&\text{RHS of (4)} = \\
&\sum_{r \in S} \left(\sum_{\substack{f \in \mathsf{supp}(p) \text{ s.t.} \\ f(s+d)=r}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} r]}(r, \sigma_{\mathsf{goal}}) \\
&= \sum_{f \in \mathsf{supp}(p)} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d), \sigma_{\mathsf{goal}}) \qquad (6)
\end{aligned}$$

Now consider any $f \in \mathsf{supp}(p)$ such that $\mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d), \sigma_{\mathsf{goal}}) > 0$. By induction there exists $i \leq n$, a zone $\rho^f \in \mathit{Waiting}_i$, and a policy $B^f \in \mathsf{Pol}(M)$ such that $f(s+d) \in \mathsf{tpre}(\rho^f)$ and

$$\mathsf{ProbReach}^{\leq n}_{B^f}(\rho^f, \{\sigma_{\mathsf{goal}}\}) \geq \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d), \sigma_{\mathsf{goal}}) \tag{7}$$

Let $\sigma^f = \mathsf{dpre}_{e,f}(\mathsf{tpre}(\rho^f))$. By definition of dpre we have that $s+d \in \sigma^f$. By lines 13–15 of Algorithm 1 we have that $\sigma^f \in \mathit{Waiting}_{i+1}$ and $(\sigma^f, e, f, \rho^f) \in D$. On lines 17–21 of the algorithm the following zone wil be constructed:

$$\sigma = \bigcap \{\sigma^f \mid f \in \mathsf{supp}(p) \wedge \mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(f(s+d), \sigma_{\mathsf{goal}}) > 0\}$$

Clearly $s+d \in \sigma$, thus $s \in \mathsf{tpre}(\sigma)$. Moreover, the following directions are constructed:

$$D' = \{(\sigma, e, f, \rho^f) \mid f \in \mathsf{supp}(p) \wedge \mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(f(s+d), \sigma_{\mathsf{goal}}) > 0\}$$

Now construct $\bar{\mu}$ using Definition 8, where $D_{e,\bar{\mu}} \supseteq D'$. Let $B \in \mathsf{Pol}(M)$ such that $B(\sigma) = (e, \bar{\mu})$ and $B[\sigma \xrightarrow{e,\bar{\mu}} \rho^f] = B^f$. We are now able to finish the

proof as follows:

$$
\begin{aligned}
&\mathsf{ProbReach}_B^{\leq n+1}(\sigma, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{\rho \in \Sigma} \bar{\mu}(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e,\bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) && \text{by Lemma 3} \\
&= \sum_{\rho \in \Sigma} \left(\sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\bar{\mu}}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e,\bar{\mu}} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
& && \text{by construction of } \bar{\mu} \text{ in Definition 8} \\
&\geq \sum_{(\sigma,(l,g,p),f,\rho^f) \in D'} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma \xrightarrow{e,\bar{\mu}} \rho^f]}(\rho^f, \{\sigma_{\mathsf{goal}}\}) && \text{by construction of } D_{e,\bar{\mu}} \\
&= \sum_{(\sigma,(l,g,p),f,\rho^f) \in D'} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B^f}(\rho^f, \{\sigma_{\mathsf{goal}}\}) && \text{by construction of } B \\
&\geq \sum_{(\sigma,(l,g,p),f,\rho^f) \in D'} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d), \sigma_{\mathsf{goal}}) && \text{by (7)} \\
&= \sum_{\substack{f \in \mathsf{supp}(p) \text{ s.t.} \\ \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d),\sigma_{\mathsf{goal}}) > 0}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{d,\mu} f(s+d)]}(f(s+d), \sigma_{\mathsf{goal}}) \\
& && \text{by construction of } D' \\
&= \mathsf{ProbReach}_A^{\leq n+1}(s, \sigma_{\mathsf{goal}}) && \text{by (6) and (4)}
\end{aligned}
$$

Proof of property 5 The proof is by induction on n. Now look at the case when $n = 0$. By definition of ProbReach, two cases have to be considered.

- If $\mathsf{ProbReach}_A^{\leq n}(\sigma_{\mathsf{goal}}) = 1$, then $s_{\mathsf{init}} \in \sigma_{\mathsf{goal}} \in \mathit{Waiting}_0$. But then $\sigma_{\mathsf{init}} \subseteq \sigma_{\mathsf{goal}}$. By line 4 of Algorithm 1: $\sigma_{\mathsf{init}} = \sigma_{\mathsf{goal}}$. Now, for arbitrary B we have that $\mathsf{ProbReach}_B^{\leq n}(\{\sigma_{\mathsf{goal}}\}) = 1$.
- If $\mathsf{ProbReach}_A^{\leq n}(\sigma_{\mathsf{goal}}) = 0$, for any policy $B \in \mathsf{Pol}(M)$ the inequality holds.

Now suppose that property 5 holds for n. We will prove it also holds for $n + 1 \leq \mathsf{maxlength}$. By Definition 5, we can distinguish two cases.

- A chooses a time transition, then from Algorithm 1 we know that $\mathsf{inv}(l_{\mathsf{init}}) = \bigwedge_{x \in \mathbb{X}}(x = 0)$. Therefore only the time transition with zero delay is possible from s_{init}, and by Lemma 3 we have the following:

$$\mathsf{ProbReach}_A^{\leq n+1}(\sigma_{\mathsf{goal}}) = \mathsf{ProbReach}^{\leq n}_{A[s_{\mathsf{init}} \xrightarrow{0} s_{\mathsf{init}}]}(\sigma_{\mathsf{goal}}) \tag{8}$$

By induction there exists $B \in \mathsf{Pol}(M)$ such that

$$
\begin{aligned}
&\text{RHS of (8)} \leq \mathsf{ProbReach}_B^{\leq n}(\{\sigma_{\mathsf{goal}}\}) \\
&\leq \mathsf{ProbReach}_B^{\leq n+1}(\{\sigma_{\mathsf{goal}}\}) && \text{by Lemma 1}
\end{aligned}
$$

– A chooses a delayed discrete (probabilistic) transition. If $\mathsf{ProbReach}_A^{\leq n+1}(\sigma_{\mathsf{goal}}) = 0$ the inequality holds for any policy $B \in \mathsf{Pol}(M)$. Now, assume $\mathsf{ProbReach}_A^{\leq n+1}(\sigma_{\mathsf{goal}}) > 0$. This implies $\mathsf{SupProbReach}_{[\![\mathcal{A}]\!]}^{\leq n+1}(s_{\mathsf{init}}, \sigma_{\mathsf{goal}}) > 0$. By property 4, there exists $i \leq n+1$, $\sigma \in \mathit{Waiting}_i$, $B' \in \mathsf{Pol}(M)$ such that $s_{\mathsf{init}} \in \mathsf{tpre}(\sigma)$ and

$$\mathsf{ProbReach}_A^{\leq n+1}(\sigma_{\mathsf{goal}}) \leq \mathsf{ProbReach}_{B'}^{\leq n+1}(\sigma, \{\sigma_{\mathsf{goal}}\}) \tag{9}$$

From Algorithm 1 we know that $\mathsf{inv}(l_{\mathsf{init}}) = \bigwedge_{x \in \mathbb{X}}(x = 0)$. Therefore $\mathsf{tpre}(\sigma) = \sigma$ and $s_{\mathsf{init}} \in \sigma$. In case $\sigma = \sigma_{\mathsf{goal}}$, we have that $s_{\mathsf{init}} \in \sigma_{\mathsf{goal}}$, and the result follows as in the first case for $n = 0$. We are therefore left to consider the case when $\sigma \notin \{\sigma_{\mathsf{goal}}\}$. Let $B'(\sigma) = (e, \mu')$.

$$\begin{aligned}
&\text{RHS of (9)} \\
&= \sum_{\rho \in \Sigma} \mu'(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{B'[\sigma \xrightarrow{e,\mu'} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) && \text{by Lemma 3} \\
&= \sum_{\rho \in \Sigma} \left(\sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\mu'}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{B'[\sigma \xrightarrow{e,\mu'} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
& && \text{by construction of } \mu' \text{ in Definition 8} \\
&= \sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\mu'}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B'[\sigma \xrightarrow{e,\mu'} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\})
\end{aligned}$$

Because of the invariant $\mathsf{inv}(l_{\mathsf{init}})$, we have that $\sigma_{\mathsf{init}} \subseteq \sigma$. From lines 17–19 of Algorithm 1, we conclude that we can construct $D' \subseteq D$ such that for each $(\sigma, e, f, \rho) \in D_{e,\mu'}$ there exists $(\sigma_{\mathsf{init}}, e, f, \rho) \in D'$. Using Definition 8, we have that $(\sigma_{\mathsf{init}}, e, \mu)$, where $D_{e,\mu} = D'$. Now, we can construct policy $B \in \mathsf{Pol}(M)$ such that $B(\sigma_{\mathsf{init}}) = (e, \mu)$ and $B[\sigma_{\mathsf{init}} \xrightarrow{e,\mu} \rho] = B'[\sigma \xrightarrow{e,\mu'} \rho]$ for any $\rho \in \mathsf{supp}(\mu)$. We are now able to finish the proof as follows:

$$\begin{aligned}
&\sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\mu'}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B'[\sigma \xrightarrow{e,\mu'} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{(\sigma_{\mathsf{init}},(l,g,p),f,\rho) \in D'} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma_{\mathsf{init}} \xrightarrow{e,\mu} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{\rho \in \Sigma} \mu(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{B[\sigma_{\mathsf{init}} \xrightarrow{e,\mu} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \\
&= \mathsf{ProbReach}_B^{\leq n+1}(\{\sigma_{\mathsf{goal}}\}) && \text{by Lemma 3}
\end{aligned}$$

A.3 Proof of Theorem 2

We will need the following lemma.

Lemma 5. *For any two zones $\rho, \sigma \in \Sigma$, when $\rho \supset \sigma$, then either $\rho \supset\!\!- \sigma$ or there exists a zone $\upsilon \in \Sigma$ such that $\rho \supset \upsilon \supset\!\!- \sigma$.*

Proof. See [11].

We will now define a notion of weak simulation on reachability graphs needed for this proof and later proofs. It differs from the traditional notion in that a transition $s \xrightarrow{a} r$ will *not* be simulated by a path $\bar{s} \Rightarrow \xrightarrow{a} \Rightarrow \bar{r}$, but a path $\bar{s} \xrightarrow{a} \Rightarrow \bar{r}$ instead. This limitation is essential to Lemma **??**. The necessity of the limitation is best shown with the following small example.

Assume a probabilistic transition (s, a, μ) in $\mathsf{MDP}(M)$, that is generated by Definition 8 from $D_{e,\mu} = \{(s, e, f_1, r), (s, e, f_2, r)\}$. Without the limitation there may exist paths $\bar{s} \Rightarrow q_1 \xrightarrow{e,f_1} \Rightarrow \bar{r}$ and $\bar{s} \Rightarrow q_2 \xrightarrow{e,f_2} \Rightarrow \bar{r}$, but we are not guaranteed that there exists a distribution with two outcomes, since q_1 and q_2 may differ, thus the probability of reaching $\bar{r}$ from $\bar{s}$ is lower than reaching r from s.

Definition 10 (Weak Simulation). *Given reachability graphs $Q = (S, s_{\mathsf{init}}, D)$ and $\bar{Q} = (\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$, we say that $\bar{Q}$* simulates *Q if there exists a relation $\mathrm{R} \subseteq S \times \bar{S}$ such that*

1. $s_{\mathsf{init}} \,\mathrm{R}\, \bar{s}_{\mathsf{init}}$
2. *if $s \,\mathrm{R}\, \bar{s}$ and $s \xrightarrow{a} r$, then either $a = \tau$ and $r \,\mathrm{R}\, \bar{s}$, or there exists an $\bar{r}$ such that $\bar{s} \xrightarrow{a} \Rightarrow \bar{r}$ and $r \,\mathrm{R}\, \bar{r}$.*

R *is called a* weak simulation *relation.*

Lemma 6. *Given reachability graphs $Q = (S, s_{\mathsf{init}}, D)$, $\bar{Q} = (\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$, and $G \subseteq S$. Assume $\bar{Q}$ simulates Q via* R. *Let $\bar{G} = \{\bar{s} \mid \exists s \in G. s \,\mathrm{R}\, \bar{s}\}$, then:*

1. *For any $A \in \mathsf{Pol}(\mathsf{MDP}(Q))$, there exists $\bar{A} \in \mathsf{Pol}(\mathsf{MDP}(\bar{Q}))$ such that*

$$\mathsf{ProbReach}_{\bar{A}}(\bar{G}) \geq \mathsf{ProbReach}_{A}(G)$$

2.

$$\mathsf{SupProbReach}_{\mathsf{MDP}(\bar{Q})}(\bar{G}) \geq \mathsf{SupProbReach}_{\mathsf{MDP}(Q)}(G)$$

Proof. Result 2 follows in a straightforward manner from result 1. The proof of result 1 follows straightforwardly from the following lemma which is more general. □

Lemma 7. *Given reachability graphs $Q = (S, s_{\mathsf{init}}, T)$, $\bar{Q} = (\bar{S}, \bar{s}_{\mathsf{init}}, \bar{T})$, and $G \subseteq S$. Assume $\bar{Q}$ simulates Q via* R. *Let $\bar{G} = \{\bar{s} \mid \exists s \in G. s \,\mathrm{R}\, \bar{s}\}$. For any $s \in S$, $A \in \mathsf{Pol}(Q)$, $n \in \mathbb{N}$, and $\bar{s} \in \bar{S}$ with $s \,\mathrm{R}\, \bar{s}$, we have that there exists $\bar{A} \in \mathsf{Pol}(\bar{Q})$ such that*

$$\mathsf{ProbReach}_{\bar{A}}(\bar{s}, \bar{G}) \geq \mathsf{ProbReach}^{\leq n}_{A}(s, G)$$

Proof. When $s \in G$, the righthand side equals 1 by definition of ProbReach. By definition of $\bar{G}$, we have that $\bar{s} \in \bar{G}$, thus also the lefthand side equals 1. Now assume $s \notin G$. When $\bar{s} \in \bar{G}$, the lefthand side equals 1 by definition of ProbReach, and the inequality follows trivially. Now assume $\bar{s} \notin \bar{G}$.

The proof is by induction on n. For $n = 0$, the righthand side equals 0 by definition of $\mathsf{ProbReach}$, and the inequality follows trivially.

Now assume the lemma holds for n, we prove it also holds for $n+1$. Let $A(s) = (a, \mu)$. By Lemma 3:

$$\mathsf{ProbReach}_A^{\leq n+1}(s, G) = \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{a,\mu} r]}(r, G) \tag{10}$$

Look at the case when $a \neq \tau$. By Definition 8: $\mu(r) = \sum_{(s,(l,g,p),f,r) \in D_{s,\mu}} p(f)$ for some $D_{e,\mu}$, and $a = e$. By weak simulation, for every $\delta = (s, e, f, r) \in D_{e,\mu}$, there exists $\omega^\delta \in \mathsf{Paths}^*_{\bar{Q}}$ of the form $(\omega^\delta)^0 \Rightarrow \mathrm{last}(\omega^\delta)$ such that $\bar{s} \xrightarrow{e,f} \omega^\delta$ and $r \,\mathrm{R}\, \mathrm{last}(\omega^\delta)$. By induction there exists $A^\delta \in \mathsf{Pol}(\bar{Q})$ such that:

$$\mathsf{ProbReach}_{A^\delta}(\mathrm{last}(\omega^\delta), \bar{G}) \geq \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{e,\mu} r]}(r, G)$$

Let $D' = \{(\bar{s}, e, f, (\omega^\delta)^0) \mid \delta = (s, e, f, r) \in D_{e,\mu}\}$. Now construct $\bar{\mu}$ using Definition 8, where $D_{e,\bar{\mu}} \supseteq D'$. Let $\bar{A} \in \mathsf{Pol}(\bar{M})$ such that $\bar{A}(\bar{s}) = (e, \bar{\mu})$, $\mathsf{Prob}_{\bar{A}}(\bar{s})(\mathcal{C}(\bar{s} \xrightarrow{e,\bar{\mu}} \omega^\delta)) = \bar{\mu}((\omega^\delta)^0)$ and $\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} \omega^\delta] = A^\delta$ for any $\delta \in D_{e,\mu}$. Note that when the state $(\omega^\delta)^0$ is reached after resolving the probabilistic choice of $\bar{\mu}$, policy $\bar{A}$ will take all the τ transitions of ω^δ. We conclude the case as

follows:

$$
\begin{aligned}
&\mathsf{ProbReach}_{A}^{\leq n+1}(s, G) \\
&= \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{e,\mu} r]}(r, G) && \text{by (10)} \\
&= \sum_{r \in S} \left(\sum_{(s,(l,g,p),f,r) \in D_{e,\mu}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{e,\mu'} r]}(r, G) && \text{by Definition 8} \\
&= \sum_{(s,(l,g,p),f,r) \in D_{e,\mu}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{e,\mu} r]}(r, G) && \text{by rewriting} \\
&\leq \sum_{\delta = (s,(l,g,p),f,r) \in D_{e,\mu}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A^{\delta}}(\mathrm{last}(\omega^{\delta}), \bar{G}) && \text{by induction} \\
&= \sum_{\delta = (s,(l,g,p),f,r) \in D_{e,\mu}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} (\omega^{\delta})^0]}((\omega^{\delta})^0, \bar{G}) \\
& && \text{by construction of } \bar{A} \\
&= \sum_{(s,(l,g,p),f,\bar{r}) \in D'} p(f) \cdot \mathsf{ProbReach}_{\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} \bar{r}]}(\bar{r}, \bar{G}) && \text{by construction of } D' \\
&\leq \sum_{(s,(l,g,p),f,\bar{r}) \in D_{e,\bar{\mu}}} p(f) \cdot \mathsf{ProbReach}_{\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} \bar{r}]}(\bar{r}, \bar{G}) && \text{since } D_{e,\bar{\mu}} \supseteq D' \\
&= \sum_{\bar{r} \in \bar{S}} \left(\sum_{(s,(l,g,p),f,r) \in D_{e,\bar{\mu}}} p(f) \right) \cdot \mathsf{ProbReach}_{\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} \bar{r}]}(\bar{r}, \bar{G}) && \text{by rewriting} \\
&= \sum_{\bar{r} \in \bar{S}} \bar{\mu}(\bar{r}) \cdot \mathsf{ProbReach}_{\bar{A}[\bar{s} \xrightarrow{e,\bar{\mu}} \bar{r}]}(\bar{r}, \bar{G}) && \text{by construction of } \bar{\mu} \text{ in Definition 8} \\
&= \mathsf{ProbReach}_{\bar{A}}(\bar{s}, \bar{G}) && \text{by Lemma 3}
\end{aligned}
$$

In case $a = \tau$, we have that $\mu = \{r \mapsto 1\}$ for some $r \in S$.

$$\text{RHS of (10)} = \mathsf{ProbReach}^{\leq n}_{A[s \xrightarrow{a,\mu} r]}(r, G) \tag{11}$$

By induction there exists $\bar{A} \in \mathsf{Pol}(\bar{M})$ such that

$$\mathsf{ProbReach}_{\bar{A}}(\bar{s}, \bar{G}) \geq \text{RHS of (11)}$$

and we are done. □

Assume that $(S, s_{\mathsf{init}}, D)$ and $(\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$ are the reachability graphs generated by the original and new algorithm, respectively. Assume that $M = (S, s_{\mathsf{init}}, T)$ and $\bar{M} = (\bar{S}, \bar{s}_{\mathsf{init}}, \bar{T})$ are the MDPs generated on the output of the original and new algorithm, respectively, using Definition 8. Note that $\bar{s}_{\mathsf{init}} = s_{\mathsf{init}}$.

We will prove the following equation, from which the theorem follows straightforwardly.

$$\mathsf{SupProbReach}_{\bar{M}}(\{\sigma_{\mathsf{goal}}\}) = \mathsf{SupProbReach}_{M}(\{\sigma_{\mathsf{goal}}\})$$

We have the following equation:

$$\bar{D} = \{(\sigma, e, f, \rho) \in D \mid \nexists (\sigma, e, f, \rho') \in D.\rho' \subset \rho\} \cup \\ \{(\sigma, (l, \mathsf{true}, p), \emptyset, \rho) \in D \mid \rho \supset\!\!- \sigma \wedge (\sigma \text{ uses } l) \wedge p = \{(l, \emptyset) \mapsto 1\}\} \quad (12)$$

The second part of the union follows directly from Optimization 1 Fragment B. Since Fragment B is executed only at the very end, we see that before executing Fragment B, $\bar{D}$ should be equal to the first part of the union. This follows directly from Optimization 1 Fragment A that is executed after each added edge.

It follows directly from Optimization 1 that $\bar{S} = S$. We define relations $\mathrm{R}_1, \mathrm{R}_2 :\subseteq S \times S$ as follows: $\mathrm{R}_1 = \{(s, \bar{s}) \mid s = \bar{s}\}$ and $\mathrm{R}_2 = \{(\bar{s}, s) \mid \bar{s} \supseteq s\}$. We will prove $(\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$ simulates $(S, s_{\mathsf{init}}, D)$ via R_1, and $(S, s_{\mathsf{init}}, D)$ simulates $(\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$ via R_2, both non-probabilistically in the sense of Definition 10. The theorem then follows straightforwardly by Lemma **??**.

Clearly $s_{\mathsf{init}} \,\mathrm{R}_1\, \bar{s}_{\mathsf{init}}$. Assume $s \,\mathrm{R}_1\, \bar{s}$ and $s \xrightarrow{a} r$. Note that $s = \bar{s}$ by construction of R_1. In case this direction is also present in $\bar{D}$ condition 2 holds. In case this direction is not present in $\bar{D}$ this is as a result of Fragment A. Fragment A may delete multiple directions, but in the end, for each deleted direction $s \xrightarrow{a} r$ there will be a direction $s \xrightarrow{a} r'$ with $r' \subset r$. Now, by Lemma 5, Fragment B makes sure there is a path of directions $r' \Rightarrow r$ in $\bar{D}$. Then condition 2 holds.

Clearly $\bar{s}_{\mathsf{init}} \,\mathrm{R}_2\, s_{\mathsf{init}}$. Assume $\bar{s} \,\mathrm{R}_2\, s$ and $\bar{s} \xrightarrow{a} \bar{r}$. In case this direction was not added as a result of Fragment B, we know that it also exists in D. Note that $\bar{s} \supseteq s$ by construction of R_2. Because $\bar{s} \cap s = s \neq \emptyset$ on line 18, and due to line 19 there will be a direction $s \xrightarrow{a} r$, with $r = \bar{r}$. Since $\bar{r} \,\mathrm{R}_2\, r$ we are done. In case $\bar{s} \xrightarrow{a} \bar{r}$ was added as a result of Fragment B, we know that $a = \tau$ and $\bar{r} \supset\!\!- \bar{s}$. But then $\bar{r} \supseteq s$, which implies $\bar{r} \,\mathrm{R}_2\, s$ and we are done.

A.4 Proof of Theorem 3

Assume $D, \bar{D}$ are the directions generated by the original and new algorithm, respectively. Assume $M = (\Sigma, \sigma_{\mathsf{init}}, T)$ and $\bar{M} = (\bar{\Sigma}, \bar{\sigma}_{\mathsf{init}}, \bar{T})$ are the MDPs generated on the output of the original and new algorithm, respectively, using Definition 8. Note that $\bar{\sigma}_{\mathsf{init}} = \sigma_{\mathsf{init}}$.

We need the following property. For any $\sigma \in \Sigma$ and $A \in \mathsf{Pol}(M)$, there exists $\bar{\sigma} \in \bar{\Sigma}$ and $\bar{A} \in \mathsf{Pol}(\bar{M})$ such that $\bar{\sigma} \supseteq \sigma$ and

$$\mathsf{ProbReach}^{\leq n}_{\bar{A}}(\bar{\sigma}, \{\sigma_{\mathsf{goal}}\}) = \mathsf{ProbReach}^{\leq n}_{A}(\sigma, \{\sigma_{\mathsf{goal}}\})$$

Using this property we conclude there exists $\bar{\sigma} \supseteq \sigma_{\mathsf{init}}$ and $\bar{A} \in \mathsf{Pol}(\bar{M})$ such that

$$\mathsf{ProbReach}^{\leq n}_{\bar{A}}(\bar{\sigma}, \{\sigma_{\mathsf{goal}}\}) = \mathsf{ProbReach}^{\leq n}_{A}(\{\sigma_{\mathsf{goal}}\})$$

It follows that the condition on the intersection of line 18' holds for the intersection $\bar{\sigma} \cap \sigma_{\mathsf{init}}$. Since $\bar{\sigma} \cap \sigma_{\mathsf{init}} = \sigma_{\mathsf{init}}$, by line 19 for every $(\sigma, e, f, \rho) \in D$, if $\sigma = \bar{\sigma}$, then also $(\sigma_{\mathsf{init}}, e, f, \rho) \in D$. Let $\bar{A}(\bar{\sigma}) = (a, \mu)$. By Definition 8: μ is defined by some $D_{e,\mu}$. Let $D_{e,\mu'} = \{(\sigma_{\mathsf{init}}, e, f, \rho) \mid (\bar{\sigma}, e, f, \rho) \in D_{e,\mu}\}$, which by Definition 8 defines μ'. Let $A' \in \mathsf{Pol}(M)$ be the policy such that $A'(\sigma_{\mathsf{init}}) = (a, \mu')$ and

$A'[\sigma_{\mathsf{init}} \xrightarrow{a,\mu'} \rho] = A[\bar{\sigma} \xrightarrow{a,\mu} \rho]$ for any $\rho \in \bar{\Sigma}$. Clearly we can conclude the proof with

$$\mathsf{ProbReach}^{\leq n}_{A'}(\{\sigma_{\mathsf{goal}}\}) = \mathsf{ProbReach}^{\leq n}_{\bar{A}}(\bar{\sigma}, \{\sigma_{\mathsf{goal}}\})$$

Proof of the property The proof is by induction on n. If $n = 0$ and $\sigma = \sigma_{\mathsf{goal}}$ then $\mathsf{ProbReach}_A(\sigma, \{\sigma_{\mathsf{goal}}\}) = 1$. We can choose $\bar{\sigma} = \sigma_{\mathsf{goal}}$ since $\sigma_{\mathsf{goal}} \in \bar{\Sigma}$, thus we are done. For $n = 0$ and $\sigma \neq \sigma_{\mathsf{goal}}$ then $\mathsf{ProbReach}_A(\sigma, \{\sigma_{\mathsf{goal}}\}) = 0$ and we are done.

Now assume the property holds for n. We will prove it will also holds for $n+1$. In case $\sigma = \sigma_{\mathsf{goal}}$, the proof is completed as for $n = 0$. Now assume $\sigma \neq \sigma_{\mathsf{goal}}$. Let $A(\sigma) = (a, \mu)$. By Lemma 3:

$$\mathsf{ProbReach}^{\leq n+1}_{A}(\sigma, \{\sigma_{\mathsf{goal}}\}) = \sum_{\rho \in \Sigma} \mu(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{A[\sigma \xrightarrow{a,\mu} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\}) \qquad (13)$$

By Definition 8: $\forall \rho \in \Sigma.\mu(\rho) = \sum_{(\sigma,e,f,\rho) \in D_{e,\mu}} p(f)$ for some $D_{e,\mu} \subseteq D$, where $e = (l, g, p)$. By induction, for every $(\sigma, e, f, \rho) \in D_{e,\mu}$ there exists $\bar{\rho} \in \bar{\Sigma}$ and $A_{\bar{\rho}} \in \mathsf{Pol}(\bar{M})$ such that $\bar{\rho} \supseteq \rho$ and

$$\mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) = \mathsf{ProbReach}^{\leq n}_{A[\sigma \xrightarrow{a,\mu} \rho]}(\rho, \{\sigma_{\mathsf{goal}}\})$$

Zone $\bar{\sigma} = \mathsf{dpre}_{e,f}(\mathsf{tpre}(\bar{\rho}))$ will be generated by line 15 since $\bar{\rho} \in \bar{\Sigma}$, thus $(\bar{\sigma}, e, f, \bar{\rho}) \in \bar{D}$. From the definitions of dpre and tpre it is easy to see that $\bar{\sigma} \supseteq \sigma$.

Let $D' \subseteq \bar{D}$ be such that for every $(\sigma, e, f, \rho) \in D_{e,\mu}$ there exists a $(\bar{\sigma}, e, f, \bar{\rho}) \in D'$ as described above. Now let $\sigma' = \bigcap_{(\bar{\sigma},e,f,\bar{\rho}) \in D'} \bar{\sigma}$. And let $D_{e,\bar{\mu}} = \{(\sigma', e, f, \bar{\rho}) \mid \exists \bar{\sigma}.(\bar{\sigma}, e, f, \bar{\rho}) \in D'\}$. From $D_{e,\mu}$ by Definition 8 the new condition in line 18' will hold for each pair of elements of $D_{e,\bar{\mu}}$. By Definition 8 we have $(\sigma', \bar{\mu}) \in \bar{T}$.

Now let $\bar{A}$ be the policy such that $\bar{A}(\sigma') = (a, \bar{\mu})$ and $\bar{A}(\sigma' \xrightarrow{e,f,\bar{\mu}} \bar{\rho}) = A_{\bar{\rho}}(\bar{\rho})$. Now putting everything together:

$$
\begin{aligned}
\text{RHS of (13)} &= \sum_{\rho \in \Sigma} \mu(\rho) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) && \text{by induction} \\
&= \sum_{\rho \in \Sigma} \left(\sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\mu}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) && \text{by Definition 8} \\
&= \sum_{(\sigma,(l,g,p),f,\rho) \in D_{e,\mu}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{(\sigma',(l,g,p),f,\bar{\rho}) \in D_{e,\bar{\mu}}} p(f) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) && \text{by construction of } D_{e,\bar{\mu}} \\
&= \sum_{\bar{\rho} \in \bar{\Sigma}} \left(\sum_{(\sigma',(l,g,p),f,\bar{\rho}) \in D_{e,\bar{\mu}}} p(f) \right) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) \\
&= \sum_{\bar{\rho} \in \bar{\Sigma}} \bar{\mu}(\bar{\rho}) \cdot \mathsf{ProbReach}^{\leq n}_{A_{\bar{\rho}}}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) && \text{by Definition 8} \\
&= \sum_{\bar{\rho} \in \bar{\Sigma}} \bar{\mu}(\bar{\rho}) \cdot \mathsf{ProbReach}^{\leq n}_{\bar{A}[\bar{\sigma} \to \rho]}(\bar{\rho}, \{\sigma_{\mathsf{goal}}\}) && \text{by definition of } \bar{A} \\
&= \mathsf{ProbReach}^{\leq n+1}_{\bar{A}}(\bar{\sigma}, \{\sigma_{\mathsf{goal}}\}) && \text{by Lemma 3}
\end{aligned}
$$

A.5 Proof of Theorem 4

Let $Q = (S, s_{\mathsf{init}}, D)$ and $\bar{Q} = (\bar{S}, \bar{s}_{\mathsf{init}}, \bar{D})$. Let $M = \mathsf{MDP}(Q)$ and $\bar{M} = \mathsf{MDP}(\bar{Q})$. By definition of Optimization 3: $\bar{S} = S \cup I$ and $\bar{s}_{\mathsf{init}} = s_{\mathsf{init}}$.

We will prove the following two properties, from which the theorem follows straightforwardly.

1. For any $s \in S$, $G \subseteq S$, and $A \in \mathsf{Pol}(M)$, there exists $\bar{A} \in \mathsf{Pol}(\bar{M})$ such that
$$\mathsf{ProbReach}_A(s, G) \leq \mathsf{ProbReach}_{\bar{A}}(s, G)$$
2. For any $n \in \mathbb{N}$, $s \in S$, $G \subseteq S$, and $\bar{A} \in \mathsf{Pol}(\bar{M})$, there exists $A \in \mathsf{Pol}(M)$ such that
$$\mathsf{ProbReach}^{\leq n}_{\bar{A}}(s, G) \leq \mathsf{ProbReach}_A(s, G)$$

Proof of property 1 We define a relation $\mathrm{R} \subseteq S \times \bar{S}$ as follows: $\mathrm{R} = \{(s, \bar{s}) \mid s = \bar{s}\}$. We will prove $\bar{Q}$ simulates Q via R non-probabilistically in the sense of Definition 10. The property then follows straightforwardly by Lemma ??.

Clearly $s_{\mathsf{init}} \,\mathrm{R}\, \bar{s}_{\mathsf{init}}$. Assume $s \,\mathrm{R}\, \bar{s}$ and $s \xrightarrow{a} r$. Note that $s = \bar{s}$ by construction of R. In case this direction is also present in $\bar{D}$ condition 2 holds. In case this direction is not present in $\bar{D}$, there exist directions $s \xrightarrow{a} (s, a)$ and $(s, a) \xrightarrow{\tau} r$ in $\bar{D}$, thus condition 2 holds.

Proof of property 2 The proof is by induction on n. For $n = 0$, in case $s \in G$ both sides are 1. In case $s \notin G$ the left side is 0.

Now assume the property holds for n. We will prove it also holds for $n+1$. In case $s \in G$ both sides are 1. Now assume $s \notin G$, then by Lemma 3, when $\bar{A}(s) = (a, \mu)$:

$$\mathsf{ProbReach}_{\bar{A}}^{\leq n+1}(s, G) = \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}_{A[s \xrightarrow{a,\mu} r]}^{\leq n}(r, G)$$

By induction there exist policies A_r such that:

$$\leq \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}_{A_r}(r, G)$$

Let A be an policy such that $A(s \xrightarrow{a,\mu} \omega') = A_r(\omega')$, and $A(\omega)$ is arbitrary in case ω does not have the form $s \xrightarrow{a,\mu} \omega'$. By Definition 3:

$$\begin{aligned} &= \sum_{r \in S} \mu(r) \cdot \mathsf{ProbReach}_{A[s \xrightarrow{a,\mu} r]}(r, G) \\ &= \mathsf{ProbReach}_A(s, G) && \text{by Lemma 3} \end{aligned}$$

□