Verifying PCTL Specifications on Markov Decision Processes via Reinforcement Learning

Yu Wang[1], Nima Roohi[2], Matthew West[3],
Mahesh Viswanathan[3], and Geir E. Dullerud[3]

[1] Duke University, Durham, NC 27707, USA.
`yu.wang094@duke.edu`
[2] University of California San Diego, La Jolla, CA 92093, USA.
`nroohi@ucsd.edu`
[3] University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.
{`mwest, vmahesh, dullerud`}`@illinois.edu`

Abstract. There is a growing literature on verifying temporal logic specifications using reinforcement learning on probabilistic systems with nondeterministic actions, e.g., Markov Decision Processes (MDPs), with unknown transition probabilities. In those works, the optimal policy for the satisfaction probability of a temporal logic specification is learned by optimizing a corresponding reward structure on a product MDP, derived by combining the dynamics of the initial MDP and that of the automata realizing the specification. In this work, we propose a new reinforcement learning method without using the product MDP technique to avoid the expansion of the state space. Specifically, we consider a variant of Probabilistic Computation Tree Logic (PCTL) that includes the (bounded and unbounded) until and release operators and allows reasoning over maximal/minimal satisfaction probability. This logic is verified on MDPs whose states are labeled deterministically by a set of atomic propositions. We first consider PCTL formulas with non-nested probability operators. For non-nested bounded-time PCTL specification, we use an upper-confidence-bound (UCB) Q-learning method to learn the optimal satisfaction probability of interest. The Q-learning is performed inductively on the time horizon, since the corresponding optimal policy can be memory-dependent. Then, we show that the verification technique for non-nested bounded-time specifications can be extended to handle non-nested unbounded-time specifications by finding a proper truncation time. To verify a nested PCTL specifications, a hierarchy of optimal policies corresponding to the nesting structure of the specifications is engaged; we propose a hierarchical Q-learning method to learn those policies simultaneously. Finally, we evaluate the proposed method on several case studies.

1 Introduction

During the past decade, statistical verification of temporal logic specifications has drawn increasing attention [1,20]. The general idea is to infer the correctness of the temporal specifications from samples with probabilistic guarantees using statistical tools. Compared to symbolic approaches, statistical model checkers have two advantages: they are usually more scalable to large problems with complicated stochastic behavior, and they can handle "black-box" stochastic systems with unknown transition probabilities [25].

Currently, most statistical model checkers only handle purely probabilistic models. However, many probabilistic models of interest also have nondeterminism. For example, on a Markov decision process (MDP), the transition probabilities varies under different nondeterministic policies (i.e., the actions taken along the path). In this case, to verify if a given temporal logic specification is satisfied for all (or for some) nondeterministic actions, it is required to identify the optimal actions and to check the temporal logic specification for those actions. For example, checking if the satisfaction probability of a temporal logic formula ϕ on an MDP is less than p for any policy, is equivalent to checking if the satisfaction probability of ϕ is less than p for the optimal policy.

In this work, we study such problems of statistically verifying temporal logic specifications on Markov decision processes (MDPs), via finding the optimal policies for them. This is related, but different from the works of optimizing a given reward function, while subject to constraints defined by temporal logic specifications, by, for example, converting the temporal logic constraints into a barrier function or a shield [14, 21].

When the MDP is known, the optimal policy with respect to the temporal logic formula can be computed analytically from the transition probabilities by, for example, dynamic programming [3,7]. On the other hand, when the transition probabilities are unknown, the optimal policy should be learned from samples. Generally, there are two classes of learning methods. One is model-based [2, 9, 13], in which, the transition probabilities of the MDP are inferred with high accuracy from sufficient sampling for each state-action pair, and then the optimal policy is computed analytically from the estimated transition probabilities. The disadvantage of this approach is the lack of efficiency; to identify the optimal policy, it is not necessary to estimate all the transition probability with high accuracy. To alleviate this problem, an iterative estimation and verification is proposed by repeating the above procedure for batches of samples [7].

Another class of methods is to use model-free learning. In this approach, the optimal policy of the temporal logic specification is learned directly from samples, thus it is generally more sample efficient as all the samples are used to learn the quantities of interest [26]. In designing model-free learning algorithms for temporal logic objectives, the major challenge is to find a surrogate reward structure that is state-dependent and accumulative over time. To be more specific, existing learning algorithms can only handle reward functions where the return of a path is the sum or discounted sum of the rewards from all the visited states; applying them to temporal logic specifications requires converting the specifications into surrogate reward structures of that form.

For specific temporal logic specification, such as safety, a surrogate reward function can be found in a straight-forward manner [10]. For certain variations of temporal logic specifications, interpreted under special semantics (usually involving a metric), the surrogate reward can be found based on the semantics [15, 22, 23].

For temporal logic specifications under the standard semantics [3], a common technique to convert them to proper surrogate reward functions is to construct the product MDP [6, 11, 12]. The product MDP is derived by combing the dynamics of the initial MDP and the limit deterministic Büchi automaton (LDBA) realizing the temporal logic objective. Then, the temporal logic objectives on the initial MDP is first converted to a repeated reachability objective, and then to a state dependent reward function [6, 11, 12].

A disadvantage of the product MDP is the expansion of the states. For example, a bounded-time formula with time horizon T, e.g., $\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2$, generally converts to an LDBA with T states. Accordingly, the product MDP can have $T-1$ times more states than the initial MDP. Another disadvantage is for the discounted reward structure — the discount factor usually needs to be set to close to 1, making the convergence rate of the learning algorithm slow.

In this work, we propose a new learning-based statistical verification method for PCTL on MDPs without using the produce MDP approach. The learning algorithm directly infers the optimal policy on the MDP without lifting the problem onto the product MDP, thus avoid the problem of state-space expansion. Here, the PCTL logic includes temporal operators *next*, *bounded until*, *unbounded until*, *bounded release*, and *unbounded release*. As bounded temporal operators are allowed, the optimal policy on the MDP is not necessarily memoryless; and our learning algorithm directly learns an optimal policy with memory. This is different from [16, 29], where only memoryless policies are considered.

To efficiently identify and evaluate the optimal policies, without over sampling on sub-optimal policies, we propose to use Q-learning with the principle of optimism in the face of uncertainty (OFU). The efficiency of the OFU approach has been proved for simpler decision-learning problems like the K-bandit, and has been numerically demonstrated on several harder problems involving MDPs (see [27] for a review). The general idea of this approach is to use existing samples to construct the upper confidence bound (UCB) for the estimated reward of taking a state-action pair. This UCB represents the maximal possible reward for taking that state-action pair. By choosing the policy greedily to the largest UCB for each iteration, it is ensured that, after several iterations, only the optimal state-action pairs receive the most samples.

We then compute the optimal satisfaction probability by Q-learning using the OFU principle. The value of the Q-function $Q(s, a)$ gives the maximal/minimal satisfaction probability of the goal set after taking an action a on a state s, depending on the PCTL specification to verify. For each iteration, we construct the UCBs on the Q-function to estimate the maximal possible rewards of an action, select the actions greedily with the largest UCB, and then refine the estimation on that UCB with new samples. The initial UCBs for all actions are set high. Thus, the UCB of an action is large if (i) the actual reward is large, or (ii) the action has not been sampled often. Accordingly, new samples will either explore an under-sampled action or evaluate an optimal action. If the under-sampled action is a sub-optimal action, the refined UCB will be very likely to drop below the optimal one, and the sub-optimality will be detected after a few tries. In this way, the proposed algorithm achieves more efficient identification and evaluation of the optimal policy than previous works [4, 7, 9].

Based on this, we design online UCB-based Q-learning algorithms to statistically check PCTL specifications with desired confidence levels on MDPs with unknown transition probabilities. For given confidence levels, we derive the corresponding termination conditions, and prove the probabilistic correctness of the result. We construct the UCBs using Hoeffding's inequality with the boundedness of the satisfaction probabilities. Admittedly, these Hoeffding's bounds may not be tight, but, to our knowledge, this is the best possible way to yield hard probabilistic guarantees, when the MDP is unknown. Tighter bounds like Bernstein's bounds generally only hold asymptomatically [24].

In addition, we consider the statistical verification of nested PCTL formulas. This requires searches for a hierarchy of policies corresponding to the satisfaction of each sub-formula. For example, consider $\psi = \mathbf{P}^{\max}_{>p_1}(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$ with $\rho = \mathbf{P}^{\max}_{>p_2}(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$. Its verification requires searching for a hierarchy of two policies: (i) a high-level policy Π_H for verifying ψ, and (ii) a low-level policy Π_L for verifying the sub-formula ρ.

In the policy hierarchy, search for the high-level policy Π_H depends on the knowledge of the low-level policy Π_L, i.e., which states satisfy ρ. In this work, we design a hierarchical learning algorithm that searches both Π_L and Π_H simultaneously [5]. Iteratively, the algorithm (i) coarsely searches for Π_L on all the states and roughly estimates their satisfaction of ρ, (ii) searches for Π_H based on the rough estimations, (iii) refines the search for Π_L on states likely to be reached under Π_H, (iv) returns to (ii) until the result is accurate enough.

The rest of the paper is organized as follows. The preliminaries on labeled MDPs and PCTL are given in Section 2. In Section 3, using the principle of optimism in the face of uncertainty, we design Q-learning algorithms to solve finite-time and infinite-time probabilistic satisfaction, and give finite sample probabilistic guarantees for the correctness of the algorithms. Then, we study the statistical verification of nested PCTL formulas via hierarchical learning in Section 4. We implement and evaluate the proposed algorithms on several case studies in Section 5. Finally, we conclude this work in Section 6.

2 Preliminaries

The set of integers and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. The cardinality of a set is denoted by $|\cdot|$. The set of finite-length sequences taken from a finite set S is denoted by S^*.

2.1 Markov Decision Process

Markov decision processes (MDPs) are a common probabilistic model with non-deterministic actions. In this work, we consider an MDP where the states are labeled deterministically by a set of atomic propositions AP and only part of actions from A are on each state.

Definition 1 (Markov Decision Process). *A labeled Markov decision process (MDP) is a tuple* $\mathcal{M} = (S, A, \mathbf{T}, s_{\text{init}}, \text{AP}, L)$ *where*

- S *is a finite set of* states.
- A *is a finite set of* actions.
- $\mathbf{T} : S \times A \times S \to [0,1]$ *is a (partial)* transition probability *function. For any state* $s \in S$ *and any action* $a \in A$*,*

$$\sum_{s' \in S} \mathbf{T}(s, a, s') = \begin{cases} 0, & \textit{if } a \textit{ is not allowed on } s \\ 1, & \textit{otherwise.} \end{cases}$$

  With a slight abuse of notation, let $A(s)$ *be the set of allowed actions on the state* s*.*
- $s_{\text{init}} \in S$ *is an* initial state.
- AP *is a finite set of* labels.
- $L : S \to 2^{\text{AP}}$ *is a* labeling function.

The nondeterminism in the MDP is resolved by fixing a (memory/history-dependent) policy.

Definition 2 (Policy). *A policy* $\Pi : S^* \to A$ *decides the current action from the sequence of states visited so far. Applying the policy* Π *on the MDP* $\mathcal{M}$ *gives a purely probabilistic model* $\mathcal{M}_\Pi$*, which is not necessarily Markovian, i.e., a discrete-time Markov chain.*

2.2 PCTL

In this work, we consider a version of Probabilistic Computation Tree Logic (PCTL) that includes the "bounded until" operators, the release operators and allows reasoning over maximal/minimal satisfaction probability. This logic is a fraction of PCTL* in [3].

Definition 3 (PCTL Syntax). *Let* AP *be a set of atomic propositions. A PCTL state formula is defined by*

$$\begin{aligned} \phi ::= \mathsf{a} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi) \mid \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi) \\ \mid \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) \mid \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) \mid \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) \mid \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) \end{aligned}$$

where $\mathsf{a} \in \text{AP}$*,* $\bowtie \in \{<, >, \leq, \geq\}$*,* $T \in \mathbb{N} \cup \{\infty\}$ *is a (possibly infinite) time horizon, and* $p \in [0,1]$ *is a threshold, The operators* $\mathbf{P}^{\min}_{\bowtie p}$ *and* $\mathbf{P}^{\max}_{\bowtie p}$ *are called probability operators, and the "next", "until" and "release" operators* $\mathbf{X}$*,* $\mathbf{U}_T$*,* $\mathbf{R}_T$ *are called temporal operators.*

More temporal operators can be derived by composition: for example, “or” is $\phi_1 \vee \phi_2 ::= \neg(\neg\phi_1 \wedge \neg\phi_2)$; “true” is $\texttt{True} = \mathsf{a} \vee (\neg \mathsf{a})$; “finally” is $\mathbf{F}_T \phi ::= \texttt{True}\mathbf{U}_T \phi$; and “always” is $\mathbf{G}_T \phi ::= \texttt{False}\mathbf{R}_T \phi$. For simplicity, we write $\mathbf{U}_\infty$, $\mathbf{R}_\infty$, $\mathbf{F}_\infty$ and $\mathbf{G}_\infty$ as $\mathbf{U}$, $\mathbf{R}$, $\mathbf{F}$ and $\mathbf{G}$, respectively. The semantics of PCTL is defined with respect to a labeled MDP; the probability operators quantify over the probability of a random path of the MDP.

Definition 4 (PCTL Semantics). *Let $\mathcal{M} = (S, A, \mathbf{T}, s_{\text{init}}, \text{AP}, L)$ be a labeled MDP. The satisfaction relation $\models$ for state formulas is defined on a state s of $\mathcal{M}$ by*

$$\begin{array}{ll}
s \models \mathsf{a} & \textit{iff}\ \ \mathsf{a} \in L(s), \\
s \models \neg\phi & \textit{iff}\ \ s \not\models \phi, \\
s \models \phi_1 \wedge \phi_2 & \textit{iff}\ \ s \models \phi_1 \textit{ and } s \models \phi_2, \\
s \models \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi) & \textit{iff}\ \ \min_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \mathbf{X}\phi\big] \bowtie p, \\
s \models \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi) & \textit{iff}\ \ \max_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \mathbf{X}\phi\big] \bowtie p, \\
s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) & \textit{iff}\ \ \min_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \phi_1 \mathbf{U}_T \phi_2\big] \bowtie p, \\
s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) & \textit{iff}\ \ \max_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \phi_1 \mathbf{U}_T \phi_2\big] \bowtie p, \\
s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) & \textit{iff}\ \ \min_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \phi_1 \mathbf{R}_T \phi_2\big] \bowtie p, \\
s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) & \textit{iff}\ \ \max_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\big[\sigma \models \phi_1 \mathbf{R}_T \phi_2\big] \bowtie p, \\
\sigma \models \mathbf{X}\phi & \textit{iff}\ \ \sigma(1) \models \phi, \\
\sigma \models \phi_1 \mathbf{U}_T \phi_2 & \textit{iff}\ \ \exists i \leq T.\ \sigma(i) \models \phi_2 \wedge \big(\forall j < i.\ \sigma(i) \models \phi_1\big), \\
\sigma \models \phi_1 \mathbf{R}_T \phi_2 & \textit{iff}\ \ \sigma \not\models \neg\phi_1 \mathbf{U}_T \neg\phi_2
\end{array}$$

where $\bowtie \in \{<, >, \leq, \geq\}$ and $\sigma \sim \mathcal{M}_{\Pi,s}$ means the path σ is drawn from the MDP $\mathcal{M}$ under the policy Π, starting from the state s from.

Generally, the PCTL formulas $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi)$ (or $s \models \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi)$) means that the maximal (or minimal) satisfaction probability of “next” ϕ is $\bowtie p$. The PCTL formulas $s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$ (or $s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$) means that the maximal (or minimal) satisfaction probability that ϕ_1 holds “until” ϕ_2 holds is $\bowtie p$. Verifying these PCTL formulas involves identifying an optimal policy Π; this optimal policy generally depends on the history/memory of the path. In this work, we identify such an optimal policy by Q-learning.

3 Verifying Non-Nested PCTL Specifications

In this section, we consider the statistical verification of non-nested PCTL formulas, by using an upper-confidence-bound (UCB) version of Q-learning. For simplicity, we focus on $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$; the treatment for other cases is similar. In Section 3.1, we study PCTL formulas of time horizon 1 – i.e., $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$. Then in Section 3.2, we move to PCTL formulas of finite time horizons – i.e., $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$ for $T > 1$. Finally in Section 3.3, we extend the statistical verification technique to PCTL formulas of infinite time horizons – i.e., $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R} \mathsf{a}_2)$.

Since a statistical approach is adopted, we make the following assumption to ensure the accuracy of the proposed statistical verification algorithms increases with the number of samples.

Assumption 1 *In verifying* $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ *and* $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$ *for* $T \in \mathbb{N} \cup \{\infty\}$ *and* $\bowtie \in \{<, >, \leq, \geq\}$, *we assume that* $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{U}_T \phi_2\big] \neq p$ *and* $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{R}_T \phi_2\big] \neq p$, *respectively.*

When it holds, as the number of samples increases, the samples will be increasingly concentrated on one side of the threshold p by the Central Limit Theorem. Therefore, a statistical analysis based on the majority of the samples has increasing accuracy. When it is violated, the samples would be evenly distributed between the two sides of the boundary p, regardless of the sample size. Thus, no matter how the sample size increases, the accuracy of any statistical test would not increase. Compared to other statistical verification algorithms based on sequential probability ratio tests (SPRT) [28, 30], we do not assume a known indifference region.

By Assumption 1, we have the additional semantic equivalence between the PCTL formulas: $\mathbf{P}^{\max}_{<p}\psi \equiv \mathbf{P}^{\max}_{\leq p}\psi$ and $\mathbf{P}^{\max}_{>p}\psi \equiv \mathbf{P}^{\max}_{\geq p}\psi$; thus, we will not distinguish them below.

To verify the two PCTL formulas on the MDP $\mathcal{M} = (S, A, \mathbf{T}, s_{\text{init}}, \text{AP}, L)$ from Definition 1, we introduce the following lemma.

Lemma 1. *For* $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ *with* $T \in \mathbb{N} \cup \{\infty\}$, *if* $\mathsf{a}_1 \notin L(s)$ *and* $\mathsf{a}_2 \notin L(s)$, *then* $\mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{U}_T \phi_2\big] = 0$ *for any policy* Π *— we denote these states by* S_0*; if* $\mathsf{a}_2 \in L(s)$, *then* $\mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{U}_T \phi_2\big] = 1$ *for any policy* Π *— we denote these states by* S_1. *Similarly, for* $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$, *if* $\mathsf{a}_1 \notin L(s)$ *and* $\mathsf{a}_2 \notin L(s)$, *then* $\mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{R}_T \phi_2\big] = 0$ *for any policy* Π *(*S_0 *states). if* s *is in an end component labeled by* a_2 *(this requires knowing the topology of the MDP), then* $\mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}} \big[\sigma \models \phi_1 \mathbf{R}_T \phi_2\big] = 1$ *for any policy* Π *(*S_1 *states).*

3.1 Single Time Horizon

To begin with, we verify PCTL formulas $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$. For $s \in S \backslash (S_0 \cup S_1)$, $\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2$ (or $\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2$) holds on a random path σ starting from the state s if and only if $\sigma(1) \in S_1$, where S_0 and S_1 are from Lemma 1. To verify $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ (or $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$), it suffices to learn from samples whether

$$\max_{a \in A(s)} Q_1(s, a) > p, \tag{1}$$

where

$$Q_1(s, a) = \mathbb{P}_{\substack{\sigma(1) \sim T(s,a,\cdot) \\ \sigma(0) = s}} \big[\sigma \models \phi_1 \mathbf{U}_1 \phi_2\big]$$

and $\sigma(1) \sim T(s, a, \cdot)$ means $\sigma(1)$ is drawn from the transition probability $T(s, a, \cdot)$ for the given state s and action a. This is a $|A(s)|$-arm bandit problem; we solve this problem by upper-confidence-bound (UCB) strategies [8, 18].

Specifically, for the iteration k, let $N^{(k)}(s, a, s')$ be the number samples for the one-step path (s, a, s'), and with a slight abuse of notation, let

$$N^{(k)}(s, a) = \sum_{s' \in S} N^{(k)}(s, a, s'). \tag{2}$$

The unknown transition probability function $\mathbf{T}(s, a, s')$ is estimated by the empirical transition probability function

$$\hat{\mathbf{T}}^{(k)}(s, a, s') = \begin{cases} \frac{N^{(k)}(s,a,s')}{N^{(k)}(s,a)}, & \text{if } N^{(k)}(s, a) > 0 \\ \frac{1}{|S|}, & \text{if } N^{(k)}(s, a) = 0 \end{cases} \tag{3}$$

And the estimation of $Q_1(s, a)$ from the existing k samples is

$$\hat{Q}_1^{(k)}(s, a) = \sum_{s' \in S_1} \hat{\mathbf{T}}^{(k)}(s, a, s'). \tag{4}$$

Since the value of the Q-function $Q_1(s, a) \in [0, 1]$ is bounded, we can construct a confidence interval for the estimate $\hat{Q}_1^{(k)}$ with confidence level $1 - \delta$ using Hoeffding's inequality by

$$\begin{aligned} \underline{Q}_1^{(k)}(s, a) &= \max \left\{ \hat{Q}_1^{(k)}(s, a) - \sqrt{\frac{|\ln(\delta/2)|}{2N^{(k)}(s, a)}}, 0 \right\}, \\ \overline{Q}_1^{(k)}(s, a) &= \min \left\{ \hat{Q}_1^{(k)}(s, a) + \sqrt{\frac{|\ln(\delta/2)|}{2N^{(k)}(s, a)}}, 1 \right\}, \end{aligned} \tag{5}$$

where we set the value of the division to be ∞ for $N^{(k)}(s, a) = 0$.

The sample efficiency for learning for the bandit problem (1) depends critically on the choice of sampling strategy, decided from the existing samples. A provably best solution is to use the principle of optimism in the face of uncertainty (OFU) [8, 18]. Specifically, an upper confidence bound (UCB) is constructed for each state-action pair using the number of samples and the observed reward, and choose the best action with the highest possible reward, namely the UCB. The OFU principle chooses the policy that gives the maximal possible reward greedily — in this case, we choose

$$\pi_1^{(k)}(s) = \operatorname{argmax}_{a \in A(s)} \overline{Q}_1^{(k)}(s, a), \tag{6}$$

with an optimal action chosen arbitrarily when there are multiple candidates. The choice of $\pi_1^{(k)}$ in (6) ensures that the policy giving the upper bound of the value function gets most frequently sampled in the long run.

Remark 1. To verify $s \models \mathbf{P}^{\min}_{>p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ or $s \models \mathbf{P}^{\min}_{>p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$, it suffices choose the actions that minimizes the satisfaction probability, i.e., to replace argmax with argmin in (6). The same statements also holds for general PCTL formulas, as discussed in Sections 3.2 and 3.3

Algorithm 1 Verifying $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$

```
Require: MDP $\mathcal{M}$, parameter $\delta$.
1: Initialize the Q-function, and the policy by (7)(6).
2: Obtain $S_0$ and $S_1$ by Lemma 1.
3: while True do
4:     Sample from $\mathcal{M}$, and update the transition probability function by (2)(3).
5:     Update the bounds on the Q-function and the policies by (5)(6).
6:     Check termination condition (8).
7: end while
```

To initialize the iteration, the Q-function is set to

$$\overline{Q}_1^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \notin S_0 \\ 0, & \text{otherwise,} \end{cases} \quad \underline{Q}_1^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \in S_1 \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

to ensure that every state-action is sampled at least once. The termination condition of the above algorithm is

$$\begin{cases} \text{true}, & \text{if } \max_{a \in A(s)} \underline{Q}_1^{(k)}(s) > p \\ \text{false}, & \text{if } \max_{a \in A(s)} \overline{Q}_1^{(k)}(s) < p \\ \text{continue}, & \text{otherwise,} \end{cases} \tag{8}$$

where p is the probability threshold in the non-nested PCTL formula.

Remark 2. To verify $s \models \mathbf{P}^{\max}_{<p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{<p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$, it suffices change the termination condition (8) by returning true if $\overline{Q}_1^{(k)}(s) < p$, and returning false if $\underline{Q}_1^{(k)}(s) > p$. The same statements also holds for general PCTL formulas, as discussed in Sections 3.2 and 3.3

The above discussion is summarized by Algorithm 1 and Theorem 1.

Theorem 1. *Algorithm 1 returns the correct answer with probability at least* $1 - |A|\delta$.

Proof. We provide the proof of a more general statement in Theorem 2.

Remark 3. The Hoeffding confidence intervals in (5) are conservative. Consequently, as shown in the simulations in Section 5, the actual error probabilities of the algorithms proposed in this work are considerably smaller than the nominal confidence levels. However, as the MDP is unknown, finding tighter confidence intervals is challenging. One possible solution is to use asymptotic bounds, such as the Bernstein's bounds [24] — accordingly the algorithm gives only asymptotic probabilistic guarantees.

3.2 Finite Time Horizon

Now, we consider PCTL formulas of time horizon $T \in \mathbb{N}$. For clarity, we focus on $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ (and similarly for $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$); other formulas are handled in the same way. Again, on the MDP $\mathcal{M}$, if $s \in S_0$ or $s \in S_1$, where S_0 and S_1 are from Lemma 1, then the maximal satisfaction probabilities of $\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2$ is 0 and 1, respectively; the verification is trivial. For $s \in S \backslash (S_0 \cup S_1)$, let

$$\begin{aligned} V_h(s) &= \max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}(\sigma \models \mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2), \\ Q_h(s,a) &= \max_{\Pi(s)=a} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}(\sigma \models \mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2), \quad h \in [T], \end{aligned} \tag{9}$$

i.e., $V_h(s)$ and $Q_h(s,a)$ are the maximal satisfaction probability of $\mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2$ for a random path starting from s for any policy and any policy with first action being a, respectively. By definition, $V_h(s)$ and $Q_h(s,a)$ satisfies the Bellman equation

$$\begin{aligned} V_h(s) &= \max_{a \in A} Q_h(s,a), \\ Q_{h+1}(s,a) &= \sum_{s' \in S} \mathbf{T}(s,a,s') V_h(s') \\ &= \sum_{s \in S \backslash (S_0 \cup S_1)} \mathbf{T}(s,a,s') V_h(s') + \sum_{s' \in S_1} \mathbf{T}(s,a,s'). \end{aligned} \tag{10}$$

where the second equality of the second equation holds by the semantics of PCTL that

$$V_h(s) = \begin{cases} 0, & \text{if } s \in S_0, \\ 1, & \text{if } s \in S_1. \end{cases}$$

From (10), we can statistically verify $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2)$ by induction on the time horizon T. For $h \in T$, the lower and upper bounds for $Q_h(s,a)$ can be derived using the bounds on the value function for the previous step — for $h = 1$ from (5) and for $h > 0$ by the following lemma.

$$\begin{aligned} \underline{Q}^{(k)}_{h+1}(s,a) = \max \Bigg\{ 0, & \sum_{s \in S \backslash (S_0 \cup S_1)} \hat{\mathbf{T}}^{(k)}(s,a,s') \underline{V}_h(s') + \\ & \sum_{s' \in S_1} \hat{\mathbf{T}}^{(k)}(s,a,s') - \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}} \Bigg\}, \\ \overline{Q}^{(k)}_{h+1}(s,a) = \max \Bigg\{ 1, & \sum_{s \in S \backslash (S_0 \cup S_1)} \hat{\mathbf{T}}^{(k)}(s,a,s') \underline{V}_h(s') + \\ & \sum_{s' \in S_1} \hat{\mathbf{T}}^{(k)}(s,a,s') + \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}} \Bigg\}, \end{aligned} \tag{11}$$

and

$$\overline{V}^{(k)}_h(s) = \max_{a \in A(s)} \overline{Q}^{(k)}_h(s,a), \quad \underline{V}^{(k)}_h(s) = \max_{a \in A(s)} \underline{Q}^{(k)}_h(s,a), \tag{12}$$

where δ_h is a significance parameter such that $Q_h(s,a) \in [\underline{Q}_h^{(k)}(s,a), \overline{Q}_h^{(k)}(s,a)]$ with probability at least $1-\delta_h$. The bounds (11) derives from (10) by applying Hoeffding's inequality, using the fact that $\mathbb{E}[\hat{\mathbf{T}}^{(k)}(s,a,s')] = \mathbf{T}(s,a,s')$ and the Q-functions are bounded within $[0,1]$.

From the boundedness of $Q_h(s,a) \in [0,1]$, we note that this confidence interval encompasses the statistical error in both the estimated transition probability function $\hat{\mathbf{T}}^{(k)}(s,a,s')$ and the bounds $\overline{V}_h^{(k)}(s,a)$ and $\underline{V}_h^{(k)}(s,a)$ of the value function. Accordingly, the policy $\pi_h^{(k)}$ chosen by the OFU principle at the h step is

$$\pi_h^{(k)}(s) = \mathrm{argmax}_{a \in A(s)} \overline{Q}_h^{(k)}(s,a), \tag{13}$$

with an optimal action chosen arbitrarily when there are multiple candidates, to ensure that the policy giving the upper bound of the value function is sampled the most in the long run. To initialize the iteration, the Q-function is set to

$$\overline{Q}_h^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \notin S_0 \\ 0, & \text{otherwise,} \end{cases} \quad \underline{Q}_h^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \in S_1 \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

for all $h \in [T]$, to ensure that every state-action is sampled at least once.

Sampling by the updated policy $\pi_h^{(k)}(s)$ can be performed in either episodic or non-episodic ways [27]. The only requirement is that the state-action pair $(s, \pi_h^{(k)}(s))$ should be performed frequently for each $h \in [T]$ and the state s satisfying $s \in S \backslash (S_0 \cup S_1)$. In addition, a batch samples may be drawn, namely sampling over the state-action pairs multiple times before updating the policy. In this work, for simplicity, we use a non-episodic, non-batch sampling method, by drawing

$$s' \sim \mathbf{T}(s, \pi_h^{(k)}(s), \cdot), \quad \text{for all } h \in [T], \text{ and } s \text{ with } \mathsf{a}_1 \in L(s), \mathsf{a}_2 \notin L(s). \tag{15}$$

The Q-function and the value function are set and initialized by (12) and (14). The termination condition is give by

$$\mathbf{P}_{>p}^{\max}\phi : \begin{cases} \text{false}, & \text{if } \overline{V}_H^{(k)}(s_0) < p, \\ \text{true}, & \text{if } \underline{V}_H^{(k)}(s_0) > p, \\ \text{continue}, & \text{otherwise,} \end{cases} \tag{16}$$

where p is the probability threshold in the non-nested PCTL formula. The above discussion is summarized by Algorithm 2 and Theorem 2.

Theorem 2. *Algorithm 2 terminates with probability* 1 *and has a confidence level of* $1 - N|A|\sum_{h \in [T]} \delta_h$, *where* $N = |S \backslash (S_0 \cup S_1)|$.

Proof. By construction, as the number of iterations $k \to \infty$, $\overline{V}_T^{(k)} - \underline{V}_T^{(k)} \to 0$. Thus, by Assumption 1, the termination condition (16) will be satisfied with

Algorithm 2 Verifying $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$

```
Require: MDP $\mathcal{M}$, parameters $\delta_h$ for $h \in [T]$.
1: Initialize the Q-function and the policy by (14)(13).
2: Obtain $S_0$ and $S_1$ by Lemma 1.
3: while true do
4:     Sample by (15), and update the transition probability function by (2)(3).
5:     Update the bounds by (11)(12) and the policy by (13).
6:     Check the termination condition (16).
7: end while
```

probability 1. Now, let $\mathtt{E}$ be the event that Algorithm 2 gives right answer and let $\mathtt{F}_k$ be the event that Algorithm 2 terminates at the iteration k, then we have $\mathbb{P}(\mathtt{E}) = \sum_{k \in \mathbb{N}} \mathbb{P}(\mathtt{E}|\mathtt{F}_k)\mathbb{P}(\mathtt{F}_k)$. For any k, the event $\mathtt{E}$ happens given that $\mathtt{F}_k$ holds, if the Hoeffding confidence intervals given by (11) hold for any actions $a \in A$, $h \in [T]$, and state s with $s \in S \backslash (S_0 \cup S_1)$. Thus, we have $\mathbb{P}(\mathtt{E}|\mathtt{F}_k) \geq 1 - N|A| \sum_{h \in [T]} \delta_h$, where $N = |S \backslash (S_0 \cup S_1)|$, implying that the confidence level of Algorithm 2 is $\mathbb{P}(\mathtt{E}) \geq 1 - N|A| \sum_{h \in [T]} \delta_h$.

Remark 4. By Theorem 2, the desired overall significance level splits into the significance levels for each state-action pairs through the time horizon. For implementation, we can split the significance level equally, namely $\delta_1 = \cdots = \delta_H$.

Remark 5. Statistically verifying $\mathbf{P}^{\min}_{\bowtie p}(\phi)$ for $\bowtie \in \{<, >, \leq, \geq\}$ is derived by replacing argmax with argmin in (13), and max with min in (12). The termination condition is the same as (16).

Remark 6. Due to the semantics in Definition 4, running Algorithm 2 proving $\mathbf{P}^{\max}_{>p}(\phi)$ or disproving $\mathbf{P}^{\max}_{<p}(\phi)$ is easier than disproving $\mathbf{P}^{\max}_{>p}(\phi)$ or proving $\mathbf{P}^{\max}_{<p}(\phi)$; and the difference increases with the number of actions $|A|$ and the time horizon T. This is because proving $\mathbf{P}^{\max}_{>p}(\phi)$ or disproving $\mathbf{P}^{\max}_{<p}(\phi)$ requires only finding and evaluating some policy Π with $\mathbb{P}_{\mathcal{M}_\Pi}[s \models \phi] > p$, while disproving it requires evaluating all possible policies with sufficient accuracy. This is illustrated by the simulation results presented in Section 5.

3.3 Infinite Time Horizon

Infinite-step satisfaction probability can be estimated from finite-step satisfaction probabilities, using the monotone convergence of the value function in the time step H,

$$V_0(s) \leq \ldots \leq V_H(s) \leq \ldots \leq V(s) = \lim_{H \to \infty} V_H(s). \tag{17}$$

Therefore, if the satisfaction probability is larger than p for some step H, then the verification algorithm should terminate – i.e.,

$$\begin{cases} \text{false}, & \text{if } \underline{V}_H^{(k)}(s_0) > p, \\ \text{continue}, & \text{otherwise}, \end{cases} \tag{18}$$

Algorithm 3 Verifying $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$ by UCB Q-learning

```
Require: MDP $\mathcal{M}$, parameters $\delta_h$ for $h \in \mathbb{N}$.
1: Initialize two sets of Q-function and the policy by (14)(13) for (i) $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$
       and (ii) $\mathbf{P}^{\min}_{>1-p}(\neg\mathsf{a}_1 \mathbf{R} \neg\mathsf{a}_2)$, respectively.
2: Obtain $S_0$ and $S_1$ for (i) and (ii) respectively by Lemma 1.
3: while True do
4:     Sample by (15), and update $\hat{\mathbf{T}}^{(k)}(s, a, s')$ by (2)(3).
5:     Update the bounds on the Q-function, the policies, the value function, and the
           time horizon by (11)(13)(12)(19) respectively for (i) and (ii).
6:     Check the termination condition (18).
7: end while
```

where p is the probability threshold in the non-nested PCTL formula.

The general idea in using the monotonicity to check infinite horizon satisfaction probability in finite time is that if we check both $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$ and its negation $\mathbf{P}^{\min}_{>1-p}(\neg\mathsf{a}_1 \mathbf{R} \neg\mathsf{a}_2)$ at the same time, one of them should terminate in finite time. Here $\neg\mathsf{a}_1$ and $\neg\mathsf{a}_2$ are treated as atomic propositions. We can use Algorithm 2 to check their satisfaction probabilities for any time horizon T simultaneously. The termination in finite time is guaranteed, if the time horizon for both computation increases with the iterations. The simplest choice is to increase H by 1 for every K iterations; however, this brings the problem of tuning K. Here, we use the convergence of the best policy as the criterion for increasing H for each satisfaction computation. Specifically, for all the steps h in each iteration, in addition to finding the optimal policy $\pi^{(k)}_h(s)$ with respect to the UCBs of the Q-functions $\overline{Q}^{(k)}_h(s,a)$ by (13), we also consider the the optimal policy with respect to the lower confidence bounds of the Q-functions $\underline{Q}^{(k)}_h(s,a)$. Obviously, when $\pi^{(k)}_h(s) \in \operatorname{argmax}_{a \in A} \underline{Q}^{(k)}_h(s,a)$, we know that the policy $\pi^{(k)}_h(s)$ is optimal for all possible Q-functions within $[\underline{Q}^{(k)}_h, \overline{Q}^{(k)}_h]$. This implies that these bounds are fine enough for estimating Q_H; thus, if the algorithm does not terminate by the condition (18), we let

$$H \leftarrow \begin{cases} 1, & \text{initially} \\ H+1, & \text{if } \pi^{(k)}_H(s) \in \operatorname{argmax}_{a \in A} \underline{Q}^{(k)}_H(s,a) \text{ for all } s \in S \\ \text{Continue}, & \text{otherwise.} \end{cases} \tag{19}$$

Combining the above procedure, we derive Algorithm 3 and Theorem 3 below for statistically verifying PCTL formula $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$.

Theorem 3. *Algorithm 3 terminates with probability* 1 *and has a confidence level of* $1 - |A| \max\{N_1, N_2\} \sum_{h \in [T]} \delta_h$*, where H is the largest time horizon when the algorithm stops,* $N_1 = |S \backslash (S^\phi_0 \cup S^\phi_1)|$ *and* $N_2 = |S \backslash (S^\psi_0 \cup S^\psi_1)|$ *with* $S^\phi_0 \cup S^\phi_1$ *and* $S^\psi_0 \cup S^\psi_1$ *derived from Lemma 1 for* $\phi = \mathsf{a}_1 \mathbf{U} \mathsf{a}_2$ *and* $\psi = \mathsf{a}_1 \mathbf{R} \mathsf{a}_2$*, respectively.*

Proof. Terminates with probability 1 follows easily from (17). Following the proof of Theorem 2, if the procedure of statistically verifying either $\mathbf{P}^{\max}_{<p}\phi$ or its

negation $\mathbf{P}^{\min}_{<1-p}(\neg\phi)$ stops and the largest time horizon is H, then the probability that the result is correct is at least $1 - |A| \max\{N_1, N_2\} \sum_{h\in[T]} \delta_h$. Thus, the theorem holds.

Remark 7. By Theorem 3, given the desired overall confidence level δ, we can split it geometrically by $\delta_h = (1-\lambda)\lambda^{h-1}\delta$, where $\lambda \in (0,1)$.

Remark 8. Similar to in Section 3.2, statistically verifying $\mathbf{P}^{\min}_{\sim p}(\phi)$ for $\sim \in \{<, >, \leq, \geq\}$ is derived by replacing argmax with argmin in (13), and max with min in (12). The termination condition is the same as (18).

Remark 9. Finally, we note that the exact savings of sample costs for Algorithms 2 and 3 depend on the structure of the MDP. Specifically, the proposed method is more efficient more than [4, 7, 9], when the satisfaction probabilities differ significantly among actions, as it can quickly detect sub-optimal actions without over-sampling on them. On the other hand, if all the state-action pairs yield the same Q-value, then an equal amount of samples will be spent on each of them — in this case, the sample cost of Algorithms 2 and 3 is the same as [4,7,9].

4 Verifying Nested PCTL Specifications

The statistical verification of a nested PCTL formula requires searching for a hierarchy of policies corresponding to the satisfaction of each probabilistic sub-formula. Here, we take the nested formula $\psi = \mathbf{P}^{\max}_{>p_1}(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$ with $\rho = \mathbf{P}^{\max}_{>p_2}(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$ as an example. As shown in Figure 1, by the semantics from Definition 4, its verification requires searching for a hierarchy of two policies: (i) a high-level policy Π_H for verifying ψ, and (ii) a low-level policy Π_L for verifying the sub-formula ρ. More specifically, the high-level policy Π_H drives the path of $\mathcal{M}$ from an initial state to maximize the satisfaction probability of $\mathsf{a}_1 \mathbf{U}_{T_1} \rho$; and, after that, the low-level policy Π_L drives the path to maximize the satisfaction probability of $\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3$.

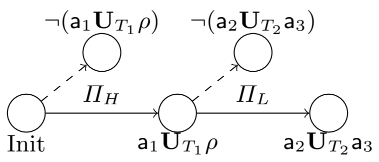
$\neg(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$
$\neg(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$
Π_H
Π_L
Init
$\mathsf{a}_1 \mathbf{U}_{T_1} \rho$
$\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3$

Fig. 1. Hierarchy of policies for verifying nested PCTL formula $\psi = \mathbf{P}^{\max}_{>p_1}(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$ with $\rho = \mathbf{P}^{\max}_{>p_2}(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$.

In the policy hierarchy, the search for the high-level Π_H depends on the knowledge of the low-level Π_L, i.e., which states satisfy ρ. Conventionally, this is done by first exhaustively checking ρ on all the states of $\mathcal{M}$ with high probabilistic accuracy by extensively searching for Π_L, and then, based on those results, checking the whole formula ψ by searching for Π_H. This exhaustive search is inefficient because the overall accuracy of checking ψ is mainly dependent on the accuracy of checking ρ on the states that are likely to be reached under Π_H. In other words, accurately checking ρ on the states that are not reachable from an initial state under the policy Π_H is not needed.

To remedy this, we design a hierarchical learning algorithm that searches both Π_L and Π_H simultaneously. Iteratively, the algorithm (i) coarsely searches for Π_L on all the states and roughly estimates their satisfaction of ρ, (ii) searches for Π_H based on the rough estimates, (iii) refines the search for Π_L on states likely to be reached under Π_H, (iv) returns to (ii) until the result is accurate enough.

Specifically, for statistically verifying $\psi = \mathbf{P}^{\max}_{>p_1}(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$ with $\rho = \mathbf{P}^{\max}_{>p_2}(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$, we can first verify $s \models \rho$ on all the states $s \in S$ with confidence level $1 - \alpha$; this assertion is denoted by

$$\mathcal{A}(s \models \rho) = \begin{cases} 0, & \text{if Algorithm 2 returns no,} \\ 1, & \text{otherwise.} \end{cases} \tag{20}$$

Based on these (probabilistically correct) assertions, we can verify ψ. Let $V^{\psi}_h(s)$ and $Q^{\psi}_h(s,a)$ be the maximal satisfaction probability of $\mathsf{a}_1 \mathbf{U}_h \rho$ for a random path starting from s for any policy and any policy with first action being a from (9), respectively. Then, they should satisfy the Bellman equation for any $h \in [T]$,

$$\begin{aligned} Q^{\psi}_{h+1}(s,a) &= \sum_{s' \in S} \mathbf{T}(s,a,s') V^{\psi}_h(s') \\ &= \sum_{\substack{\mathsf{a}_1 \in L(s') \\ s' \not\models \rho}} \mathbf{T}(s,a,s') V^{\psi}_h(s') + \sum_{s' \models \rho} \mathbf{T}(s,a,s'), \end{aligned} \tag{21}$$

where

$$V^{\psi}_h(s) = \max_{a \in A} Q^{\psi}_h(s,a). \tag{22}$$

The value $Q^{\psi}_h(s,a)$ in (21) is approximated by replacing the condition of $s' \models \rho$ with the return $\mathcal{A}(s \models \rho)$ of Algorithm 2.

Lemma 2. *If the assertion is correct with probability at least $1 - \alpha(s)$ on the state s, then for the k step, $Q^{\psi}_h(s,a) \in [\underline{Q}^{\psi,(k)}_h(s,a), \overline{Q}^{\psi}, (k)_h(s,a)]$ holds with*

Algorithm 4 Verifying $\psi = \mathbf{P}^{\max}_{>p_1}(\mathsf{a}_1 \mathbf{U}_{T_1} \rho)$ with $\rho = \mathbf{P}^{\max}_{>p_2}(\mathsf{a}_2 \mathbf{U}_{T_2} \mathsf{a}_3)$

```
Require: MDP $\mathcal{M}$, desired significance level $\alpha$
1: while True do
2:     For $s \in S$, verify $\rho$ on $s$ with significance level $\alpha(s)$.
3:     Using (23)(24) to update $\Pi_H$ and the value function $\overline{V}(s)$ with significance $\alpha_d$
           Until (i) Equation (13) is satisfied or (ii) $\overline{V}(s_0) - \underline{V}(s_0) < 2\min_{s\in S} \alpha_s$ or
           (iii) after $N = 100$ iterations
4:     $\alpha_s \leftarrow \alpha_s \exp(-c\overline{V}(s))$
5: end while
```

probability at least $1 - \delta_h$*, where*

$$\begin{aligned}\underline{Q}^{\psi,(k)}_{h+1}(s,a) = \max\Bigg\{0, &\sum_{\substack{\mathsf{a}_1 \in L(s') \\ \mathcal{A}(s \models \rho)=0}} \hat{\mathbf{T}}^{(k)}(s,a,s')\underline{V}^{\psi,(k)}_h(s')(1-\alpha(s'))+ \\ &\sum_{\mathcal{A}(s\models\rho)=1} \hat{\mathbf{T}}^{(k)}(s,a,s')(1-\alpha(s')) - \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}}\Bigg\}, \\ \overline{Q}^{\psi,(k)}_{h+1}(s,a) = \max\Bigg\{1, &\sum_{\substack{\mathsf{a}_1 \in L(s') \\ \mathcal{A}(s \models \rho)=0}} \hat{\mathbf{T}}^{(k)}(s,a,s')\underline{V}^{\psi,(k)}_h(s') + \sum_{\mathcal{A}(s\models\rho)=1} \hat{\mathbf{T}}^{(k)}(s,a,s') \\ &\sum_{\substack{\mathsf{a}_1 \notin L(s') \\ \mathcal{A}(s \models \rho)=0}} \hat{\mathbf{T}}^{(k)}s(s,a,s')\underline{V}^{\psi,(k)}_h(s')\alpha(s') + \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}}\Bigg\},\end{aligned} \tag{23}$$

and

$$\overline{V}^{\psi,(k)}_h(s) = \max_{a \in A(s)} \overline{Q}^{\psi,(k)}_h(s,a), \quad \underline{V}^{\psi,(k)}_h(s) = \max_{a \in A(s)} \underline{Q}^{\psi,(k)}_h(s,a), \tag{24}$$

Proof. The lemma holds because $\mathcal{A}(s \models \rho) = 1$ coincides with $s \models \rho$ (and $\mathcal{A}(s \models \rho) = 0$ coincides with $s \not\models \rho$) with probability at least $1 - \alpha(s)$.

Using Lemma 2, we derive the following hierarchical learning algorithm for nested PCTL formulas.

In Algorithm 4, the low-level and high-level estimations are refined iteratively. If the low-level estimation is too coarse, the confidence parameters will be refined by $\alpha_s \leftarrow \alpha_s \exp(-c\overline{V}(s))$.

5 Simulation

To evaluate the performance of the proposed algorithms, we ran them on two different sets of examples. In all the simulations, the transition probabilities are unknown to the algorithm (this is different from [7]).

The first set contains 10 randomly generated MDPs with different sizes. For these MDPs, we considered the formula $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$ for the finite horizon, and $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U} \alpha_2)$ for the infinite horizon. In both cases, α_1, α_2, and H are chosen arbitrarily. The second set contains 9 versions of the Sum of Two Dice program. The standard version [17] models the sum of two fair dice, each with 6 different possibilities numbered $1, \ldots, 6$. To consider MDPs with different sizes, we consider 9 cases where each dice is still fair, but has n possibilities numbered $1, \ldots, n$, with $n = 3$ in the smallest example, and $n = 17$ in the largest example. For these MDPs, we considered the formula $\mathbf{P}^{\max}_{<p}(\mathbf{F}_H \alpha)$ for the finite horizon, and $\mathbf{P}^{\max}_{<p}(\mathbf{F} \alpha)$ for the infinite horizon. In both cases α encodes the atomic predicate that is true iff values of both dice are chosen and their sum is less than an arbitrarily chosen constant. Also, in the finite case, $H = 5$ in the smallest example and 10 everywhere else.

For each MDP, we first numerically estimate the values of $\max_{\Pi} \mathbb{P}_{\mathcal{M}_{\Pi}}[\alpha_1 \mathbf{U}_H \alpha_2]$ and $\max_{\Pi} \mathbb{P}_{\mathcal{M}_{\Pi}}[\alpha_1 \mathbf{U} \alpha_2]$, for the randomly generated MDPs, and $\max_{\Pi} \mathbb{P}_{\mathcal{M}_{\Pi}} [\mathbf{F}_H \alpha]$, and $\max_{\Pi} \mathbb{P}_{\mathcal{M}_{\Pi}}[\mathbf{F} \alpha]$, for the variants of the two-dice examples, using the known models on PRISM [19]. Then, we use Algorithms 2 and 3 to perform verifications on the example models with only knowledge of the topology of the MDPs and without knowing the exact transition probabilities. For every MDP, we tested each algorithm with two different thresholds p, one smaller and the other larger than the estimated probability, to test the proposed algorithms, with the significance level δ set to 5%. We ran each randomly generated test 100 times and each two-dice variant test 10 times. Here we only report average running time and average number of iterations. All tests returned the correct answers — this suggests that the Hoeffding's confidence intervals used in the proposed algorithms are conservative (see Remark 3). The algorithms are implemented in Scala and ran on Ubuntu 18.04 with i7-8700 CPU 3.2GHz and 16GB memory.

Tables 1 and 2 show the results for finite horizon reachability. An interesting observation in these tables is that in all examples, disproving the formula is 3 to 100 times faster. We believe this is mainly because, to disprove $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$, all we need is *one* policy Π for which $\mathbf{P}_{>p}(\alpha_1 \mathbf{U}_H \alpha_2)$ holds. However, to prove $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$, one needs to show that *every* policy Π satisfies $\mathbf{P}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$ (see Remark 6).

Tables 3 and 4 show the results for infinite horizon reachability. Note that Algorithm 3 considers both the formula and its negation, and contrary to the finite horizon reachability, disproving a formula is not always faster for the infinite case. In most of the larger examples that are randomly generated, H_1 and H_2 are very small on average. This shows that in these examples, the algorithm was smart enough to learn there is no need to increase H in order to solve the problem. However, this is not the case for two-dice examples. We believe this is because in the current implementation, the decision to increase H does not consider the underlying graph of the MDP. For example, if during the execution, the policy forces the state to enter a self-loop with only one enabled action, which is the case in two-dice examples, then after every iteration the value of H will be increased by 1.

$\lvert S\rvert$	$\lvert A\rvert$	H	Iter.	Time (s)	p	PRISM Est.
3	3	4	208.5	0.02	0.25	≈ 0.3533
		4	3528.7	0.19	0.45	
4	2	4	171.8	0.01	0.09	≈ 0.1934
		4	3671.7	0.22	0.29	
5	2	4	441.7	0.04	0.05	≈ 0.1176
		4	4945.5	0.42	0.21	
10	2	4	544.7	0.14	0.04	≈ 0.0965
		4	5193.3	1.45	0.19	
15	2	3	873.1	0.28	0.04	≈ 0.0946
		3	4216.7	1.28	0.19	
20	4	5	337.6	0.99	0.12	≈ 0.2225
		5	9353.3	28.06	0.32	
25	5	10	270.5	7.57	0.09	≈ 0.1912
		10	25709.8	728.49	0.29	
30	5	10	355.6	14.35	0.08	≈ 0.1731
		10	27161.7	1085.77	0.27	
35	5	10	328.9	18.82	0.09	≈ 0.1826
		10	27369.6	1529.84	0.28	
40	5	10	390.0	26.79	0.08	≈ 0.1674
		10	30948.8	2122.24	0.26	

Table 1. Verifying $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$ on random MDPs. Atomic propositions α_1 and α_2 are chosen arbitrarily. Column "Iter." is the average number of iterations, Column "PRISM Est." is the PRISM's estimation of the actual probability.

n	$\lvert S\rvert$	H	Iter.	Time (s)	p	PRISM Est.
3	36	5	15.6	0.79	0.27	≈ 0.3750
			39.7	1.31	0.47	
5	121	10	32.9	4.07	0.20	≈ 0.3027
			93.8	11.30	0.40	
6	169	10	29.2	4.95	0.29	≈ 0.3955
			90.4	15.33	0.49	
7	196	10	33.4	6.85	0.31	≈ 0.4101
			70.7	13.84	0.51	
9	400	10	41.1	15.87	0.21	≈ 0.3105
			110.3	43.47	0.41	
11	529	10	46.4	24.07	0.28	≈ 0.3867
			111.9	58.15	0.48	
13	729	10	25.8	18.52	0.20	≈ 0.3046
			109.9	80.28	0.40	
15	1156	10	42.1	47.90	0.05	≈ 0.1025
			155.3	178.29	0.20	
17	1369	10	24.0	32.17	0.06	≈ 0.1328
			155.0	208.78	0.23	

Table 2. Verifying $\mathbf{P}^{\max}_{<p}(\mathbf{F}_H \alpha)$ on variants of sum-of-two-dice MDPs. Atomic proposition α is true iff both dice have chosen their numbers and their sum is less than an arbitrary constant. Column n is the number of alternatives on each dice. Each MDP has two actions (*i.e.* $|A| = 2$). Column "Iter." is the average number of iterations, Column "PRISM Est." is the PRISM's estimation of the actual probability.

6 Conclusion

In this work, we proposed statistical verification methods for Probabilistic Computation Tree Logic (PCTL) with (bounded and unbounded) until and release

$\|S\|$	$\|A\|$	Iter.	Time (s)	p	PRISM Est.	H_1	H_2
3	3	126.0	0.03	0.25	≈ 0.3537	14.7	15.8
		111.3	0.01	0.45		12.9	13.0
4	2	103.7	0.01	0.09	≈ 0.1934	13.0	27.2
		73.0	0.01	0.29		9.6	19.2
5	2	239.9	0.06	0.05	≈ 0.1176	12.4	59.3
		92.7	0.00	0.21		6.9	24.3
10	2	891.4	0.24	0.04	≈ 0.0965	3.2	225.6
		79.6	0.00	0.19		1.1	21.5
15	2	1862.0	0.61	0.04	≈ 0.0946	1.3	117.8
		161.3	0.01	0.19		1.0	11.0
20	4	1336.2	0.27	0.12	≈ 0.2225	1.0	1.0
		16843.8	3.02	0.32		1.0	1.8
25	5	1619.2	0.64	0.09	≈ 0.1912	1.0	1.0
		154621.6	56.56	0.29		1.0	2.0
30	5	2246.8	1.05	0.08	≈ 0.1731	1.0	1.0
		86617.0	42.53	0.27		1.0	1.3
35	5	1925.1	0.82	0.09	≈ 0.1826	1.0	1.0
		13219.0	5.79	0.28		1.0	1.0
40	5	2401.5	1.35	0.08	≈ 0.1674	1.0	1.0
		12158.6	6.66	0.26		1.0	1.0

Table 3. Verifying $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U} \alpha_2)$ on random MDPs. Columns H_1 and H_2 are values of the corresponding variables in Algorithm 3 at termination.

n	$\|S\|$	Iter.	Time (s)	p	PRISM Est.	H_1	H_2
3	36	191.1	3.39	0.56	≈ 0.6666	192.1	110.9
		232.6	2.82	0.76		233.6	117.4
5	121	359.9	15.01	0.29	≈ 0.3999	153.6	360.9
		378.6	18.15	0.49		149.0	379.6
6	169	268.0	8.15	0.31	≈ 0.4166	140.2	71.9
		572.4	19.88	0.51		127.5	73.6
7	196	291.2	17.30	0.32	≈ 0.4285	136.3	292.1
		281.1	16.42	0.52		129.7	282.0
9	400	545.1	109.93	0.55	≈ 0.6543	394.6	199.2
		593.8	129.73	0.75		445.9	200.2
11	529	493.4	106.65	0.44	≈ 0.5454	210.3	179.1
		799.0	173.00	0.64		288.9	177.2
13	729	720.6	231.51	0.36	≈ 0.4615	116.3	276.8
		393.1	118.15	0.56		127.2	300.6
15	1156	2027.9	1762.51	0.36	≈ 0.4666	134.9	627.0
		1374.7	833.66	0.56		155.2	489.2
17	1369	1995.4	1105.89	0.37	≈ 0.4705	107.1	161.3
		1469.4	767.87	0.57		100.7	165.4

Table 4. Verifying $\mathbf{P}^{\max}_{<p}(\mathbf{F}\alpha)$ on sum-of-two-dice MDPs. Columns H_1 and H_2 are values of the corresponding variables in Algorithm 3 at termination.

operators on Markov decision processes with unknown transition probabilities using reinforcement learning. We first verified PCTL formulas with non-nested probability operators and bounded time horizon, using an upper-confidence-bounds (UCB) version of Q-learning to learn the optimal satisfaction probability of interest. Then, we showed that the verification technique for non-nested bounded-time specifications can be extended to handle non-nested unbounded-time specifications by finding a proper truncation time. In addition, we proposed a hierarchical Q-learning method to verify nested PCTL specifications. Finally, we evaluated the proposed method on several case studies.

References

1. Agha, G., Palmskog, K.: A Survey of Statistical Model Checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018)
2. Ashok, P., Křetínský, J., Weininger, M.: PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games. arXiv:1905.04403 [cs] (2019)
3. Baier, C., Katoen, J.P.: Principles of Model Checking (2008)
4. Balle, B., Mohri, M.: Learning Weighted Automata. In: Algebraic Informatics, vol. 9270, pp. 1–21 (2015)
5. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems **13**(1-2), 41–77 (2003)
6. Bozkurt, A.K., Wang, Y., Zavlanos, M., Pajic, M.: Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA) (Submitted) (2020)
7. Brázdil, T., Chatterjee, K., Chmelík, M., Forejt, V., Křetínský, J., Kwiatkowska, M., Parker, D., Ujma, M.: Verification of Markov Decision Processes Using Learning Algorithms. In: Automated Technology for Verification and Analysis. pp. 98–114 (2014)
8. Bubeck, S.: Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. Foundations and Trends® in Machine Learning **5**(1), 1–122 (2012)
9. Fu, J., Topcu, U.: Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. arXiv:1404.7073 [cs] (2014)
10. Fulton, N., Platzer, A.: Verifiably Safe Off-Model Reinforcement Learning. arXiv:1902.05632 [cs] **11427**, 413–430 (2019)
11. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-Regular Objectives in Model-Free Reinforcement Learning. In: Tools and Algorithms for the Construction and Analysis of Systems, vol. 11427, pp. 395–412 (2019)
12. Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G.J., Lee, I.: Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. arXiv:1909.05304 [cs, eess, stat] (2019)
13. Henriques, D., Martins, J.G., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical Model Checking for Markov Decision Processes. In: 2012 Ninth International Conference on Quantitative Evaluation of Systems. pp. 84–93 (2012)
14. Jansen, N., Könighofer, B., Junges, S., Bloem, R.: Shielded Decision-Making in MDPs. arXiv:1807.06096 [cs] (2018)
15. Jones, A., Aksaray, D., Kong, Z., Schwager, M., Belta, C.: Robust Satisfaction of Temporal Logic Specifications via Reinforcement Learning. arXiv:1510.06460 [cs] (2015)
16. Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.P.: Safety-Constrained Reinforcement Learning for MDPs. arXiv:1510.05880 [cs] (2015)
17. Knuth, D., Yao, A.: Algorithms and Complexity: New Directions and Recent Results, chap. The complexity of nonuniform random number generation. Academic Press (1976)

18. Kuleshov, V., Precup, D.: Algorithms for multi-armed bandit problems. arXiv:1402.6028 [cs] (2014)
19. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Computer Aided Verification, vol. 6806, pp. 585–591 (2011)
20. Larsen, K.G., Legay, A.: Statistical Model Checking: Past, Present, and Future. In: Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques. pp. 3–15 (2016)
21. Li, X., Belta, C.: Temporal Logic Guided Safe Reinforcement Learning Using Control Barrier Functions. arXiv:1903.09885 [cs, stat] (2019)
22. Li, X., Vasile, C.I., Belta, C.: Reinforcement Learning With Temporal Logic Rewards. arXiv:1612.03471 [cs] (2016)
23. Littman, M.L., Topcu, U., Fu, J., Isbell, C., Wen, M., MacGlashan, J.: Environment-Independent Task Specifications via GLTL. arXiv:1704.04341 [cs] (2017)
24. Mnih, V., Szepesvári, C., Audibert, J.Y.: Empirical Bernstein stopping. In: Proceedings of the 25th International Conference on Machine Learning - ICML '08. pp. 672–679 (2008)
25. Roohi, N., Wang, Y., West, M., Dullerud, G.E., Viswanathan, M.: Statistical verification of the Toyota powertrain control verification benchmark. In: 20th ACM International Conference on Hybrid Systems: Computation and Control (HSCC). pp. 65–70 (2017)
26. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction p. 352 (2018)
27. Szepesvári, C.: Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **4**(1), 1–103 (2010)
28. Wang, Y., Roohi, N., West, M., Viswanathan, M., Dullerud, G.E.: Statistical verification of PCTL using stratified samples. In: 6th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), IFAC-PapersOnLine. vol. 51, pp. 85–90 (2018)
29. Wen, M., Ehlers, R., Topcu, U.: Correct-by-synthesis reinforcement learning with temporal logic constraints. arXiv:1503.01793 [cs] (2015)
30. Zuliani, P.: Statistical model checking for biological applications. International Journal on Software Tools for Technology Transfer **17**(4), 527–536 (2015)