A PAC Learning Algorithm for LTL and Omega-Regular Objectives in MDPs

Mateo Perez, Fabio Somenzi, Ashutosh Trivedi

University of Colorado Boulder
Mateo.Perez@colorado.edu, Fabio@colorado.edu, Ashutosh.Trivedi@colorado.edu

Abstract

Linear temporal logic (LTL) and ω-regular objectives—a superset of LTL—have seen recent use as a way to express non-Markovian objectives in reinforcement learning. We introduce a model-based probably approximately correct (PAC) learning algorithm for ω-regular objectives in Markov decision processes (MDPs). As part of the development of our algorithm, we introduce the ε-recurrence time: a measure of the speed at which a policy converges to the satisfaction of the ω-regular objective in the limit. We prove that our algorithm only requires a polynomial number of samples in the relevant parameters, and perform experiments which confirm our theory.

Introduction

Reinforcement learning (RL) (Sutton and Barto 2018) is a sampling-based approach to learning a controller. Inspired by models of animal behavior, the RL agent interacts with the environment and receives feedback on its performance in terms of a numerical reward, that either reinforces or punishes certain behaviors. This learning approach has produced impressive results in recent years (Mnih et al. 2015; Silver et al. 2016). However, failure to precisely capture designer's intent in reward signals can lead to the agent learning unintended behavior (Amodei et al. 2016). As a response, formal languages—in particular linear temporal logic (LTL) and ω-regular languages—have been proposed to unambiguously capture learning objectives. While these languages have enjoyed practical success (Hahn et al. 2019; Bozkurt et al. 2020), their theoretical complexity is relatively underexplored. In this paper we propose and study a model-based probably approximately correct RL algorithm for LTL and ω-regular languages.

Probably approximately correct (PAC) learning (Valiant 1984) is a framework for formalizing guarantees of a learning algorithm: a user selects two parameters, $\varepsilon > 0$ and $\delta > 0$. A learning algorithm is then (efficient) PAC if it returns a solution that is ε close to optimal with probability at least $1 - \delta$ using a polynomial number of samples. In RL, many PAC learning algorithms have been proposed for both discounted and average reward (Kakade 2003; Brafman and Tennenholtz 2003). These algorithms usually provide sample bounds in terms of the sizes of the state and action spaces of the Markov decision process (MDP) that describes the environment. Finite-horizon and discounted reward both have the property that small changes to the transition probabilities result in small changes to the value of the objective. This means that the sample complexity is independent of the transition probabilities of the MDP. However, infinite-horizon, undiscounted objectives, like average reward and the satisfaction of LTL properties, are sensitive to small changes in probabilities, and their sample complexity is dependent on some knowledge of the transition probabilities. Hence, if only the number of state/action pairs is allowed, alongside $1/\varepsilon$ and $1/\delta$, as parameters, creating a PAC learning algorithm for undiscounted, infinite-horizon properties is not possible. Specifically for LTL, this has been observed by Yang, Littman, and Carbin (2021) and Alur et al. (2022).

Example 1 (Intractability of LTL)**.** *Figure 1 is an example adopted from (Alur et al. 2022) that shows the number of samples required to learn safety properties is dependent on some property of the transition structure. The objective in this example is to stay in the initial state s_0 forever. This can be specified with average reward (a reward of 1 in s_0 and 0 otherwise) and in LTL ($\varphi = \mathbf{G}s_0$). The transition from s_0 to s_1 under action b must be observed in order to distinguish action a from action b and produce an ε-optimal policy for any $\varepsilon < 1$. The number of samples required to see this transition with high probability is affected by the value of p. Smaller values of p means it takes longer for a policy's finite behavior to match its infinite behavior.*

This non-PAC-learnability may motivate using discounted versions of LTL (Littman et al. 2017; Alur et al. 2023), which, however, have significantly different semantics from

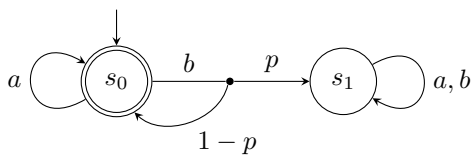

Figure 1: Example adopted from (Alur et al. 2022). The objective is to remain in s_0 forever.

the undiscounted logic. One may argue instead that the complexity of the dynamics of an MDP is not entirely captured by the number of state-action pairs. For example, for average reward, Kearns and Singh (2002) use the ε-return mixing time, a measure of how fast the average reward is achieved in a particular system, for this purpose. They argue that in order to know the learning speed of an algorithm, one must know the speed at which the policy achieves the limit average reward. The R-MAX algorithm of Brafman and Tennenholtz (2003) also utilizes the ε-return mixing time.

The ε-return mixing time is defined based off of a given reward function, which we do not have in our context. Therefore, we require an alternative notion. We propose the ε-recurrence time as a way to reason about the speed at which an ω-regular objective is achieved. Informally, the ε-recurrence time is the expected time for a set of recurring states to be visited twice. In Figure 1, the ε-recurrence time increases when p decreases. We will show that this additional parameter is sufficient for defining a PAC algorithm for ω-regular objectives.

Contributions. We introduce a model-based PAC learning algorithm for LTL and ω-regular objectives in Markov decision processes. For our algorithm, we introduce the ε-recurrence time: a measure of the speed at which a policy converges to the satisfaction of the ω-regular objective in the limit. We show that the number of samples required by our algorithm is polynomial in the relevant input parameters. Our algorithm only requires the ability to sample trajectories of the system, and does not require prior knowledge of the exact graph structure of the MDP. Finally, we demonstrate the practicality of our algorithm on a set of case studies.

Related work. A PAC learning algorithm for LTL was introduced by Fu and Topcu (2014) that uses sampled trajectories to learn, but requires knowledge of the graph structure of the MDP, i.e., which transitions occur with nonzero probability. Brázdil et al. (2014) propose an algorithm with PAC guarantees for unbounded reachability by using the minimum nonzero transition probability, and describe how to extend their method to LTL. Ashok, Kretinsky, and Weininger (2019) utilize the minimum nonzero transition probability to develop an anytime statistical model-checking algorithm for unbounded reachability. Although they do not discuss it, in principle their method can extended to LTL similarly. Voloshin et al. (2022) provide an algorithm with PAC guarantees for LTL which assumes access to a generative model of the system.

Daca et al. (2017) describe a PAC algorithm capable of checking the satisfaction of LTL on a Markov chain. They observe that "some information about the Markov chain is necessary for providing statistical guarantees." The aforementioned works of Alur et al. (2022) and Yang, Littman, and Carbin (2021) formalize this observation.

Alur et al. (2023) study model-free RL for *discounted* LTL, while we do not assume discounting. Hahn et al. (2022) show that Rabin automata are unsuitable for model-free RL of ω-regular objectives. We can use Rabin automata because our algorithm is model-based.

Preliminaries

A Markov decision process (MDP) is a tuple $\mathcal{M} = (S, A, P, s_0)$ where S is the set of states, A is the set of actions, $P : S \times A \times S \to [0, 1]$ is the transition function, and $s_0 \in S$ is the initial state. A run of an MDP is an infinite sequence $s_0, a_0, s_1, a_1, \ldots$ such that $P(s_i, a_i, s_{i+1}) > 0$ for all $i \geq 0$. A Markov chain $M = (S, P, s_0)$ is an MDP where the set of actions is singleton, i.e. S is the set of states, $P : S \times S \to [0, 1]$ is the transition function, and $s_0 \in S$ is the initial state. A bottom strongly connected component (BSCC) of a Markov chain is a bottom strongly connected component of the graph formed by the positive probability edges of the Markov chain. Equivalently, a BSCC of a Markov chain is a set of states $B \subseteq S$ where for all $s, s' \in B$ the probability of reaching s' from s is positive and the probability of reaching a state $s'' \in S \setminus B$ is zero. A policy is a recipe for selecting actions. A policy is positional if it is of the form $\pi : S \to A$. A policy π induces a probability distribution over runs. We denote the probabilities under this distribution by $\Pr_\pi^{\mathcal{M}}(\cdot)$.

Let AP be the set of atomic propositions. An LTL formula has the following grammar

$$\varphi := \top \mid b \in AP \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$$

We write $\bot := \neg\top$, $\varphi \wedge \varphi := \neg(\neg\varphi \vee \neg\varphi)$, the *finally* operator as $\mathbf{F}\varphi := \top \mathbf{U} \varphi$, and the *globally* operator $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$. For a formula φ and an infinite word $\sigma = \sigma_0\sigma_1 \ldots \in (2^{AP})^\omega$ we write $w \vDash \varphi$ to denote that σ satisfies φ. We write $\sigma_{i:\infty} = \sigma_i\sigma_{i+1} \ldots$ for the substring of σ starting at position i. The semantics of LTL are defined as

$$\begin{aligned}
&\sigma \vDash \top \\
&\sigma \vDash a && \text{iff } a \in \sigma_0 \\
&\sigma \vDash \neg\varphi && \text{iff } \sigma \nvDash \varphi \\
&\sigma \vDash \varphi_1 \vee \varphi_2 && \text{iff } \sigma \vDash \varphi_1 \text{ or } \sigma \vDash \varphi_2 \\
&\sigma \vDash \mathbf{X}\varphi && \text{iff } \sigma \vDash \sigma_{1:\infty} \\
&\sigma \vDash \varphi_1 \mathbf{U} \varphi_2 && \text{iff } \exists j \geq 0 \text{ s.t. } \sigma_{j:\infty} \vDash \varphi_2 \\
& && \quad \text{and } \sigma_{i:\infty} \vDash \varphi_1, \forall\, 0 \leq i < j \ .
\end{aligned}$$

Omega-regular languages are a generalization of regular languages for infinite strings. Like regular languages are accepted by finite automata, ω-regular languages are accepted by ω-automata. An ω-automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the (nondeterministic) transition function, $q_0 \in Q$ is an initial state, and F is an acceptance condition over states. The Büchi acceptance condition is $F \subseteq Q$, a subset of accepting states. A Büchi automaton accepts an infinite word σ if there exists a run in $\mathcal{A}$ that visits accepting states infinitely often. We denote the acceptance of an infinite word σ by $\mathcal{A}$ as $\sigma \vDash \mathcal{A}$. It is well known that LTL expresses a subset of the ω-regular languages. There exists many translations from LTL to ω-automata (Duret-Lutz et al. 2016).

Let $\mathcal{M} = (S, A, P, s_0, AP, L)$ be an MDP equipped with atomic propositions AP and a labeling function $L : S \to 2^{AP}$, and let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be an ω-automaton. The probability of satisfaction of $\mathcal{A}$ under a policy π in $\mathcal{M}$ is

$p_\pi^{\mathcal{M}} = \Pr_\pi^{\mathcal{M}}(L(s_0)L(s_1)\ldots \vDash \mathcal{A})$. The optimal probability of satisfaction is $p^{\mathcal{M}} = \sup_\pi p_\pi^{\mathcal{M}}$. If $p_\pi^{\mathcal{M}} = p^{\mathcal{M}}$ then we say that π is optimal. For some $\varepsilon > 0$, if $p_\pi^{\mathcal{M}} \geq p^{\mathcal{M}} - \varepsilon$ then π is ε-optimal.

One can form the product MDP $\mathcal{M}^\times = (S^\times, A^\times, P^\times, (s_0, q_0), F)$ by synchronizing the states of $\mathcal{M}$ and $\mathcal{A}$, i.e. $S^\times = S \times Q$, $A^\times = A \times Q$, and $P^\times((s,q),(a,q),(s',q')) = P(s,a,s')$ if $q' \in \delta(q, L(s))$ and $P^\times((s,q),(a,q),(s',q')) = 0$ otherwise. Note that we can form the product on-the-fly, i.e., we can sample trajectories from $\mathcal{M}^\times$ by simply updating the states of $\mathcal{M}$ and $\mathcal{A}$ separately, and concatenating them at the end. Note that the nondeterminism in the automaton is resolved as actions in the product. If $\mathcal{A}$ is good-for-MDPs (GFM) (Hahn et al. 2020), then the optimal probability of satisfaction in the product $\mathcal{M}^\times$ is the same as the optimal probability of satisfaction of $\mathcal{A}$ in $\mathcal{M}$. Deterministic ω-regular automata are always GFM. One can lift the policy computed in $\mathcal{M}^\times$ to a memoryful policy in $\mathcal{M}$ that uses $\mathcal{A}$ as memory. We note that for the popular acceptance conditions Büchi, parity, and Rabin, computing optimal policies can be done in polynomial time in the size of $\mathcal{M}^\times$. We refer the reader to (Baier and Katoen 2008) for further details.

Main Results

For generality, we examine the setting in which we are given an MDP $\mathcal{M} = (S, A, P, s_0, F)$ equipped with an acceptance condition F on states. To avoid unneeded complexity, we assume that the acceptance condition F is such that there are positional optimal policies. This is true for the popular acceptance conditions Büchi, parity, and Rabin.[1]

Recall that if one begins with an MDP $\mathcal{M} = (S, A, P, s_0, AP, L)$ equipped with atomic propositions AP and a labeling function $L : S \to 2^{AP}$, and a suitable GFM ω-automaton—which may have been constructed from an LTL formula—then one can form the product MDP. The policy produced on the product MDP can be lifted to a memoryful strategy on the original MDP with the same guarantees. Additionally, recall that the product MDP construction can be done on-the-fly for unknown MDPs. Thus, our problem formulation is general enough to capture producing policies for LTL and ω-regular objectives on MDPs with its states labeled by atomic propositions.

We begin by defining the ε-recurrence time in Markov chains.

Definition 1. *The ε-recurrence time in a Markov chain $M = (S, P, s_0)$ is the smallest time T such that with probability at least $1 - \varepsilon$ a trajectory of length T starting from the initial state s_0 visits every state in some BSCC twice.*

Intuitively, the ε-recurrence time T is the time needed so that the recurrent behavior of a trajectory of length T matches the recurrent behavior of an infinite extension of that trajectory with probability at least $1 - \varepsilon$. It may be that the ε-recurrence time is unknown, but other parameters, like the minimum positive transition probability $p_{\min}$ are known. In such a case, the following lemma provides an upper bound.

Lemma 1. *Let $M = (S, P, s_0)$ be a Markov chain and $p_{\min} = \min_{\{s,s' | P(s,s')>0\}} P(s, s')$ be the minimum positive transition probability in M. Then the ε-recurrence time T satisfies $T \leq 2|S| \frac{\log(\varepsilon/2)}{\log(1-p_{\min}^{|S|})}$.*

Proof. In the worst case, every state in the Markov chain must be seen at least twice and visiting every state in the Markov chain requires taking a path of length $|S|$ that occurs with probability $p_{\min}^{|S|}$. Attempting this path k times, the probability of succeeding at least once is $1 - (1 - p_{\min}^{|S|})^k$. If $k \geq \frac{\log(\varepsilon/2)}{\log(1-p_{\min}^{|S|})}$ then $1 - (1 - p_{\min}^{|S|})^k \geq 1 - \varepsilon/2$. A lower bound on succeeding twice in $2k$ attemps is $(1 - \varepsilon) \leq (1 - \varepsilon/2)(1-\varepsilon/2)$. Finally, each of the k attempts takes $|S|$ steps in the worst case to yield $T \leq |S|2k = 2|S| \frac{\log(\varepsilon/2)}{\log(1-p_{\min}^{|S|})}$. □

We define the ε-recurrence time in MDPs so that we can reason about all positional policies in an MDP, as the optimal policies of interest are positional.

Definition 2. *The ε-recurrence time of an MDP $\mathcal{M} = (S, A, P, s_0)$ is the maximum ε-recurrence time amongst all Markov chains induced by positional policies in $\mathcal{M}$.*

The ε-recurrence time provides a measure of the speed at which finite trajectories converge to their infinite behavior, i.e., eventually dwell in a BSCC forever. To demonstrate the intuition behind the ε-recurrence time being sufficient to understand long term behavior from finite trajectories, we will sketch a simple model-free algorithm for estimating the probability of satisfaction p in a Markov chain $M = (S, P, s_0, F)$. We will not use this algorithm when we consider MDPs, but it shows that the ε-recurrence time provides sufficient information to learn long term behavior.

Our algorithm samples C trajectories of length T from the initial state and observes the fraction of trajectories that are winning. As we will show, this algorithm has two sources of estimation error: the first since we sample finite length trajectories, and the second since we only sample finitely many times. To analyze the first type of error, we will utilize the definition of the ε-recurrence time. The second type follows from standard statistical results.

Fix $\varepsilon > 0$ and $\delta > 0$, and let T be the ε-recurrence time in M. Let p be the probability of satisfaction in M. Given a trajectory, we can form the trajectory graph by adding an edge from state s to state s' in the graph if a transition from s to s' was observed in the trajectory. Note that if we sampled infinite length trajectories, then the BSCC in the trajectory graph would correspond to a BSCC of the Markov chain. We identify a sampled trajectory as winning if the BSCC in the trajectory graph is winning. The proportion of infinite length trajectories identified as winning is exactly p. We now need to determine the error we accumulate from using trajectories of finite length T.

If we sample trajectories of length T then the BSCC in the trajectory graph is also a BSCC in the Markov chain with probability at least $1 - \varepsilon$, from the definition of the

[1]The results that follow can extended to a Muller condition by replacing instances of "positional" with "deterministic finite memory with memory N" where N is a property dependent constant.

ε-recurrence time. This means that at least $1-\varepsilon$ of the trajectories are identified as if we had access to an oracle. Thus, our first type of error is at most ε.

Let $\widehat{p}$ be the proportion of trajectories of length T that are identified as winning in expectation. We have that $|\widehat{p}-p| \leq \varepsilon$. Sampling trajectories of length T thus gives us a coin biased by $\widehat{p}$ to toss. For the second type of error, we can give a statistical guarantee on estimating the weight of this coin from finite samples within some bound $\varepsilon' > 0$ of its true value. By using Hoeffding's inequality, we get that by sampling C trajectories

$$P(|\widehat{p} - \mathbb{E}[\widehat{p}]| \leq \varepsilon') \geq 1 - 2\exp(-2\varepsilon'^2 C)$$
$$P(|\widehat{p} - p| \leq \varepsilon' + \varepsilon) \geq 1 - 2\exp(-2\varepsilon'^2 C)$$

For simplicity, we can set $\varepsilon' = \varepsilon$, and then select $C \geq \frac{-\ln(\delta/2)}{2\varepsilon^2}$ so that $1 - 2\exp(-2\varepsilon^2 C) \geq 1-\delta$. In summary, this algorithm is a model-free PAC algorithm for identifying the probability of satisfaction in Markov chains, i.e., it returns an estimated probability of satisfaction that is within 2ε of the true value p with probability at least $1-\delta$ after polynomially-many samples.

We have shown that the ε-recurrence time is sufficient to reason about LTL and ω-regular properties in Markov chains. We now turn our attention to MDPs, where we will develop a model-based PAC algorithm that uses the ε-recurrence time.

The ω-PAC Algorithm

For MDPs, we will develop a model-based PAC algorithm inspired by R-MAX (Brafman and Tennenholtz 2003) that utilizes the ε-recurrence time T of $\mathcal{M} = (S, A, P, s_0, F)$. We call our algorithm ω-PAC.

The general approach of our algorithm is to learn the transition probabilities of the MDP with high accuracy (within $\frac{\varepsilon}{|S|T}$ of their true values) and high confidence. We show that this implies that optimal policies on the learned MDP are 6ε-optimal on the real MDP with high confidence (cf. Lemma 4 and Theorem 1). To obtain our polynomial sample complexity results, we design our learned MDP to be *optimistic*: one that provides an upper bound of the probability of satisfaction. This ensures that we continue to explore edges that we do not yet know with high accuracy sufficiently often (cf. Lemma 5 and Theorem 2).

Specifically, our approach keeps track of an estimate $\widehat{\mathcal{M}}$ of the real system. State-action pairs in $\widehat{\mathcal{M}}$ are kept in two categories: *known* and *unknown*. Known edges are edges we have sampled at least k times, while unknown edges we have sampled less than k times. Intuitively, we select k so that known edges are edges we know with high accuracy (within $\frac{\varepsilon}{|S|T}$) and high confidence. For known edges, we use the observed transition distribution. For unknown edges, we set them as transitions to an accepting sink.[2] By setting the values of unknown edges optimistically high, an optimal positional policy π in $\widehat{\mathcal{M}}$ naturally explores the MDP. The algorithm computes an optimal positional policy π in $\widehat{\mathcal{M}}$, samples trajectories of length T from s_0 with π, and then updates $\widehat{\mathcal{M}}$ from these samples. When all edges that π can visit in T steps in $\widehat{\mathcal{M}}$ are known, the algorithm stops and returns π.

[2]For Büchi, one can add the sink state to the accepting set. For parity, one can give this sink state an overriding winning priority (the largest odd priority for max odd semantics). For Rabin, one can add another pair that wins by visiting this sink state forever.

Algorithm 1: ω-PAC

Input: $|S|$, $|A|$, T, $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, and threshold $k > 0$
Output: 6ε-optimal policy π

1: Initialize $\widehat{\mathcal{M}}$, policy π, and visit counts c
2: **while** $\widehat{\mathcal{M}}$ not known **do**
3: Compute optimal positional policy π in $\widehat{\mathcal{M}}$
4: Sample with π for T steps from initial state in $\mathcal{M}$
5: Update $\widehat{\mathcal{M}}$ with threshold k
6: **end while**
7: **return** π

We now present some more details of the ω-PAC algorithm (see Algorithm 1). We initialize the visit counts $c(s,a,s') \leftarrow 0$ for all $s, s' \in S$ and $a \in A$, and π to an arbitrary positional policy (Line 1). Let $c(s,a) = \sum_{s' \in S} c(s,a,s')$. An edge is unknown if $c(s,a) < k$ and is known if $c(s,a) = k$. After sampling a trajectory $\tau \sim \{(s_0,a_0),\ldots,(s_{T-1},a_{T-1})\}$ (Line 4), for each $i \in \{0,1,\ldots,T-1\}$ we update $c(s_i,a_i,s_{i+1}) \leftarrow c(s_i,a_i,s_{i+1})+1$ only if $c(s_i,a_i) < k$. Once $c(s_i,a_i) = k$, we do not continue incrementing the visit counts. We use $\widehat{\mathcal{M}} = (\widehat{S}, A, \widehat{P}, s_0, \widehat{F})$ where $\widehat{S} = S \cup \{\texttt{sink}\}$,

$$\widehat{P}(s,a,s') = \begin{cases} \mathbb{1}_{s'=\texttt{sink}} & c(s,a) < k \vee s = \texttt{sink} \\ \frac{c(s,a,s')}{c(s,a)} & c(s,a) = k \wedge s \neq \texttt{sink} \end{cases}$$

and

$$\widehat{F}(s) = \begin{cases} F(s) & s \neq \texttt{sink} \\ \texttt{accepting} & s = \texttt{sink} \end{cases}$$

for all instances of $\widehat{\mathcal{M}}$ (Lines 1 and 5), where $\mathbb{1}_{s'=\texttt{sink}}$ is the indicator function for $s' = \texttt{sink}$. Note that these updates can be performed without knowing S, A, or F apriori as we only update $\widehat{P}$ to something nontrivial for states that have been visited.

A naive stopping condition (Line 2) would be to stop only when all edges are marked as known. Instead, we will use a more general condition, that all of the edges reachable in T steps under π are known. Formally, let $S_T \subseteq S$ be the set of states reachable in T steps with positive probability under π in $\widehat{\mathcal{M}}$ from s_0. The condition on line 2 holds if $c(s,a) = k$ for all $s \in S_T$ and $a \in A$.

We have presented ω-PAC as an algorithm that returns a single policy π. The same algorithm can also be phrased as producing an infinite sequence of policies π_i for all timesteps $i \geq 0$ where π_i be the policy π in the ω-PAC learning loop after i samples of the system have been taken. If i is greater than the number of samples ω-PAC takes, we define π_i as the policy returned by ω-PAC.

We show that for a selection of k that is polynomial in the input parameters, the policy π returned by the ω-PAC algorithm is 6ε-optimal with probability at least $1-\delta$ (Theorem 1). We will also show our algorithm has a polynomial sample complexity, i.e., the policy π while the algorithm is running is not 9ε-optimal at most some polynomial number of times with probability at least $1-2\delta$ (Theorem 2). We will now introduce the machinery required to prove these results.

We begin by defining an (α, T)-approximation. This is an approximation of an MDP where the probabilities of all transitions up to a depth T are known within α.

Definition 3. *An (α, T)-approximation of an MDP $\mathcal{M} = (S, A, P, s_0, F)$ is an MDP $\mathcal{M}' = (S, A, P', s_0, F)$ such that for all $s, s' \in S_T$ and $a \in A$, $|P(s,a,s') - P'(s,a,s')| \leq \alpha$ and, if $P(s,a,s') = 0$, then $P'(s,a,s') = 0$, where $S_T \subseteq S$ are the states reachable with positive probability in T steps from s_0 under some strategy.*

Note that an (α, T)-approximation of an MDP can be obtained by averaging samples of observed trajectories of length T to produce an estimate of the transition probabilities $P'(s,a,s')$. If $P(s,a,s') = 0$ then that transition is never observed, so $P'(s,a,s') = 0$. Additionally, enough samples will yield $|P(s,a,s') - P'(s,a,s')| \leq \alpha$ with high probability. We show this explicitly in the following lemma.

Lemma 2. *Let $0 < \delta < 1$, $\alpha > 0$, and $M = (S, P, s_0, F)$ be a Markov chain. Let $\widehat{P}(s,s') = \frac{c(s,s')}{k}$ where $c(s,s')$ is the number of observed transitions from s to s' obtained after sampling transitions from a state s, $k \geq \left\lceil \frac{-\ln(\delta/2)}{2\alpha^2} \right\rceil$ times. Then with probability at least $1-\delta$, $|\widehat{P}(s,s') - P(s,s')| \leq \alpha$ and $\widehat{P}(s,s') = 0$ if $P(s,s') = 0$ for all $s' \in S$.*

Proof. Fix $s' \in S$. Since $P(s,s') = 0$ implies that $c(s,s') = 0$ and thus $\widehat{P}(s,s') = 0$, all we need to show is that $|\widehat{P}(s,s') - P(s,s')| \leq \alpha$ with probability at least $1-\delta$. We apply Hoeffding's inequality to get

$$\Pr(|\hat{P}(s,s') - P(s,s')| \leq \alpha) \geq 1 - 2\exp(-2\alpha^2 k).$$

Substituting, we get that

$$1 - 2\exp(-2\alpha^2 k) \geq 1 - \delta. \qquad \square$$

This lemma is helpful for giving a bound on the number of samples required to learn an (α, T)-approximation. In order to determine the appropriate α to select, we'd like to give a bound on the change in the probability of satisfaction between an MDP $\mathcal{M}$ and its (α, T)-approximation $\mathcal{M}'$. To provide such a bound, we use the following result.

Lemma 3. *Let $M = (S, P, s_0, F)$ be a Markov chain, $M' = (S, P', s_0, F)$ be an (α, T)-approximation of M, $N = |S|$ denote the size of the state space. If the probability to reach $s' \in S$ from $s \in S$ in at most T steps in M is p, then the probability to reach s' from s in at most T steps in M' is at least $p - \alpha NT$.*

Proof. Let R_i and R'_i be the events that we reached s' from s in at most i steps in M and M' respectively. We'd like to show that $\Pr(R'_i) \geq \Pr(R_i) - \alpha N i$ for all $i \geq 0$. We show this by induction. For the base case, it is clear that $\Pr(R'_0) = \Pr(R_0)$.

For convenience, we define $p_i = \Pr(R_i)$ and $p'_i = \Pr(R'_i)$. We also define $p_{i|i-1} = \Pr(R_i | \neg R_{i-1})$ and $p'_{i|i-1} = \Pr(R'_i | \neg R'_{i-1})$. Since there are N total transitions, the worst case reduction in the single step transition probabilities between states is at most αN. Thus, $p'_{i|i-1} \geq p_{i|i-1} - \alpha N$. For the inductive step, we can write for $i > 0$

$$\begin{aligned}
\Pr(R'_i) &= \Pr(R'_i | \neg R'_{i-1}) \Pr(\neg R'_{i-1}) + \Pr(R'_{i-1}) \\
&= p'_{i|i-1}(1 - p'_{i-1}) + p'_{i-1} \\
&\geq p'_{i|i-1}(1 - p_{i-1}) + (p_{i-1} - \alpha N(i-1)) \\
&\geq (p_{i|i-1} - \alpha N)(1 - p_{i-1}) + (p_{i-1} - \alpha N(i-1)) \\
&\geq p_{i|i-1}(1 - p_{i-1}) + p_{i-1} - \alpha N i \\
&= \Pr(R_i) - \alpha N i \qquad \square
\end{aligned}$$

We are now ready to bound the difference in the probability of satisfaction between a Markov chain M and its (α, T)-approximation M'.

Lemma 4. *Let $M = (S, P, s_0, F)$ be a Markov chain, $N = |S|$ denote the size of the state space, $\varepsilon > 0$, and T be the ε-recurrence time in M. Let $M' = (S, P', s_0, F)$ be an $(\frac{\varepsilon}{NT}, T)$-approximation of M, and T' be the 2ε-recurrence time in M'. Let p and p' be the probability of satisfaction from s_0 in M and M', respectively. Then,*

1. *$T' \leq T$*
2. *$|p' - p| \leq 3\varepsilon$.*

Proof. We begin by defining the unrolling of a Markov chain and the associated set of lasso states. The unrolling of a Markov chain $M = (S, P, s_0, F)$ is a Markov chain $M_x = (S_x, P_x, (s_0, \mathbf{0}), F_x)$ that has the same dynamics as M, but keeps track of the visitation counts of each state. The set of lasso states L of M_x is the set of states such that there exists a BSCC B in M such that all the visitation counts are greater than or equal to 2 for all $s \in B$. Given a state $s \in L$, we define $L^{-1}(s) = B$ as the function that returns the BSCC B in M corresponding to that state in M_x.

Consider the unrolled Markov chains M_x and M'_x, and their lasso states L and L', for M and M', respectively. The probability of visiting a state $s \in L$ from $(s_0, \mathbf{0})$ in M_x in T steps is at least $1 - \varepsilon$, by definition. By Lemma 3, the probability of visiting a state $s \in L$ from $(s_0, \mathbf{0})$ in M'_x in T steps is at least $1 - 2\varepsilon$. Let $X \subseteq L$ be the set of states in L that are reached with positive probability in T steps in M'_x, and let $\mathcal{B} = \{L^{-1}(x) : x \in X\}$. For each $B \in \mathcal{B}$, all states $s \in B$ can reach each other in M' with positive probability in T steps by definition, and thus are part of the same SCC in M'. Since $P'(s,s') = 0$ if $P(s,s') = 0$, these states form a BSCC in M'.

In summary, every BSCC $B \in \mathcal{B}$ is a BSCC in M and M', and the probability of reaching a state $s \in B$ in T steps from s_0 in M' is at least $1 - 2\varepsilon$. Thus, T is an upper bound on the 2ε-recurrence time in M', proving part 1. Finally, let $p_{\mathcal{B}}$ and $p'_{\mathcal{B}}$ be the probability of reaching a winning BSCC

$B \in \mathcal{B}$ in T steps from s_0 in M and M', respectively. Then,

$$|p' - p| \leq |p'_{\mathcal{B}} - p_{\mathcal{B}}| + 2\varepsilon \quad (1)$$
$$\leq \varepsilon + 2\varepsilon = 3\varepsilon \quad (2)$$

where (1) follows from the fact that the BSCCs in $\mathcal{B}$ are reached with probability at least $1 - 2\varepsilon$ in M', and (2) follows from applying Lemma 3. This proves part 2. □

Since Definition 2 is concerned with positional policies, and optimal policies are positional, Lemma 4 applies directly to an MDP $\mathcal{M}$ and its $(\frac{\varepsilon}{NT}, T)$-approximation $\mathcal{M}'$, producing the same bounds. This motivates selecting the number of samples k to mark an edge as known in the ω-PAC algorithm to be such that we are highly confident that we have an $(\frac{\varepsilon}{NT}, T)$-approximation of $\mathcal{M}$. We can use Lemma 2 to select such a k. We are now ready to show the correctness of the ω-PAC algorithm under the appropriate selection of k.

Theorem 1 (Correctness)**.** *Let $0 < \delta < 1$, $\varepsilon > 0$, $\mathcal{M} = (S, A, P, s_0, F)$ be an MDP, $N = |S|$ denote the size of the state space, $K = |A|$ denote the size of the action space, and T be the ε-recurrence time of $\mathcal{M}$. Let $\varepsilon' = \frac{\varepsilon}{NT}$ and $\delta' = \frac{\delta}{NK}$. For $k = \lceil \frac{-\ln(\delta'/2)}{2\varepsilon'^2} \rceil$, the policy π returned by ω-PAC is 6ε-optimal with probability at least $1 - \delta$.*

Proof. Let $\mathcal{M}'$ be some $(\frac{\varepsilon}{NT}, T)$-approximation of $\mathcal{M}$. Let σ be an optimal positional policy in $\mathcal{M}'$. Let p be the optimal probability of satisfaction in $\mathcal{M}$, and let p_σ and p'_σ be the probability of satisfaction in $\mathcal{M}$ and $\mathcal{M}'$ under σ, respectively. By Lemma 4, we have that

$$|p - p_\sigma| \leq |p - p'_\sigma| + |p'_\sigma - p_\sigma|$$
$$\leq 3\varepsilon + 3\varepsilon = 6\varepsilon.$$

Thus, all we need to show is that with probability at least $1-\delta$ there exists an $(\frac{\varepsilon}{NT}, T)$-approximation $\mathcal{M}'$ of $\mathcal{M}$ such that π is optimal in $\mathcal{M}'$.

Let $\widehat{\mathcal{M}}$ denote the optimistic MDP when ω-PAC terminates. We say that a state-action pair $s \in S$, $a \in A$ in $\widehat{\mathcal{M}}$ is α-accurate if for all $s' \in S$, $|\widehat{P}(s, a, s') - P(s, a, s')| \leq \alpha$ and if $P(s, a, s') = 0$ then $\widehat{P}(s, a, s') = 0$. By Lemma 2, a state-action pair marked as known is $\frac{\varepsilon}{NT}$-accurate with probability at least $1 - \delta'$. Since there are NK total state-action pairs, the probability that all state-action pairs marked as known are $\frac{\varepsilon}{NT}$-accurate is at least $(1-\delta')^{NK} \geq 1-\delta$. Let $\mathcal{M}' = (S, A, \widehat{P}', s_0, F')$ be an MDP such that the transition probabilities for all known state-action pairs are identical to $\widehat{\mathcal{M}}$, are $\frac{\varepsilon}{NT}$-accurate for unknown state-action pairs that are reachable in T steps from s_0 with positive probability under some strategy, and are accepting sinks otherwise. With probability at least $1 - \delta$, $\mathcal{M}'$ is a $(\frac{\varepsilon}{NT}, T)$-approximation of $\mathcal{M}$. Finally, note that the probability of satisfaction in $\widehat{\mathcal{M}}$ and $\mathcal{M}'$ under π is the same since ω-PAC terminates when π only visits known state-action pairs in T steps. Therefore, since the optimal probability of satisfaction $\widehat{p}$ in $\widehat{\mathcal{M}}$ is an upper bound on the probability of satisfaction in $\mathcal{M}'$, by the construction of $\widehat{\mathcal{M}}$, π is optimal in $\mathcal{M}'$. □

Note that $k = \tilde{O}(|S|^2T^2/\varepsilon^2)$ selected in the previous theorem is bounded by a polynomial in the input parameters. For Theorem 1, we assume we run the algorithm until termination, which occurs with probability 1: if it has not terminated, π visits an unknown state-action pair with positive probability in T steps, and there can only be $k|S||A|$ such visits before all state-action pairs are marked as known. We now show sample complexity bounds for the ω-PAC algorithm by showing that the number of timesteps that π is not 9ε-optimal is bounded by a polynomial in $|S|$, $|A|$, T, $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$ with probability at least $1 - 2\delta$. For such a sample complexity result, we need to reason about how often unknown state-action pairs are visited. We show this in the following lemma.

Lemma 5. *Let $\mathcal{M} = (S, A, P, s_0, F)$ be an MDP. Let $\widehat{\mathcal{M}} = (S \cup \{\texttt{sink}\}, A, \widehat{P}, s_0, \widehat{F})$ be identical to $\mathcal{M}$ except some arbitrary set U of state-action of pairs are converted into transitions to the accepting sink. Let π be a positional optimal policy in $\widehat{\mathcal{M}}$, $\alpha > 0$, $\varepsilon > 0$, and T be the ε-recurrence time of $\mathcal{M}$. Then at least one of the following holds:*

1. *π is α-optimal from s_0 in $\mathcal{M}$, or*
2. *a trajectory in $\widehat{\mathcal{M}}$ of length T from s_0 under π visits a state-action pair in U with probability at least $\alpha - \varepsilon$.*

Proof. To prove this lemma, it is sufficient to show that if π is not α-optimal from s_0 in $\mathcal{M}$, then a trajectory in $\widehat{\mathcal{M}}$ of length T from s_0 under π visits a state-action pair in U with probability at least $\alpha - \varepsilon$. Let p_π and $\widehat{p}_\pi$ be the probability of satisfaction that π obtains from s_0 in $\mathcal{M}$ and $\widehat{\mathcal{M}}$, respectively. Let p be the maximum probability of satisfaction in $\mathcal{M}$. We begin by noting that $\widehat{p}_\pi \geq p$ by the construction of $\widehat{\mathcal{M}}$. If π is not α-optimal from s_0 in M, this means that $p - p_\pi \geq \alpha$, which implies $\widehat{p}_\pi - p_\pi \geq \alpha$. As the values $\widehat{p}_\pi$ and p_π only differ due to π reaching state-action pairs in U in $\widehat{\mathcal{M}}$, this means that π must reach a state-action pair in U in $\widehat{\mathcal{M}}$ from s_0 with probability at least α.

Finally, note that $T + 1$ is an upper bound on the ε-recurrence time in $\widehat{\mathcal{M}}$. This is because any policy π in $\widehat{\mathcal{M}}$ that takes a state-action pair in U will visit the BSCC formed by the sink after one additional timestep. For reasoning about reaching a state-action pair in U once, this additional timestep due to the sink state has no effect. Thus, if the probability to reach a state-action pair in U in $\widehat{\mathcal{M}}$ from s_0 under π is at least α, it must be at least $\alpha - \varepsilon$ in T steps. □

We are now able to show the sample complexity of our algorithm. Note that the bound in Theorem 2 on the number of samples $C = \tilde{O}(|S|^3|A|T^3/\varepsilon^4)$ is bounded by a polynomial in $|S|$, $|A|$, T, $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$.

Theorem 2 (Sample Complexity)**.** *Let $0 < \delta < 1$, $\varepsilon > 0$, $\mathcal{M} = (S, A, P, s_0, F)$ be an MDP, $N = |S|$ denote the size of the state space, $K = |A|$ denote the size of the action, and T be the ε-recurrence time of $\mathcal{M}$. Let $\varepsilon' = \frac{\varepsilon}{NT}$ and $\delta' = \frac{\delta}{NK}$. Let π_i be an infinite sequence of policies produced by ω-PAC. For $k = \lceil \frac{-\ln(\delta'/2)}{2\varepsilon'^2} \rceil$, π_i is not 9ε-optimal*

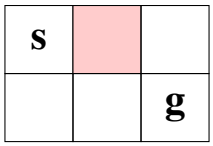

Figure 2: The *gridworld example*. The red cell is a trap. The goal is to visit the two states s and g repeatedly.

for at most $C = T\lceil \max\left(\frac{kNK}{\varepsilon}, \frac{kNK-\ln(\delta)}{2\varepsilon^2}\right)\rceil$ *steps with probability at least* $1-2\delta$.

Proof. From the proof of Theorem 1, all of the state-action pairs marked as known in $\widehat{\mathcal{M}}$ are $\frac{\varepsilon}{NT}$-accurate with probability at least $1-\delta$. For ease of presentation, we will assume that this occurs and incorporate its probability at end of the proof.

Let $\mathcal{M}'$ be an $(\frac{\varepsilon}{NT}, T)$-approximation of $\mathcal{M}$ that matches $\widehat{\mathcal{M}}$ for all of the state-action pairs marked as known at the end of training. By Lemma 4, the 2ε-recurrence time in $\mathcal{M}'$ is at most T. The maximum number of visits to unknown state-action pairs is kNK, since all NK state-action pairs will be marked as known after this. By Lemma 5, if the policy π_i is not 3ε-optimal in $\mathcal{M}'$, the algorithm will visit an unknown state-action pair with probability at least ε. Let m be the number of steps that π_i is not 3ε-optimal in $\mathcal{M}'$ over the course of training. We now show that $\Pr(m \leq C) \geq 1-\delta$. Let $\mathcal{S}$ be the number of successes of a binary random variable that occurs with probability ε sampled C/T times. Since $kNK \leq \varepsilon\frac{C}{T}$, we can apply Hoeffding's inequality to get that

$$\begin{aligned}\Pr(m \leq C) &\geq \Pr(\mathcal{S} > kNK)\\ &\geq 1-\exp(-\tfrac{2C}{T}(\varepsilon - \tfrac{kNKT}{C})^2)\\ &\geq 1-\delta.\end{aligned}$$

From the proof of Theorem 1, since $\mathcal{M}'$ is an $(\frac{\varepsilon}{NT}, T)$-approximation, π_i is 6ε-optimal in $\mathcal{M}$. Recalling that we assumed that all state-action pairs in $\widehat{\mathcal{M}}$ are $\frac{\varepsilon}{NT}$-accurate, which occurs with probability $1-\delta$, we have that the overall probability of producing a 9ε-optimal strategy is at least $(1-\delta)(1-\delta) \geq 1-2\delta$. □

Experiments

We implemented ω-PAC inside of the tool Mungojerrie (Hahn et al. 2023).[3] Mungojerrie can compute optimal policies with respect to a parity automaton in MDPs and is written in C++. All experiments were run on a computer with an Intel i7-8750H processor and 16 GB of memory.

Gridworld example. Figure 2 shows a gridworld example. In this example, the agent has four actions, north-east, north-west, south-east, and south-west. For a given direction, the agent moves in one of the corresponding cardinal directions with probability 0.4, in the other corresponding cardinal direction with probability 0.4, and does not move with probability 0.2. If the agent would move into a wall, it does not move. In the trap state denoted in red, the agent becomes stuck and all actions cause the agent to not move. The property is to visit the states s and g infinitely often, which is expressible in LTL as $\varphi = \mathbf{GF}s \wedge \mathbf{GF}g$. We set $\varepsilon = 1/20$, and $\delta = 1/10$. The product contains $|S| = 12$ states, $|A| = 4$ actions, and has a ε-recurrence time of $T = 19$. Our implementation of the ω-PAC algorithm takes approximately 40 minutes to terminate on this example, under the parameter selection for k suggested by Theorem 1. We did not observe a run where the resulting policy produced was not optimal under this parameter selection, suggesting that the k in Theorem 1 may be needlessly large in practice.

[3] Available at https://plv.colorado.edu/mungojerrie/omega-pac.

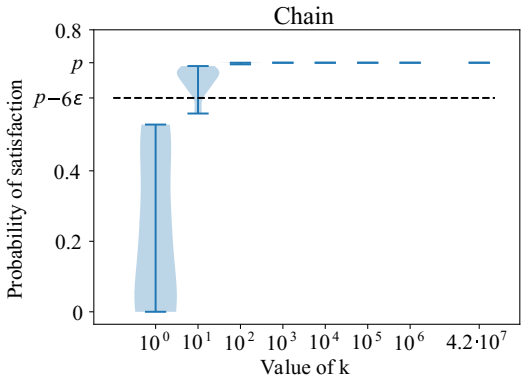

Figure 3: The distribution of the probability of satisfaction of learned policies for different values of k for the *chain example*. We also show the optimal probability of satisfaction p and the threshold for 6ε-optimality.

Chain example. To investigate the effect of different values of k on the performance of ω-PAC, we examined a simple MDP consisting of a chain of states with two actions: one action continues, and the other goes to an accepting sink with probability $1/2^s$ for the s^{th} state. In this example, $|S| = 8$, $|A| = 2$, $T = 8$, $\varepsilon = 1/60$, and $\delta = 1/10$. Figure 3 shows the distribution of probabilities of satisfaction of the policies produced by ω-PAC for 20 runs under different k, up to the k used in Theorem 1. We see that in practice, a small k typically suffices, and that results of this example are in line with Theorem 1.

Conclusion

We introduced ω-PAC, a PAC learning algorithm for LTL and ω-regular objectives in MDPs. For this algorithm, we introduced the notion of the ε-recurrence time. Intuitively, the ε-recurrence time measures the time it takes for finite trajectories to match the recurrent behavior of infinite trajectories with high probability. We proved that the ω-PAC algorithm has a sample complexity that is polynomial in the relevant parameters, the size of the state space $|S|$, the size of the action space $|A|$, the ε-recurrence time T, $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$. Finally, we performed experiments with ω-PAC that suggest that the bounds of our theory can be tightened as part of future work.

Acknowledgements

We thank Mahmoud Salamati and Suguman Bansal for valuable discussions. This work was supported in part by the NSF through grant CCF-2009022 and the NSF CAREER award CCF-2146563.

References

Alur, R.; Bansal, S.; Bastani, O.; and Jothimurugan, K. 2022. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, 604–624. Springer.

Alur, R.; Bastani, O.; Jothimurugan, K.; Perez, M.; Somenzi, F.; and Trivedi, A. 2023. Policy Synthesis and Reinforcement Learning for Discounted LTL. In *Computer Aided Verification*, 415–435.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Ashok, P.; Kretinsky, J.; and Weininger, M. 2019. PAC statistical model checking for Markov decision processes and stochastic games. In *International Conference on Computer Aided Verification*, 497–519. Springer.

Baier, C.; and Katoen, J.-P. 2008. *Principles of model checking*. MIT press.

Bozkurt, A. K.; Wang, Y.; Zavlanos, M. M.; and Pajic, M. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 10349–10355. IEEE.

Brafman, R. I.; and Tennenholtz, M. 2003. R-Max - a General Polynomial Time Algorithm for near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3.

Brázdil, T.; Chatterjee, K.; Chmelik, M.; Forejt, V.; Křetínskỳ, J.; Kwiatkowska, M.; Parker, D.; and Ujma, M. 2014. Verification of Markov decision processes using learning algorithms. In *Automated Technology for Verification and Analysis: 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings 12*, 98–114. Springer.

Daca, P.; Henzinger, T. A.; Kretinsky, J.; and Petrov, T. 2017. Faster statistical model checking for unbounded temporal properties. *ACM Transactions on Computational Logic (TOCL)*, 18(2): 1–25.

Duret-Lutz, A.; Lewkowicz, A.; Fauchille, A.; Michaud, T.; Renault, E.; and Xu, L. 2016. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, 122–129. Springer.

Fu, J.; and Topcu, U. 2014. Probably approximately correct MDP learning and control with temporal logic constraints. *arXiv preprint arXiv:1404.7073*.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2019. Omega-regular objectives in model-free reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I*, 395–412. Springer.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2020. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems: 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings, Part I*, 306–323. Springer.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2022. An Impossibility Result in Automata-Theoretic Reinforcement Learning. In *Automated Technology for Verification and Analysis*, volume 13505 of *Lecture Notes in Computer Science*, 42–57.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2023. Mungojerrie: Linear-Time Objectives in Model-Free Reinforcement Learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, 527–545.

Kakade, S. M. 2003. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom).

Kearns, M.; and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2): 209–232.

Littman, M. L.; Topcu, U.; Fu, J.; Isbell, C.; Wen, M.; and MacGlashan, J. 2017. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Silver, D.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529: 484–489.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142.

Voloshin, C.; Le, H.; Chaudhuri, S.; and Yue, Y. 2022. Policy optimization with linear temporal logic constraints. *Advances in Neural Information Processing Systems*, 35: 17690–17702.

Yang, C.; Littman, M.; and Carbin, M. 2021. On the (in) tractability of reinforcement learning for LTL objectives. *arXiv preprint arXiv:2111.12679*.