Reinforcement Learning for General LTL Objectives Is Intractable

Abstract

In recent years, researchers have made significant progress in devising reinforcement-learning algorithms for optimizing linear temporal logic (LTL) objectives and LTL-like objectives. Despite these advancements, there are fundamental limitations to how well this problem can be solved that previous studies have alluded to but, to our knowledge, have not examined in depth. In this paper, we address theoretically the hardness of learning with general LTL objectives. We formalize the problem under the probably approximately correct learning in Markov decision processes (PAC-MDP) framework, a standard framework for measuring sample complexity in reinforcement learning. In this formalization, we prove that the optimal policy for any LTL formula is PAC-MDP-learnable only if the formula is in the most limited class in the LTL hierarchy, consisting of only finite-horizon-decidable properties. Practically, our result implies that it is impossible for a reinforcement-learning algorithm to obtain a PAC-MDP guarantee on the performance of its learned policy after finitely many interactions with an unconstrained environment for non-finite-horizon-decidable LTL objectives.

1 Introduction

In reinforcement learning, we situate an autonomous agent in an unknown environment and specify an objective. We want the agent to learn the optimal behavior for achieving the specified objective by interacting with the environment.

Specifying an Objective. The objective for the agent is a specification of the possible trajectories of the overall system, consisting of the environment and the agent. Each trajectory is an infinite sequence of the states of the system, evolving through time. The objective specifies which trajectories are desirable so that the agent can distinguish the optimal behaviors from non-optimal behaviors.

The Reward Objective. One form of an objective is a reward function. A reward function specifies a scalar value, a reward, for each state of the system. The desired trajectories are those with higher cumulative discounted rewards. The reward-function objective is well studied (Sutton and Barto 1998). It has desirable properties that allow reinforcement-learning algorithms to provide performance guarantees on learned behavior (Strehl et al. 2006), meaning that algorithms can guarantee learning a behavior that achieves almost optimal cumulative discounted rewards with high probability. Due to its success, researchers have adopted the reward-function objective as the de facto standard of behavior specification in reinforcement learning.

The Temporal Logic Objective. However, reward engineering, the practice of encoding desirable behaviors into a reward function, is a difficult challenge in applied reinforcement learning (Dewey 2014; Littman et al. 2017). To reduce the burden of reward engineering, linear temporal logic (LTL) has attracted researchers' attention as an alternative objective. LTL is a formal logic used initially to specify program behaviors for system verification (Pnueli 1977).

An LTL formula is built from a set of propositions about the state of the environment, logical connectives, and temporal operators such as G (always) and F (eventually). Many reinforcement-learning tasks are naturally expressible with LTL (Littman et al. 2017). For some classic control (Brockman et al. 2016) examples, we can express 1. Cart-Pole as $\mathsf{G}\, up$ (i.e., the pole always stays up), 2. Mountain-Car as $\mathsf{F}\, goal$ (i.e., the car eventually reaches the goal), and 3. Pendulum-Swing-Up as $\mathsf{F}\,\mathsf{G}\, up$ (i.e., the pendulum eventually always stays up). Researchers have thus used LTL as an alternative objective specification for reinforcement-learning agents (Fu and Topcu 2014; Sadigh et al. 2014; Li, Vasile, and Belta 2017; Hahn et al. 2019; Hasanbeig et al. 2019; Bozkurt et al. 2020).

Given an LTL objective specified by an LTL formula, each trajectory of the system either satisfies or violates that formula. The agent should learn the behavior that maximizes the probability of satisfying that formula. Research has shown that using an LTL objective supports automated reward shaping (Jothimurugan, Alur, and Bastani 2019; Camacho et al. 2019; Jiang et al. 2020) and variance reduction (Gao et al. 2019).

A Trouble with Infinite Horizons. General LTL objectives are inherently infinite-horizons objectives. Specifically, these LTL objectives require looking at infinitely many steps of a trajectory to determine the satisfiability of those objectives. Despite the above developments on reinforcement learning with LTL objectives, as we will show, reinforcement-learning agents cannot identify a near-optimal behavior for such objectives with any confidence.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The intuition for the trouble is: Any finite number of interactions with an environment with unknown transition dynamics is insufficient to identify the environment dynamics perfectly. Further, an infinite horizon amplifies any identification inaccuracy of the environment dynamics. In particular, for these infinite-horizon objectives, a behavior's satisfaction probability under the inaccurate environment dynamics can be arbitrarily different from the behavior's satisfaction probability under the true environment dynamics. Consequently, a learner cannot guarantee with any confidence that it has identified near-optimal behavior for an infinite-horizon objective.

Relevant Works. This challenge of reinforcement learning for LTL objectives is alluded to – but not formally elaborated – in prior work (Henriques et al. 2012; Ashok, Křetínský, and Weininger 2019; Jiang et al. 2020), noting slow learning times for mastering infinite-horizon properties. Littman et al. (2017) provided a specific environment that illustrates the limitation, arguing for the use of a discounted variant of LTL. However, to our knowledge, no prior work formally examines the limitation[1].

Various works introduced approaches that we can interpret as workarounds to the hardness problem. For example, Henriques et al. (2012) considered objectives equivalent to finite-horizon objectives (i.e., the complement of infinite-horizon objectives) to obtain guarantees on the learned behavior. Ashok, Křetínský, and Weininger (2019) introduced a reinforcement-learning algorithm with best-effort guarantee for a subclass of infinite-horizon LTL objectives. Fu and Topcu (2014) and Brázdil et al. (2014) introduced reinforcement-learning algorithms with guarantees by making additional assumptions about the environment. On the other hand, our work identifies and formalizes the core hardness with which the above approaches are grappling. We connect to these approaches and categorize them by how they interact with our main theorem. We discuss our categorization and review these approaches in Section 6.2.

Though it is understood that, for infinite-horizon average-reward problems, reinforcement-learning algorithms do not have guarantees on the learned behavior without additional assumptions on the environment (Kearns and Singh 2004), a formal statement for what is achievable with LTL objectives has been lacking.

Contributions. We make the following contributions:

- A formalization of reinforcement learning with LTL objectives under the PAC-MDP framework (Fiechter 1994; Brafman and Tennenholtz 2002; Kearns and Singh 2004; Kakade 2003; Strehl et al. 2006), a standard framework for measuring sample complexity for reinforcement learning algorithms, and a formal definition of LTL-PAC-learnable, a learnability criterion for reinforcement learning with LTL objectives.
- A statement and proof of a theorem that an LTL formula is LTL-PAC-learnable if and only if it is finite-horizon-decidable. To that end, for any infinite-horizon LTL formula, we give a construction of two special families of MDPs as counterexamples with which we prove that the formula is not LTL-PAC-learnable.
- Experiments with current reinforcement-learning algorithms for LTL objectives that further empirically support our theoretical result.
- Identification of several categories of approaches that can work around the unlearnability problem of general LTL formulas and their connections to previous approaches.

[1]Concurrent to this work, Alur et al. (2021) also examine the hardness of LTL objectives. They state and prove a theorem that is a weaker version of the core theorem of this work. Specifically, their theorem identifies the hardness for one particular LTL objective, while our theorem identifies the hardness for all infinite-horizon LTL objectives. Their work was made public while this work was under conference review. We discuss their work in Appendix H.

2 Preliminaries: Reinforcement Learning

This section provides definitions for MDPs, planning, reinforcement learning, and PAC-MDP.

2.1 Markov Processes

We first review some basic notation for Markov processes.

A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, A, P, s_0)$, where S and A are finite sets of states and actions, $P\colon (S \times A) \to \Delta(S)$ is a transition probability function that maps a current state and an action to a distribution over next states, and $s_0 \in S$ is an initial state. The MDP is sometimes referred to as the *environment MDP* to distinguish it from any specific objective.

A (stochastic) *stationary policy* π for an MDP is a function $\pi\colon S \to \Delta(A)$ that maps each state of the MDP to a distribution over the actions.

A (stochastic) *non-stationary policy* π for an MDP is a function $\pi\colon ((S \times A)^* \times S) \to \Delta(A)$ that maps a history of states and actions of the MDP to a distribution over actions.

An MDP and a policy on the MDP induce a *discrete-time Markov chain* (DTMC). A DTMC is a tuple $\mathcal{D} = (S, P, s_0)$, where S is a finite set of states, $P\colon S \to \Delta(S)$ is a transition-probability function that maps a current state to a distribution over next states, and $s_0 \in S$ is an initial state. A *sample path* of $\mathcal{D}$ is an infinite sequence of states $w \in S^\omega$. The sample paths of a DTMC form a probability space.

2.2 Objective

An *objective* for an MDP $\mathcal{M} = (S, A, P, s_0)$ is a measurable function $\kappa\colon S^\omega \to \mathbb{R}$ on the probability space of the DTMC induced by $\mathcal{M}$ and a policy. The *value* of the objective for the MDP $\mathcal{M}$ and a policy π is the expectation of the objective under that probability space:

$$V^\pi_{\mathcal{M},\kappa} = \mathsf{E}_{w\sim\mathcal{D}}[\kappa(w)] \quad (\mathcal{D} \text{ induced by } \mathcal{M} \text{ and } \pi).$$

For example, the cumulative discounted rewards objective (Puterman 1994) with discount γ and a reward function $R\colon S \to \mathbb{R}$ is:

$$\kappa^{\text{reward}}(w) \triangleq \sum_{i=0}^{\infty} \gamma^i \cdot R(w[i]).$$

An *optimal policy* maximizes the objective's value: $\pi^* = \arg\max_\pi V^\pi_{\mathcal{M},\kappa}$. The *optimal value* $V^{\pi^*}_{\mathcal{M},\kappa}$ is then the objective value of the optimal policy. A policy π is *ϵ-optimal* if its value is ϵ-close to the optimal value: $V^\pi_{\mathcal{M},\kappa} \geq V^{\pi^*}_{\mathcal{M},\kappa} - \epsilon$.

2.3 Planning with a Generative Model

A planning-with-generative-model algorithm (Kearns, Mansour, and Ng 1999; Grill, Valko, and Munos 2016) has access to a *generative model*, a sampler, of an MDP's transitions but does not have direct access to the underlying probability values. It can take any state and action and sample a next state. It learns a policy from those sampled transitions.

Formally, a planning-with-generative-model algorithm $\mathcal{A}$ is a tuple $(\mathcal{A}^{\mathsf{S}}, \mathcal{A}^{\mathsf{L}})$, where $\mathcal{A}^{\mathsf{S}}$ is a *sampling algorithm* that drives how the environment is sampled, and $\mathcal{A}^{\mathsf{L}}$ is a *learning algorithm* that learns a policy from the samples obtained by applying the sampling algorithm.

In particular, the sampling algorithm $\mathcal{A}^{\mathsf{S}}$ is a function that maps from a history of sampled environment transitions $((s_0, a_0, s'_0) \dots (s_k, a_k, s'_k))$ to the next state and action to sample (s_{k+1}, a_{k+1}) , resulting in $\mathsf{s}'_{k+1} \sim \mathsf{P}(\cdot \mid s_{k+1}, a_{k+1})$. Iterative application of the sampling algorithm $\mathcal{A}^{\mathsf{S}}$ produces a sequence of sampled environment transitions.

The learning algorithm is a function that maps that sequence of sampled environment transitions to a non-stationary policy of the environment MDP. Note that the sampling algorithm can internally consider alternative policies as part of its decision of what to sample. Also, note that we deliberately consider non-stationary policies since the optimal policy for an LTL objective (defined later) is non-stationary in general (unlike a cumulative discounted rewards objective).

2.4 Reinforcement Learning

In reinforcement learning, an agent is situated in an environment MDP and can only sample from the current state of the environment. We can therefore view a reinforcement-learning algorithm as a special kind of planning-with-generative-model algorithm. In particular, a *reinforcement-learning algorithm* is a planning-with-generative-model algorithm $(\mathcal{A}^{\mathsf{S}}, \mathcal{A}^{\mathsf{L}})$ such that the sampling algorithm always follows the next state sampled from the environment and only has the choice for the next action.

Note that an alternative definition is to treat a reinforcement-learning algorithm as a non-stationary policy (Strehl et al. 2006). This view is equivalent to our definition since that non-stationary policy is how the next actions are chosen.

2.5 Probably Approximately Correct in MDPs

A successful planning-with-generative-model algorithm (or reinforcement-learning algorithm) should learn from the sampled environment transitions and produce an optimal policy for the objective in the environment MDP. However, since the environment transitions may be stochastic, we cannot expect an algorithm to always produce the optimal policy. Instead, we seek an algorithm that, with high probability, produces a nearly optimal policy. The probably approximately correct in Markov decision processes (PAC-MDP) framework (Fiechter 1994; Brafman and Tennenholtz 2002; Kearns and Singh 2004; Kakade 2003; Strehl et al. 2006), which takes inspiration from probably approximately correct (PAC) learning (Valiant 1984), formalizes this notion. The PAC-MDP framework requires an algorithm to be efficient in both sampling and algorithmic complexity. In this work, we only consider sample efficiency and thus omit the requirement on algorithmic complexity. Next, we generalize the PAC-MDP framework from reinforcement-learning with a reward objective to planning-with-generative-model with a generic objective.

Definition 1. Given an objective κ, a planning-with-generative-model algorithm $(\mathcal{A}^{\mathsf{S}}, \mathcal{A}^{\mathsf{L}})$ is *κ-PAC* (probably approximately correct for objective κ) in an environment MDP $\mathcal{M}$ if, with the sequence of transitions T of length N sampled using the sampling algorithm $\mathcal{A}^{\mathsf{S}}$, the learning algorithm $\mathcal{A}^{\mathsf{L}}$ outputs a non-stationary ϵ-optimal policy with probability at least $1 - \delta$ for any given $\epsilon > 0$ and $0 < \delta < 1$. That is:

$$\mathsf{P}_{\mathsf{T} \sim \langle \mathcal{M}, \mathcal{A}^{\mathsf{S}} \rangle_N} \left(V^{\mathcal{A}^{\mathsf{L}}(T)}_{\mathcal{M},\kappa} \geq V^{\pi^*}_{\mathcal{M},\kappa} - \epsilon \right) \geq 1 - \delta. \tag{1}$$

We use $\mathsf{T} \sim \langle \mathcal{M}, \mathcal{A}^{\mathsf{S}} \rangle_N$ to denote that the probability space is over the set of length-N transition sequences sampled from the environment $\mathcal{M}$ using the sampling algorithm $\mathcal{A}^{\mathsf{S}}$. For brevity, we will drop $\langle \mathcal{M}, \mathcal{A}^{\mathsf{S}} \rangle_N$ when it is clear from context and simply write $\mathsf{P}_{\mathsf{T}}(.)$ to denote that the probability space is over the sampled transitions.

Definition 2. Given an objective κ, a κ-PAC planning-with-generative-model algorithm is *sample efficiently κ-PAC* if the number of sampled transitions N is asymptotically polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|S|$, $|A|$.

Note that we allow the polynomial to have constant coefficients that depends on κ.

3 Linear Temporal Logic Objectives

This section describes LTL and its use as objectives.

3.1 Linear Temporal Logic

A linear temporal logic (LTL) formula is built from a finite set of atomic propositions Π, logical connectives $\neg, \wedge, \vee$, temporal next X, and temporal operators G (always), F (eventually), and U (until). Equation (2) gives the grammar of an LTL formula ϕ over the set of atomic propositions Π:

$$\phi := a \,\big|\, \neg\phi \,\big|\, \phi \wedge \phi \,\big|\, \phi \vee \phi \,\big|\, \mathsf{X}\phi \,\big|\, \mathsf{G}\,\phi \,\big|\, \mathsf{F}\,\phi \,\big|\, \phi\,\mathsf{U}\,\phi, \;\; a \in \Pi. \tag{2}$$

LTL is a logic over infinite-length words. Informally, these temporal operators have the following meanings: $\mathsf{X}\phi$ asserts that ϕ is true at the next time step; $\mathsf{G}\,\phi$ asserts that ϕ is always true; $\mathsf{F}\,\phi$ asserts that ϕ is eventually true; $\psi\,\mathsf{U}\,\phi$ asserts that ψ needs to stay true until ϕ eventually becomes true. We give the formal semantics of each operator in Appendix A.2. We write $w \vDash \phi$ to denote that the infinite word w satisfies ϕ.

3.2 MDP with LTL Objectives

In this paper, we consider LTL objectives. An LTL objective maximizes the probability of satisfying an LTL formula.

An *LTL specification* for an MDP is a tuple $(\mathcal{L}, \phi)$, where $\mathcal{L} \colon S \to 2^\Pi$ is a labeling function, and ϕ is an LTL formula

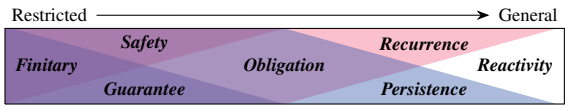

Figure 1: The hierarchy of LTL

over atomic propositions Π. The labeling function is a classifier that maps each state in the MDP to a tuple of truth values of the set of atomic propositions in ϕ. For a sample path w, we use $\mathcal{L}(w)$ to denote the element-wise application of $\mathcal{L}$ on w.

The LTL objective ξ specified by the LTL specification is the satisfaction of the formula ϕ of a sample path mapped by the labeling function $\mathcal{L}$, that is: $\kappa(w) \triangleq \mathbb{1}_{\mathcal{L}(w) \vDash \phi}$. The value of this objective is called the *satisfaction probability* of ξ:

$$V^{\pi}_{\mathcal{M},\xi} = \mathsf{P}_{w \sim \mathcal{D}}(\mathcal{L}(w) \vDash \phi) \quad (\mathcal{D} \text{ induced by } \mathcal{M} \text{ and } \pi).$$

3.3 Infinite Horizons in LTL Objectives

An LTL formula describes either a finite-horizon or infinite-horizon property. Manna and Pnueli (1987) classified LTL formulas into seven classes, as shown in Figure 1. Each class includes all the classes to the left of that class (e.g., $\mathit{Finitary} \subset \mathit{Guarantee}$, but $\mathit{Safety} \not\subset \mathit{Guarantee}$), with the *Finitary* class being the most restricted and the *Reactivity* class being the most general. Below we briefly describe the key properties of the leftmost three classes relevant to the core of this paper. We present a complete description of all the classes in Appendix A.2.

- $\phi \in \mathit{Finitary}$ iff there exists a horizon H such that infinite length words sharing the same prefix of length H are either all accepted or all rejected by ϕ. E.g., $a \wedge \mathsf{X} a$ (i.e., a is true for two steps) is in *Finitary*.
- $\phi \in \mathit{Guarantee}$ iff there exists a language of finite words L (i.e., a Boolean function on finite length words) such that $w \vDash \phi$ if L accepts a prefix of w. Informally, a formula in *Guarantee* asserts that something eventually happens. E.g., $\mathsf{F}\, a$ (i.e., eventually a is true) is in *Guarantee*.
- $\phi \in \mathit{Safety}$ iff there exists a language of finite words L such that $w \vDash \phi$ if L accepts all prefixes of w. Informally, a formula in *Safety* asserts that something always happens. E.g., $\mathsf{G}\, a$ (i.e., a is always true) is in *Safety*.

Moreover, *Finitary* is the intersection of *Guarantee* and *Safety*. Any $\phi \in \mathit{Finitary}$, or equivalently $\phi \in \mathit{Guarantee} \cap \mathit{Safety}$, inherently describes finite-horizon properties. Any $\phi \notin \mathit{Finitary}$, or equivalently $\phi \in \mathit{Guarantee}^{\complement} \cup \mathit{Safety}^{\complement}$, inherently describes infinite-horizon properties. As we will show, reinforcement-learning algorithms cannot provide PAC guarantees for LTL objectives specified by formulas that describe infinite-horizon properties.

3.4 Intuition of the Problem

Suppose that we send an agent into one of the MDPs in Figure 2, and want its behavior to satisfy "eventually reach the state h", expressed as the LTL formula $\mathsf{F}\, h$. The optimal behavior is to always choose the action along the transition

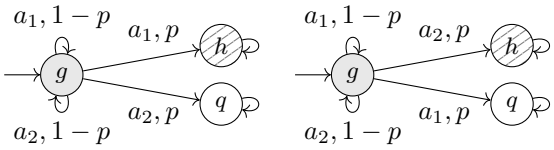

Figure 2: Two MDPs parameterized by p in range $0 < p < 1$. Action a_1 in the MDP on the left and action a_2 in the MDP on the right have probability p of transitioning to the state h. Conversely, action a_2 in the MDP on the left and action a_1 in the MDP on the right have probability p of transitioning to the state q. Both actions in both MDPs have probability $1-p$ to loop around the state g.

$g \to h$ for both MDPs (i.e., a_1 for the MDP on the left and a_2 for the MDP on the right). This optimal behavior satisfies the objective with probability one. However, the agent does not know which of the two MDPs it is in. The agent must follow its sampling algorithm to explore the MDP's dynamics and use its learning algorithm to learn this optimal behavior.

If the agent observes neither transitions going out of g (i.e., $g \to h$ or $g \to q$) during sampling, it will not be able to distinguish between the two actions. The best it can do is a 50% chance guess and cannot provide any non-trivial guarantee on the probability of learning the optimal action.

On the other hand, if the agent observes one of the transitions going out of g, it will be able to determine which action leads to state h, thereby learning always to take that action. However, the probability of observing any such transition with N interactions is at most $1-(1-p)^N$. This is problematic: with any finite N, there always exists a value of p such that this probability is arbitrarily close to 0. In other words, with any finite number of interactions, without knowing the value of p, the agent cannot guarantee (a non-zero lower bound on) its chance of learning a policy that satisfies the LTL formula $\mathsf{F}\, h$.

Further, the problem is not limited to this formula. For example, the objective "never reach the state q", expressed as the LTL formula $\mathsf{G} \neg q$, has the same problem in these two MDPs. More generally, for any LTL formula that describes an infinite-horizon property, we construct two counterexample MDPs with the same nature as the ones in Figure 2, and prove that it is impossible to provide a guarantee on the learning of the optimal behavior.

4 LTL Objectives Are Not Learnable

By specializing the κ-PAC definitions for (Definitions 1 and 2) with the definition of LTL objectives in Section 3.2, we obtain the following definitions of LTL-PAC.

Definition 3. Given an LTL objective ξ, a planning-with-generative-model algorithm $(\mathcal{A}^{\mathsf{S}}, \mathcal{A}^{\mathsf{L}})$ is *LTL-PAC* (probably approximated correct for LTL objective ξ) in an environment MDP $\mathcal{M}$ for the LTL objective ξ if, with the sequence of transitions T of length N sampled using the sampling algorithm $\mathcal{A}^{\mathsf{S}}$, the learning algorithm $\mathcal{A}^{\mathsf{L}}$ outputs a non-stationary ϵ-optimal policy with a probability of at least

$1-\delta$ for all $\epsilon > 0$ and $0 < \delta < 1$. That is,

$$\mathsf{P}_{\mathsf{T}\sim\langle\mathcal{M},\mathcal{A}^{\mathrm{S}}\rangle_N}\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M},\xi} \geq V^{\pi^*}_{\mathcal{M},\xi} - \epsilon\right) \geq 1-\delta. \tag{3}$$

We call the probability on the left of the inequality the *LTL-PAC probability* of the algorithm $(\mathcal{A}^{\mathrm{S}}, \mathcal{A}^{\mathrm{L}})$.

Definition 4. Given an LTL objective ξ, an LTL-PAC planning-with-generative-model algorithm for ξ is *sample efficiently* LTL-PAC if the number of sampled transitions N is asymptotically polynomial to $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|S|$, $|A|$.

With the above definitions, we are now ready to define the PAC learnability of an LTL objective and state the main theorem of the paper.

Definition 5. An LTL formula ϕ over atomic propositions Π is *LTL-PAC-learnable by planning-with-generative-model (reinforcement-learning)* if there exists a sample efficiently LTL-PAC planning-with-generative-model (reinforcement-learning) algorithm for all environment MDPs and all consistent labeling functions $\mathcal{L}$ (that is, $\mathcal{L}$ maps from the MDP's states to 2^{Π}) for the LTL objective specified by $(\mathcal{L}, \phi)$.

Theorem 1. *An LTL formula ϕ is LTL-PAC-learnable by reinforcement-learning (planning-with-generative-model) if (and only if) ϕ is in Finitary.*

Between the two directions of Theorem 1, the forward direction ("only if") is the more important. The forward direction states that for any LTL formula not in *Finitary* (that is, infinite-horizon properties), there does not exist a planning-with-generative-model algorithm—which by definition also excludes any reinforcement-learning algorithm—that is sample efficiently LTL-PAC for all environments. This result is the core contribution of the paper—general LTL formulas are not sample efficiently LTL-PAC-learnable.

On the other hand, the reverse direction of Theorem 1 states that, for any LTL formula in *Finitary* (finite-horizon properties), there exists a reinforcement-learning algorithm—which by definition is also a planning-with-generative-model algorithm—that is sample efficiently LTL-PAC for all environments.

4.1 Theorem 1: Proof Outline

We next prove the forward direction of Theorem 1 and reserve the proof of the reverse direction for Appendix E.

- **MDP Family.** We construct a family of pairs of MDPs. The key design behind each pair in the family is, as we will show, that no planning-with-generative-model algorithm can learn a policy that is simultaneously ϵ-optimal on both MDPs without observing a number of samples that depends on the probability of a specific transition.
- **Sample complexity of $\mathsf{F}\, h_0$.** For the singular case of the LTL formula $\mathsf{F}\, h_0$, we derive a sample complexity lower bound for any LTL-PAC planning-with-generative-model algorithm applied to our family of MDPs. This lower bound necessarily depends on a specific transition probability in the MDPs.
- **Sample complexity of Non-finitary LTL formulas.** We reduce the problem of learning the optimal satisfying policy for $\mathsf{F}\, h_0$ to the problem of learning the optimal satisfying policy for any non-finitary LTL formula (those not in the class *Finitary*). The key observation is that, from the structure of the formula itself, we can identify a pair of MDPs in our family for which the lower bound must hold. We then adapt our established bounds for $\mathsf{F}\, h_0$ to these identified MDPs.

Altogether, our approach proves that learning the optimal policy for any non-finitary LTL formula has a lower bound that may depend on a transition probability in a provided MDP. Because our definition of LTL-PAC requires an algorithm's sample complexity to be independent of the transition probabilities of the MDP, we conclude that non-finitary LTL formulas are not LTL-PAC-learnable.

4.2 MDP Family

We first give two constructions of parameterized counterexample MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ shown in Figure 3. The key design behind each pair in the family is that no planning-with-generative-model algorithm can learn a policy that is simultaneously ϵ-optimal on both MDPs without observing a number of samples that depends on the probability of a specific transition.

Both MDPs are parameterized by the shape parameters k, l, u, v, m, n, and an unknown transition probability parameter p. The actions are $\{a_1, a_2\}$, and the state space is partitioned into three regions: states $g_{0\ldots l}$ (the grey states), states $h_{0\ldots v}$ (the line-hatched states), and states $q_{0\ldots n}$ (the white states). All transitions, except $g_l \to h_0$ and $g_l \to q_0$, are the same between $\mathcal{M}_1$ and $\mathcal{M}_2$. The effect of this difference between the two MDPs is that, for $\mathcal{M}_i$:

- Action a_i in $\mathcal{M}_i$ at the state g_l will transition to the state h_0 with probability p, inducing a run that cycles in the region $h_{u\ldots v}$ forever.
- Action a_{3-i} (the alternative to a_i) in $\mathcal{M}_i$ at the state g_l will transition to the state q_0 with probability p, inducing a run that cycles in the region $q_{m\ldots n}$ forever.

Further, for any policy, a run of the policy on both MDPs must eventually reach h_0 or q_0 with probability 1, and ends in an infinite cycle in either $h_{u\ldots v}$ or $q_{m\ldots n}$.

4.3 Sample Complexity of $\mathsf{F}\, h_0$

We first consider the LTL objective ξ^{h_0} specified by the LTL formula $\mathsf{F}\, h_0$ and the labeling function $\mathcal{L}^{h_0}$ that labels only the state h_0 as *true*. A sample path on the MDPs (Figure 3) satisfies this objective iff the path reaches the state h_0.

Given $\epsilon > 0$ and $0 < \delta < 1$, our goal is to derive a lower bound on the number of sampled environment transitions performed by an algorithm, so that the satisfaction probability of π, the learned policy, is ϵ-optimal ($V^{\pi}_{\mathcal{M},\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M},\xi^{h_0}} - \epsilon$) with probability at least $1-\delta$.

The key rationale behind the following lemma is that, if a planning-with-generative-model algorithm has not observed any transition to either h_0 or q_0, the learned policy cannot be ϵ-optimal in both $\mathcal{M}_1$ and $\mathcal{M}_2$.

Lemma 2. *For any planning-with-generative-model algorithm $(\mathcal{A}^S, \mathcal{A}^L)$, it must be the case that:* $\min(\zeta_1, \zeta_2) \leq \frac{1}{2}$, *where* $\zeta_i = \mathsf{P}_{\mathsf{T}}\left(V^{\mathcal{A}^L(T)}_{\mathcal{M}_i,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_i,\xi^{h_0}} - \epsilon \,\middle|\, n(T) = 0\right)$ *and*

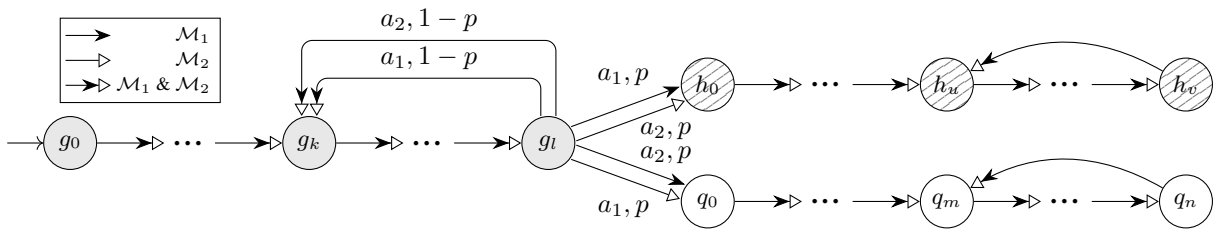

Figure 3: Counterexample MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, with transitions distinguished by arrow types (see legend). Both MDPs are parameterized by the parameter p that is in range $0 < p < 1$. Unlabeled edges are deterministic (actions a_1 and a_2 transition with probability 1). Ellipsis indicates a deterministic chain of states.

$n(T)$ is the number of transitions in T that start from g_l and end in either h_0 or q_0.

The value ζ_i is the LTL-PAC probability of a learned policy on $\mathcal{M}_i$, given that the planning-with-generative-model algorithm did not observe any information that allows the algorithm to distinguish between $\mathcal{M}_1$ and $\mathcal{M}_2$.

Proof. We present a proof of Lemma 2 in Appendix B. □

A planning-with-generative-model algorithm cannot learn an ϵ-optimal policy without observing a transition to either h_0 or q_0. Therefore, we bound the sample complexity of the algorithm from below by the probability that the sampling algorithm does observe such a transition:

Lemma 3. *For the LTL objective ξ^{h_0}, the number of samples, N, for a planning-with-generative-model algorithm to be LTL-PAC in both $\mathcal{M}_1$ and $\mathcal{M}_2$ (for any instantiation of the parameters k, l, u, v, m, n) has a lower bound of $N \geq \frac{\log(2\delta)}{\log(1-p)}$.*

Below we give a proof sketch of Lemma 3; we give the complete proof in Appendix C.

Proof Sketch of Lemma 3. First, we assert that the two inequalities of Equation (3) for both $\mathcal{M}_1$ and $\mathcal{M}_2$ holds true for a planning-with-generative-model algorithm. Next, by conditioning on $n(T) = 0$, plugging in the notation of ζ_i, and relaxing both inequalities, we get $(1-\zeta_i)\mathsf{P}_{\mathsf{T}}(n(T) = 0) \leq \delta$, for $i \in \{1, 2\}$. Then, since $n(T) = 0$ only occurs when all transitions from g_l end in g_k, we have $\mathsf{P}_{\mathsf{T}}(n(T) = 0) \geq (1-p)^N$. Combining the inequalities, we get $(1 - \min(\zeta_1, \zeta_2))(1-p)^N \leq \delta$. Finally, we apply Lemma 2 to get the desired lower bound of $N \geq \frac{\log(2\delta)}{\log(1-p)}$. □

4.4 Sample Complexity of Non-finitary Formulas

The goal of this section is to generalize our lower bound on $\mathsf{F}\, h_0$ to all non-finitary LTL formulas. The key observation we make is that for any non-finitary LTL formula, we can construct a pair of MDPs, $\mathcal{M}_1$ and $\mathcal{M}_2$, from our MDP family, whose existence implies our established lower bound.

We will use $[w_1; w_2; \ldots w_n]$ to denote the concatenation of the finite length words $w_1 \ldots w_n$. We will use w^i to denote the repetition of the finite length word w by i times, and w^∞ to denote the infinite repetition of w.

Definition 6. An accepting (resp. rejecting) infinite length word $[w_a; w_b^\infty]$ of ϕ is *uncommittable* if there exists finite length words w_c, w_d such that ϕ rejects (resp. accepts) $[w_a; w_b^i; w_c; w_d^\infty]$ for all $i \in \mathbb{N}$.

Lemma 4. *If ϕ has an uncommittable word w, there is an instantiation of $\mathcal{M}_1$ (or $\mathcal{M}_2$) in Figure 3 and a labeling function $\mathcal{L}$, such that, for any policy, the satisfaction probabilities of that policy in $\mathcal{M}_1$ (or $\mathcal{M}_2$) for the LTL objectives specified by $(\mathcal{L}, \phi)$ and $(\mathcal{L}^{h_0}, \mathsf{F}\, h_0)$ are the same.*

Proof. For an uncommittable word w, we first find the finite length words w_a,w_b,w_c,w_d according to Definition 6. We then instantiate $\mathcal{M}_1$ and $\mathcal{M}_2$ in Figure 3 as follows.

- If w is an uncommittable accepting word, we set k, l, u, v, m, n (Figure 3) to $|w_a|$, $|w_a| + |w_b|$, 0, $|w_b|$, $|w_c|$ and $|w_c| + |w_d|$, respectively. We then set the labeling function as in Equation (4).
- If w is an uncommittable rejecting word, we set k, l, u, v, m, n (Figure 3) to $|w_a|$, $|w_a| + |w_b|$, $|w_c|$, $|w_c| + |w_d|$, 0 and $|w_b|$, respectively. We then set the labeling function as in Equation (5).

$$\mathcal{L}(s) = \begin{cases} [w_a; w_b][j] & \text{if } s = g_j \\ w_b[j] & \text{if } s = h_j \\ [w_c; w_d][j] & \text{if } s = q_j \end{cases} \quad (4) \qquad \mathcal{L}(s) = \begin{cases} [w_a; w_b][j] & \text{if } s = g_j \\ [w_c; w_d][j] & \text{if } s = h_j \\ w_b[j] & \text{if } s = q_j \end{cases} \quad (5)$$

In words, for an uncommittable accepting word, we label the states $g_{0\ldots l}$ one-by-one by $[w_a; w_b]$; we label the states $h_{0\ldots v}$ one-by-one by w_b (and set $u = 0$, which eliminates the chain of states $h_{0\ldots u}$); we label the states $q_{0\ldots n}$ one-by-one by $[w_c; w_d]$. Symmetrically, for an uncommittable rejecting word, we label the states $g_{0\ldots l}$ one-by-one by $[w_a; w_b]$; we label the states $h_{0\ldots v}$ one-by-one by $[w_c; w_d]$; we label the states $q_{0\ldots n}$ one-by-one by w_b (and set $m = 0$, which eliminates the chain of states $q_{0\ldots m}$).

By the above instantiation, the two objectives specified by $(\mathcal{L}, \phi)$ and $(\mathcal{L}^{h_0}, \mathsf{F}\, h_0)$ are equivalent in $\mathcal{M}_1$ and $\mathcal{M}_2$. In particular, any path in $\mathcal{M}_1$ or $\mathcal{M}_2$ satisfies the LTL objective specified by $(\mathcal{L}, \phi)$ if and only if the path visits the state h_0 and therefore also satisfies the LTL objective specified by $(\mathcal{L}^{h_0}, \mathsf{F}\, h_0)$. Therefore, any policy must have the same satisfaction probability for both objectives. □

Lemma 5. *For $\phi \notin$ Finitary, the number of samples for a planning-with-generative-model algorithm to be LTL-PAC has a lower bound of $N \geq \frac{\log(2\delta)}{\log(1-p)}$.*

Proof. A corollary of Lemma 4 is: for any ϕ that has an uncommittable word, we can construct a pair of MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ in the family of pairs of MDPs in Figure 3, such that, in both MDPs, a policy is sample efficiently LTL-PAC for the LTL objective specified by $(\mathcal{L}, \phi)$ if it is sample efficiently LTL-PAC for the LTL objective specified by $(\mathcal{L}^{h_0}, \mathsf{F}\, h_0)$. This property implies that the lower bound in Lemma 3 for the objective specified by $(\mathcal{L}^{h_0}, \mathsf{F}\, h_0)$ also applies to the objective specified by $(\mathcal{L}, \phi)$, provided that any LTL formula $\phi \notin \mathit{Finitary}$ has an uncommittable word. In Appendix D, we prove a lemma that any LTL formula $\phi \notin \mathit{Guarantee}$ has an uncommittable accepting word, and any LTL formula $\phi \notin \mathit{Safety}$ has an uncommittable rejecting word. Since $\mathit{Finitary}$ is the intersection of $\mathit{Guarantee}$ and Safety, this completes the proof. □

4.5 Conclusion

Note that the lower bound $N \geq \frac{\log(2\delta)}{\log(1-p)}$ depends on p, the transition probability in the constructed MDPs. Moreover, for $\delta < \frac{1}{2}$, as p approaches 0, this lower bound goes to infinity. As a result, the bound does not satisfy the definition of sample efficiently LTL-PAC planning-with-generative-model algorithm for the LTL objective (Definition 2), and thus no algorithm is sample efficiently LTL-PAC. Therefore, LTL formulas not in $\mathit{Finitary}$ are not LTL-PAC-learnable. This argument proves the forward direction of Theorem 1.

5 Empirical Justifications

In this section, we empirically demonstrate our main result. Previous work has introduced various reinforcement-learning algorithms for LTL objectives (Sadigh et al. 2014; Hahn et al. 2019; Hasanbeig et al. 2019; Bozkurt et al. 2020). We therefore ask the research question: *Do the sample complexities for reinforcement-learning algorithms for LTL objectives introduced by previous works depend on the transition probabilities of the environment?*

Methodology. We consider a set of recent reinforcement-learning algorithms for LTL objectives (Hahn et al. 2019; Bozkurt et al. 2020), about which we give more details in Appendix F. These algorithms are all implemented in the Mungojerrie toolbox (Hahn et al. 2021).

We consider two pairs of LTL formulas and environment MDPs. The first pair is the formula $\mathsf{F}\, h$ and the counterexample MDP constructed as in Section 4.2. The second pair is an LTL formula and an MDP adapted from a case study from previous work (Sadigh et al. 2014). In this section, we focus on the first pair and defer the complete methodology and results to Appendix F.

We ran the considered algorithms on the chosen MDP with a range of values for the parameter p and let the algorithms perform N environment samples. We vary p by a geometric progression from 10^{-1} to 10^{-3} in 5 steps: $p(i) = 10^{-\frac{i+1}{2}}$ for $1 \leq i \leq 5$. We vary N by a geometric progression from 10^1 to 10^5 in 21 steps: $N(j) = 10^{\frac{j+4}{5}}$ for $1 \leq j \leq 21$. For each algorithm and each pair of values of p and N, we fix $\epsilon = 0.1$ and repeatedly run the algorithm to obtain a Monte Carlo estimation of the LTL-PAC probability (left side of Equation (3)) for that setting of p, N and ϵ. We repeat each setting until the estimated standard deviation of the estimated probability is within 0.01. In the end, for each algorithm we obtain $5 \times 21 = 105$ LTL-PAC probabilities and their estimated standard deviations.

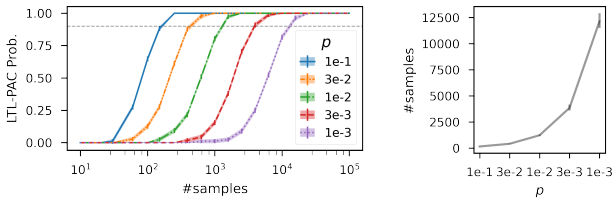

Figure 4: Left: LTL-PAC probabilities vs. number of samples, varying parameters p. Right: number of samples needed to reach 0.9 LTL-PAC probability vs. parameter p.

Results. Figure 4 presents the results for the algorithm in Bozkurt et al. (2020). Appendix F presents the results for other algorithms. On the left, we plot the LTL-PAC probabilities vs. the number of samples N, one curve for each p. On the right, we plot the intersections of the curves in the left plot with a horizontal cutoff of 0.9.

As we see from the left plot of Figure 4, for each p, the curve starts at 0 and grows to 1 in a sigmoidal shape as the number of samples increases. However, as p decreases, the MDP becomes harder: As shown on the right plot of Figure 4, the number of samples required to reach a particular LTL-PAC probability grows exponentially. Results for other algorithms are similar and lead to the same conclusion.

Conclusion. Since, in practice, the transition probabilities (p here) are unknown, one can't know which curve in the left plot a given environment will follow. Therefore, given any finite number of samples, these reinforcement-algorithms cannot provide guarantees on the LTL-PAC probability of the learned policy. This result supports Theorem 1.

6 Discussion

This section puts the hardness result in perspective.

6.1 Consequence of the Core Theorem

In Section 4, we derived a lower bound on the number of samples for any reinforcement-learning algorithm to obtain an LTL-PAC guarantee. Unfortunately, this lower bound depends on the transition probabilities of the MDP. Since the transition probabilities are not known to a reinforcement-learning algorithm—before, during, or after learning—with finitely many samples, no reinforcement-learning algorithm can give a non-trivial guarantee on the performance of its learned policy's potential error and uncertainty (that is, the ϵ and δ in LTL-PAC definition) for an arbitrary MDP. Consequently, given any arbitrarily large finite sample of transitions, the learned policy need not perform near optimally.

This consequence is unacceptable in applications that require strong guarantees of the overall system's behavior, such as traffic control, robotics, and autonomous vehi-

cles (Temizer et al. 2010; Kober, Bagnell, and Peters 2013; Schwarting, Alonso-Mora, and Rus 2018).

6.2 Directions Forward

In this section, we discuss several categories of approaches that use LTL or LTL-like specifications and work around the inherent hardness of reinforcement learning with LTL objectives. For each category, we classify approaches in the existing literature belonging to that category. We obtain the first category by using the reverse direction of Theorem 1, and each of the other categories by relaxing a specific requirement that Theorem 1 places on the learning algorithm.

Use a Finitary Objective. By Theorem 1, one can obtain an LTL-PAC guarantee for finitary LTL formulas. The *Finitary* class corresponds to those properties that are decidable by observing only a finite horizon of the input.

Researchers have introduced specification languages that express finitary properties and have applied reinforcement learning to objectives expressed in these languages (Henriques et al. 2012; Jothimurugan, Alur, and Bastani 2019). One value proposition of these approaches is that they provide succinct specifications because finitary properties written in LTL directly are verbose. For example, consider the finitary property "a holds for 100 steps"; the finitary LTL formula for this property is a conjunction of 100 terms: $a \wedge \mathsf{X}a \wedge \cdots \wedge (\underbrace{\mathsf{X}\ldots\mathsf{X}}_{99 \text{ times}} a)$.

For these succinct specification languages, by the reduction of these languages to finitary properties and the reverse direction of Theorem 1, there exist reinforcement-learning algorithms that give LTL-PAC guarantees.

Best-effort Guarantee. The definition of an LTL-PAC reinforcement-learning algorithm (Definition 3) requires the algorithm to produce a policy with satisfaction probability within ϵ of optimal, for all $\epsilon > 0$. This quantification follows from the standard definition of PAC-learnability (Valiant 1984). It is desirable because the specification is for the algorithm to learn a policy with a satisfaction probability that is arbitrarily close to the optimal. However, it is possible to relax this quantification over ϵ such that the algorithm only returns a policy with the best-available ϵ it can find.

For example, Ashok, Křetínský, and Weininger (2019) introduced a reinforcement-learning algorithm for objectives in the *Guarantee* class. Using a specified time budget, the algorithm returns a policy and an ϵ. Notably, it is possible for the returned ϵ to be 1, denoting that the learned policy is a maximally poor approximation of the optimal policy.

Know More About the Environment. The definition of an LTL-PAC reinforcement-learning algorithm (Definition 3) requires the algorithm to provide a guarantee for all environments. Providing a guarantee for all environments is desirable because, as is standard in reinforcement learning, the environment MDP's transition probabilities are unknown, and in practice an algorithm may be deployed in any possible MDP. However, on occasion, one can have prior information on the transition probabilities of the MDP at hand. In this case, the algorithm need only to provide a guarantee for MDPs that are consistent with the prior information.

For example, Fu and Topcu (2014) introduced a reinforcement-learning algorithm with a PAC-MDP guarantee that depends on the time horizon until the MDP reaches a steady state. Given an MDP, this time horizon is generally unknown; however, if one has knowledge of this time horizon *a priori*, it constrains the set of MDPs and yields an LTL-PAC guarantee dependent on this time horizon.

As another example, Brázdil et al. (2014) introduced a reinforcement-learning algorithm that provides an LTL-PAC guarantee provided a declaration of the minimum transition probability of the MDP. This constraint, again, bounds the space of considered MDPs.

Use an LTL-like Objective. Theorem 1 only considers LTL objectives. However, one opportunity for obtaining an LTL-PAC guarantee is to change the problem—use a specification language that is LTL-like, defining similar temporal operators, but also giving those operators a different semantics (Littman et al. 2017; Li, Vasile, and Belta 2017; Giacomo et al. 2019; Camacho et al. 2019).

For example, one line of work (Sadigh et al. 2014; Hahn et al. 2019; Hasanbeig et al. 2019; Bozkurt et al. 2020) uses LTL formulas as the objective, but also introduces one or more hyper-parameters λ to relax the formula's semantics. The reinforcement-learning algorithms in these works learn a policy for the environment MDP given fixed values of the hyper-parameters. Moreover, as hyper-parameter values approach a limit point, the learned policy becomes optimal for the hyper-parameter-free LTL formula.[2]

The relationship between these relaxed semantics and the original LTL semantics is analogous to the relationship between discounted and average-reward infinite-horizon MDPs. Specifically, the relaxed semantics (resp. discounted MDPs) approaches the LTL semantics (resp. the average-reward MDPs) in the limit of the hyper-parameter values (resp. discount factors) as shown by Bozkurt et al. (2020) (resp. Puterman (1994)). Since discounted MDPs are PAC-MDP-learnable (Strehl et al. 2006), we conjecture that these relaxed LTL specifications (at any fixed hyper-parameter setting) are LTL-PAC-learnable.

7 Conclusion

In this work, we formally proved that infinite-horizon LTL objectives in reinforcement learning cannot be learned in unrestricted environments. By inspecting the core result, we have identified various possible directions forward for future research. Our work resolves the apparent lack of a formal treatment of this fundamental limitation of infinite-horizon objectives, helps increase the community's awareness of this problem, and will help organize the community's efforts in reinforcement learning with LTL objectives.

[2] In general, Bozkurt et al. (2020) showed that there exists a critical setting of the parameters λ^* that produces the optimal policy. However, λ^* depends on the transition probabilities of the MDP and is therefore consistent with the findings we presented here.

References

Alur, R.; Bansal, S.; Bastani, O.; and Jothimurugan, K. 2021. A Framework for Transforming Specifications in Reinforcement Learning. *arXiv preprint arXiv:2111.00272*.

Ashok, P.; Křetínský, J.; and Weininger, M. 2019. PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games. In *Computer Aided Verification*.

Bozkurt, A.; Wang, Y.; Zavlanos, M.; and Pajic, M. 2020. Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning. In *International Conference on Robotics and Automation*.

Brafman, R. I.; and Tennenholtz, M. 2002. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3.

Brázdil, T.; Chatterjee, K.; Chmelík, M.; Forejt, V.; Křetínský, J.; Kwiatkowska, M.; Parker, D.; and Ujma, M. 2014. Verification of Markov Decision Processes Using Learning Algorithms. In *Automated Technology for Verification and Analysis*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.

Dewey, D. 2014. Reinforcement Learning and the Reward Engineering Principle. In *AAAI Spring Symposia*.

Fiechter, C.-N. 1994. Efficient Reinforcement Learning. In *Conference on Computational Learning Theory*.

Fu, J.; and Topcu, U. 2014. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In *Robotics: Science and Systems X*.

Gao, Q.; Hajinezhad, D.; Zhang, Y.; Kantaros, Y.; and Zavlanos, M. M. 2019. Reduced Variance Deep Reinforcement Learning with Temporal Logic Specifications. In *International Conference on Cyber-Physical Systems*.

Giacomo, G. D.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In *International Conference on Automated Planning and Scheduling*.

Grill, J.-B.; Valko, M.; and Munos, R. 2016. Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning. In *International Conference on Neural Information Processing Systems*.

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*.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2021. Mungojerrie: Reinforcement Learning of Linear-Time Objectives. *arXiv preprint arXiv:2106.09161*.

Hasanbeig, M.; Kantaros, Y.; Abate, A.; Kroening, D.; Pappas, G.; and Lee, I. 2019. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. In *Conference on Decision and Control*.

Henriques, D.; Martins, J. G.; Zuliani, P.; Platzer, A.; and Clarke, E. M. 2012. Statistical Model Checking for Markov Decision Processes. In *International Conference on Quantitative Evaluation of Systems*.

Jiang, Y.; Bharadwaj, S.; Wu, B.; Shah, R.; Topcu, U.; and Stone, P. 2020. Temporal-Logic-Based Reward Shaping for Continuing Learning Tasks. *arXiv preprint arXiv:2007.01498*.

Jothimurugan, K.; Alur, R.; and Bastani, O. 2019. A Composable Specification Language for Reinforcement Learning Tasks. In *International Conference on Neural Information Processing Systems*.

Kakade, S. M. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 1999. Approximate Planning in Large POMDPs via Reusable Trajectories. In *International Conference on Neural Information Processing Systems*.

Kearns, M.; and Singh, S. 2004. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49.

Kober, J.; Bagnell, J.; and Peters, J. 2013. Reinforcement Learning in Robotics: A Survey. *The International Journal of Robotics Research*, 32.

Li, X.; Vasile, C.; and Belta, C. 2017. Reinforcement learning with temporal logic rewards. *International Conference on Intelligent Robots and Systems*.

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*.

Manna, Z.; and Pnueli, A. 1987. A Hierarchy of Temporal Properties. In *Symposium on Principles of Distributed Computing*.

Pnueli, A. 1977. The Temporal Logic of Programs. In *Symposium on Foundations of Computer Science*.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Sadigh, D.; Kim, E. S.; Coogan, S.; Sastry, S. S.; and Seshia, S. A. 2014. A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications. In *Conference on Decision and Control*.

Schwarting, W.; Alonso-Mora, J.; and Rus, D. 2018. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1.

Strehl, A.; Li, L.; Wiewiora, E.; Langford, J.; and Littman, M. 2006. PAC Model-Free Reinforcement Learning. In *International Conference on Machine Learning*.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Temizer, S.; Kochenderfer, M.; Kaelbling, L.; Lozano-Perez, T.; and Kuchar, J. 2010. Collision Avoidance for Unmanned Aircraft using Markov Decision Processes. In *AIAA Guidance, Navigation, and Control Conference*.

Valiant, L. G. 1984. A Theory of the Learnable. *Communications of the ACM*, 27(11).