Reinforcement Learning for General LTL Objectives Is Intractable

Cambridge Yang[1], **Michael Littman**[2], **Michael Carbin**[1]

[1] MIT CSAIL
[2] Brown University
camyang@csail.mit.edu, mlittman@cs.brown.edu, mcarbin@csail.mit.edu

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.

1.1 The Linear 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 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).

1.2 A Trouble with Infinite Horizons

The general class of LTL objectives includes *infinite-horizon objectives*: an objective that requires inspecting infinitely many steps of a trajectory to determine if the trajectory satisfies the objective. For example, consider the LTL formula

$\mathsf{F}\ goal$ (eventually reach the goal). Given an infinite trajectory, the objective requires inspecting all steps in the trajectory in the worst case to determine that the trajectory does not satisfy the formula.

Despite the above developments on reinforcement learning with LTL objectives, these objectives present challenges that have been alluded to – but not formally elaborated – in prior work. Henriques et al. (2012); Ashok, Křetínský, and Weininger (2019); Jiang et al. (2020) noted slow learning times for mastering infinite-horizon properties. Littman et al. (2017) provided a specific environment that illustrates the hardness of learning for a specific infinite-horizon objective, arguing for the use of a discounted variant of LTL.

A similar trouble exists for the infinite-horizon, average-reward objectives. In particular, it is understood that reinforcement-learning algorithms do not have guarantees on the learned behavior for infinite-horizon, average-reward problems without additional assumptions on the environment (Kearns and Singh 2002).

However, to our knowledge, no prior work has formally stated and proved the learnability of LTL objectives[1].

Our Results. This work proves that reinforcement-learning agents cannot identify a near-optimal behavior for infinite-horizon LTL objectives with confidence. The intuition for this hardness is: Any finite number of interactions with an environment with unknown transition dynamics is insufficient to identify the environment dynamics perfectly. For an infinite-horizon objective, a behavior's satisfaction probability under the inaccurate environment dynamics can be arbitrarily different from the behavior's satisfaction probability under the true dynamics. Consequently, a learner cannot guarantee with any confidence that it has identified near-optimal behavior for an infinite-horizon objective.

1.3 Implication for Relevant and Future Work

By inspecting our core result, we identify several categories of approaches that work around the unlearnability problem of general LTL objectives. Moreover, we can interpret various previous approaches as instantiations of the identified categories. We summarize the categories and the approaches under each category below.

- Work with finite-horizon LTL objectives, the complement of infinite-horizon objectives, to obtain guarantees on the learned behavior (Henriques et al. 2012). These objectives are decidable within a known finite number of steps. E.g., the objective $a \wedge \mathsf{X} a$ (a is true for two steps) is finite-horizon.
- Seek a best-effort confidence interval (Ashok, Křetínský, and Weininger 2019). Specifically, the interval can be trivial in the worst case, denoting that learned behavior is a maximally poor approximation of the optimal behavior.
- Make additional assumptions about the environment to obtain guarantees on the learned behavior (Fu and Topcu 2014; Brázdil et al. 2014).
- Change the problem by working with LTL-like objectives such as: 1. relaxed LTL objectives that become exactly LTL in the (unreachable) limit (Sadigh et al. 2014; Hahn et al. 2019; Hasanbeig et al. 2019; Bozkurt et al. 2020) and 2. objectives that use temporal operators but employs a different semantics (Littman et al. 2017; Li, Vasile, and Belta 2017; Giacomo et al. 2019; Camacho et al. 2019). The learnability of these objectives is a potential future research direction.

1.4 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 2002; 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 LTL objectives.
- A statement and proof that: 1. Any infinite-horizon LTL formula is not LTL-PAC-learnable. 2. Any finite-horizon LTL formula is LTL-PAC-learnable. To that end, for any infinite-horizon 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.

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

[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 J.

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:

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

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 2002; 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:

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

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 the definition allows 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 $\neg, \wedge, \vee$,

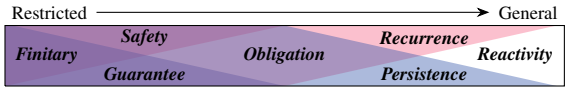

Figure 1: The hierarchy of LTL

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:

$$\phi := a \,|\, \neg\phi \,|\, \phi \wedge \phi \,|\, \phi \vee \phi \,|\, \mathsf{X}\phi \,|\, \mathsf{G}\,\phi \,|\, \mathsf{F}\,\phi \,|\, \phi\,\mathsf{U}\,\phi,\ a \in \Pi. \quad (2)$$

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 over atomic propositions Π. The labeling function is a classifier mapping each MDP state to a tuple of truth values of the 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 $\mathit{Finitary}$ class being the most restricted and the $\mathit{Reactivity}$ 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 $\mathit{Finitary}$.
- $\phi \in \mathit{Guarantee}$ 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 $\mathit{Guarantee}$ asserts that something eventually happens. E.g., $\mathsf{F}\,a$ (i.e., eventually a is true) is in $\mathit{Guarantee}$.
- $\phi \in \mathit{Safety}$ 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.

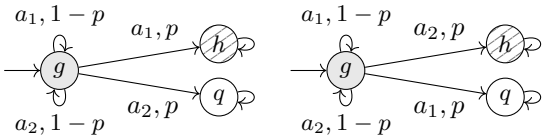

Figure 2: Two MDPs parameterized by p in range $0 < p < 1$. Action a_1 in the MDP on the left and action a_2 in the MDP on the right have probability p of transitioning to the state h. Conversely, action a_2 in the MDP on the left and action a_1 in the MDP on the right have probability p of transitioning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

Moreover, $\mathit{Finitary}$ is the intersection of $\mathit{Guarantee}$ 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 $g \to h$ 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 coun-

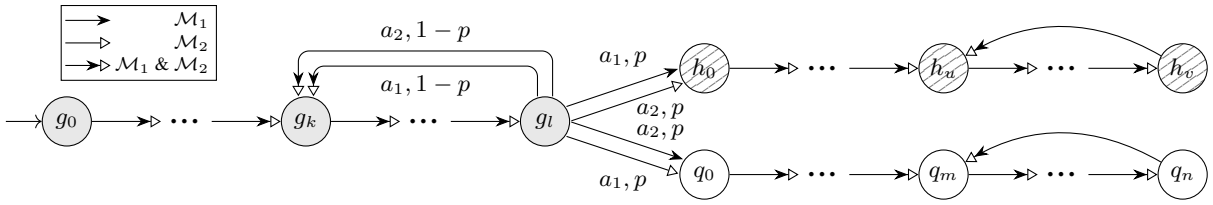

Figure 3: Counterexample MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, with transitions distinguished by arrow types (see legend). Both MDPs are parameterized by the parameter p that is in range $0 < p < 1$. Unlabeled edges are deterministic (actions a_1 and a_2 transition with probability 1). Ellipsis indicates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

terexample 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

This section states and give a proof outline to the main result.

4.1 Theorem Statement

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,

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

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

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 $finitary$.*

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.

4.2 Proof Outline

We give a proof outline to the forward direction of Theorem 1 below and give a detailed proof in Appendix B. We reserve the proof of the reverse direction for Appendix F.

• **MDP Family.** We construct a family of pairs MDPs, as shown in Figure 3. The key design behind each pair in the family is similar to Figure 2. In particular, the design is, as we will show, that no planning-with-generative-model algorithm can learn a policy that is simultaneously ϵ-optimal on both MDPs. In particular, we show that, for an algorithm to learn such a policy, the algorithm must observe at least one transition from the state g_l to the state h_0 or state q_0. Moreover, the probability of an algorithm observing such transition at least once depends on the transition probability from the state g_l to the state h_0 or state q_0.

• **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 planning-with-generative-model algorithm applied to any pair of MDPs in our constructed family in Figure 3. This lower bound depends on a specific transition probability in the constructed MDPs. In particular, we show that learning the ϵ-optimal policy for $\mathsf{F}\, h_0$ necessarily requires observing a transition from the state g_l to the state h_0 or state q_0 at least once. Further, the number of samples needed to observe such a transition (with high probability) has a lower bound of $\frac{\log(2\delta)}{\log(1-p)}$. Therefore, we show that the sample complexity for any planning-with-generative-model algorithm for our constructed MDPs has a lower bound of $\frac{\log(2\delta)}{\log(1-p)}$. Importantly, this bound depends on the probability of a transition of the constructed MDP.

• **Sample complexity of Non-finitary formulas.** We reduce the problem of learning the optimal satisfying pol-

icy for $\mathsf{F}\, h_0$ to the problem of learning the optimal satisfying policy for any non-finitary LTL formula (those not in $\mathit{Finitary}$). The key observation is that, given any non-finitary formula, from the structure of the formula, we can identify a pair of MDPs in our family. For both MDPs in this pair, finding an ϵ-optimal policy for $\mathsf{F}\, h_0$ is reducible to finding an ϵ-optimal policy for the given formula. By this reduction, the established sample complexity lower bound for the case of $\mathsf{F}\, h_0$ also applies to the case of any non-finitary formula. Therefore, the sample complexity of learning an ϵ-optimal policy for any non-finitary formula has a lower bound of $\frac{\log(2\delta)}{\log(1-p)}$. Importantly, this bound depends on the probability of a transition of the constructed MDP.

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 the 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.

4.3 Empirical Justifications

We empirically demonstrate our main result in Appendix G.

4.4 Consequence of the Core Theorem

The implication of the forward direction of Theorem 1 is: For objective specified by a non-finitary LTL formula, 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 vehicles (Temizer et al. 2010; Kober, Bagnell, and Peters 2013; Schwarting, Alonso-Mora, and Rus 2018).

5 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.

5.1 Use a Finitary Objective

By Theorem 1, one can obtain an LTL-PAC guarantee for finitary LTL formulas. Finitary properties 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.

5.2 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 $\mathit{Guarantee}$ 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.

5.3 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.

5.4 Use an LTL-like Objective

Theorem 1 only considers LTL objectives. However, one opportunity for obtaining a 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.

LTL-in-the-limit Objectives 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.

General LTL-like Objectives Prior approaches (Littman et al. 2017; Li, Vasile, and Belta 2017; Giacomo et al. 2019; Camacho et al. 2019) also use general LTL-like specifications that do not or are not known to converge to LTL in a limit. For example, Camacho et al. (2019) introduced the reward machine objective. A reward machine is a finite state automaton that specifies a reward function. As another example, Littman et al. (2017) introduced *geometric LTL*. Geometric LTL attaches a geometrically distributed horizon to each temporal operator. The learnability of these general LTL-like objectives is a potential future research direction.

6 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, Hahn et al. (2019); 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 *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*.

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 *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 *Neural Information Processing Systems*.

Kearns, M.; and Singh, S. 2002. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49(2).

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).