Appendices

A Background on Linear Temporal Logic

A.1 LTL Semantics

In this section, we give the formal semantics of LTL. Recall that the satisfaction relation $w \vDash \phi$ denotes that the infinite word w satisfies ϕ. Equation (A.1) defines this relation.

$$\begin{aligned}
&w \vDash a && \text{iff } a \in w[0] \quad a \in \Pi \\
&w \vDash \neg\phi && \text{iff } w \nvDash \phi \\
&w \vDash \phi \wedge \psi && \text{iff } w \vDash \phi \wedge w \vDash \psi \\
&w \vDash \mathsf{X}\phi && \text{iff } w[1{:}] \vDash \phi \\
&w \vDash \phi \,\mathsf{U}\, \psi && \text{iff } \exists j \geq 0.\, \big(w[j{:}] \vDash \psi \wedge \\
& && \quad \forall k \geq 0.\, k < j \Rightarrow w[k{:}] \vDash \phi\big).
\end{aligned} \tag{A.1}$$

The rest of the operators are defined as syntactic sugar in terms of operators in Equation (A.1) as: $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$, $\mathsf{F}\,\phi \equiv \text{True}\,\mathsf{U}\,\phi$, $\mathsf{G}\,\phi \equiv \neg\mathsf{F}\,\neg\phi$.

A.2 Complete Description of the LTL Hierarchy

In this section, we describe the key properties of all classes in the LTL hierarchy (see Figure 1).

- $\phi \in \mathit{Finitary}$ 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.
- $\phi \in \mathit{Obligation}$ iff ϕ is a logical combination of formulas in $\mathit{Guarantee}$ and Safety. E.g., $\mathsf{F}\,a \wedge \mathsf{G}\,b$ is in $\mathit{Obligation}$.
- $\phi \in \mathit{Persistence}$ iff there exists a language of finite words L such that $w \vDash \phi$ if L accepts all but finitely many prefixes of w. Informally, a formula in $\mathit{Persistence}$ asserts that something happens finitely often. E.g., $\mathsf{F}\,\mathsf{G}\,a$ (i.e., a is not true for only finitely many times, and eventually a stays true forever) is in $\mathit{Persistence}$.
- $\phi \in \mathit{Recurrence}$ iff there exists a language of finite words L such that $w \vDash \phi$ if L accepts infinitely many prefixes of w. Informally, a formula in $\mathit{Recurrence}$ asserts that something happens infinitely often. E.g., $\mathsf{G}\,\mathsf{F}\,a$ (i.e., a is true for infinitely many times) is in $\mathit{Recurrence}$.
- $\phi \in \mathit{Reactivity}$ iff ϕ is a logical combination of formulas in $\mathit{Recurrence}$ and $\mathit{Persistence}$. E.g., $\mathsf{G}\,\mathsf{F}\,a \wedge \mathsf{F}\,\mathsf{G}\,b$ is in $\mathit{Reactivity}$.

B Proof of Theorem 1: the Forward Direction

B.1 MDP Family

We first give two constructions of parameterized counterexample MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ shown in Figure 3 (repeated in Figure B.1 for convenience). 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 $q_{m\ldots n}$.

B.2 Sample Complexity of $\mathsf{F}\,h_0$

We next 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 (i.e., $V^{\pi}_{\mathcal{M},\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M},\xi^{h_0}} - \epsilon$) 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 B.1. *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* $\zeta_i = \mathsf{P}_{\mathsf{T}}\Big(V^{\mathcal{A}^L(T)}_{\mathcal{M}_i,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_i,\xi^{h_0}} - \epsilon \;\Big|\; n(T) = 0\Big)$ *and $n(T)$ 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 B.1 in Appendix C. □

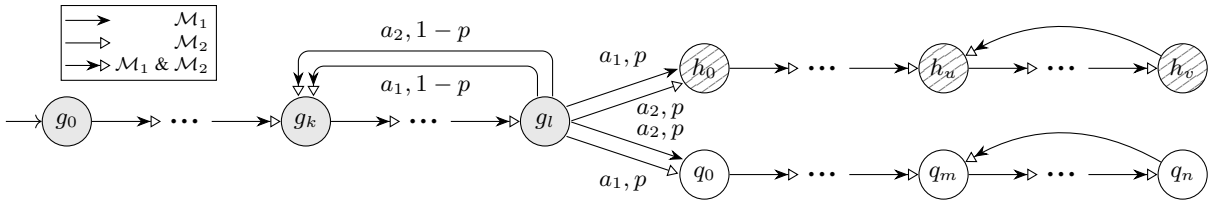


Figure B.1: Counterexample MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, with transitions distinguished by arrow types (see legend). Both MDPs are parameterized by the parameter p that is in range $0 < p < 1$. Unlabeled edges are deterministic (actions a_1 and a_2 transition with probability 1). Ellipsis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

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 B.2. *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 B.2; we give the complete proof in Appendix D.

Proof Sketch of Lemma B.2. 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 B.1 to get the desired lower bound of $N \geq \frac{\log(2\delta)}{\log(1-p)}$. □

B.3 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 1. 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 B.3. *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 1. 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 (B.1).

• 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 (B.2).

$$\mathcal{L}(s) = \begin{cases} [w_a; w_b][j] & \text{if } s{=}g_j \\ w_b[j] & \text{if } s{=}h_j \\ [w_c; w_d][j] & \text{if } s{=}q_j \end{cases} \quad \text{(B.1)} \qquad \mathcal{L}(s) = \begin{cases} [w_a; w_b][j] & \text{if } s{=}g_j \\ [w_c; w_d][j] & \text{if } s{=}h_j \\ w_b[j] & \text{if } s{=}q_j \end{cases} \quad \text{(B.2)}$$

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 B.4. *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 B.3 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 B.2 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 $\phi \notin$ *Finitary* has an uncommittable word. In Appendix E, 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. □

B.4 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.

C Proof of Lemma B.1

To the end of proving Lemma B.1, we first observe the following proposition:

Proposition C.1. *For any non-stationary policy π, the satisfaction probabilities for $\mathcal{M}_1$ and $\mathcal{M}_2$ sum to one:*

$$V^{\pi}_{\mathcal{M}_1,\xi^{h_0}} + V^{\pi}_{\mathcal{M}_2,\xi^{h_0}} = 1.$$

We give a proof of Proposition C.1 in Appendix C.1.

Proof of Lemma B.1. Note that the optimal satisfaction probabilities in both $\mathcal{M}_1$ and $\mathcal{M}_2$ is one, that is, $V^{\pi^*}_{\mathcal{M}_i,\xi^{h_0}} = 1$. This is because the policy that always chooses a_i in $\mathcal{M}_i$ guarantees visitation to the state h_0. Therefore, a corollary of Proposition C.1 is that for any policy π and any $\epsilon < \frac{1}{2}$, the policy π can only be ϵ-optimal in one of $\mathcal{M}_1$ and $\mathcal{M}_2$. Specifically, we have:

$$\mathbb{1}_{\left(V^{\pi}_{\mathcal{M}_1,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_1,\xi^{h_0}} - \epsilon\right)} + \mathbb{1}_{\left(V^{\pi}_{\mathcal{M}_2,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_2,\xi^{h_0}} - \epsilon\right)} \leq 1. \quad \text{(C.1)}$$

Consider a specific sequence of transitions T of length N sampled from either $\mathcal{M}_1$ or $\mathcal{M}_2$. If $n(T) = 0$, we must have the probability of observing T in $\mathcal{M}_1$ equals to the probability of observing T in $\mathcal{M}_2$, that is:

$$\mathsf{P}_{\mathsf{T}\sim\langle \mathcal{M}_1,\mathcal{A}^{\mathrm{S}}\rangle_N}(\,\mathsf{T} = T \mid n(T) = 0\,) = \mathsf{P}_{\mathsf{T}\sim\langle \mathcal{M}_2,\mathcal{A}^{\mathrm{S}}\rangle_N}(\,\mathsf{T} = T \mid n(T) = 0\,).$$

This is because the only differences between $\mathcal{M}_1$ and $\mathcal{M}_2$ are the transitions $g_l \to h_0$ and $g_l \to q_0$, and conditioning on $n(T) = 0$ effectively eliminates these differences.

Therefore, we can write the sum of ζ_1 and ζ_2 as:

$$\zeta_1 + \zeta_2 = \sum_{\forall T} \mathsf{P}_{\mathsf{T}}(\,\mathsf{T} = T \mid n(T) = 0\,) \cdot \left\{ \mathbb{1}_{\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_1,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_1,\xi^{h_0}} - \epsilon\right)} + \mathbb{1}_{\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_2,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_2,\xi^{h_0}} - \epsilon\right)} \right\}.$$

Plugging in Equation (C.1), we get

$$\zeta_1 + \zeta_2 \leq 1.$$

This then implies that $\min(\zeta_1, \zeta_2) \leq \frac{1}{2}$. □

C.1 Proof of Proposition C.1

Proof. We first focus on $\mathcal{M}_1$. Consider an infinite run $\tau = (s_0, a_0, s_1, a_1, \dots)$ of the policy π on $\mathcal{M}_1$. Let $\tau[:i]$ denote the partial history up to state s_i; let w denote all the states $(s_0, s_1, \dots)$ in τ. Let E_i denote the event that the visited state at step i is either h_0 or q_0: $w[i] \in \{h_0, q_0\}$. We have:

$$\begin{aligned} V^{\pi}_{\mathcal{M}_1,\xi^{h_0}} &= \mathsf{P}(\mathcal{L}(w) \vDash \mathsf{F}\, h_0) \\ &= \sum_{i=1}^{\infty} \mathsf{P}(\,\mathcal{L}(w) \vDash \mathsf{F}\, h_0 \mid E_i\,) \cdot \mathsf{P}(E_i) \end{aligned}$$

Given that E_i happens, the previous state $w[i-1]$ must be g_l. Then, the probability of satisfying the formula given the event E_i is the probability of the learned policy choosing a_1 from the state g_l after observing the partial history $\tau[i-1]$:

$$V^{\pi}_{\mathcal{M}_1,\xi^{h_0}} = \sum_{i=1}^{\infty} \mathsf{P}(\,\pi(\tau[:i-1]) = a_1 \mid E_i\,) \cdot \mathsf{P}(E_i). \quad \text{(C.2)}$$

Symmetrically for $\mathcal{M}_2$ we then have:

$$V^{\pi}_{\mathcal{M}_2,\xi^{h_0}} = \sum_{i=1}^{\infty} \mathsf{P}(\,\pi(\tau[:i-1]) = a_2 \mid E_i\,) \cdot \mathsf{P}(E_i). \quad \text{(C.3)}$$

For any policy and any given partial history, the probability of choosing a_1 or a_2 must sum to 1, that is:

$$\mathsf{P}(\,\pi(\tau[:i-1]) = a_1 \mid E_i\,) + \mathsf{P}(\,\pi(\tau[:i-1]) = a_2 \mid E_i\,) = 1$$

Therefore, we may add Equation (C.2) and Equation (C.3) to get:

$$V^{\pi}_{\mathcal{M}_1,\xi^{h_0}} + V^{\pi}_{\mathcal{M}_2,\xi^{h_0}} = \sum_{i=1}^{\infty} 1 \cdot \mathsf{P}(E_i).$$

Finally, since the event E_i must happen for some finitary i with probability 1 (i.e., either h_0 or q_0 must be reached eventually with probability 1), the expression on the right of the equation sums to 1. □

D Complete Proof of Lemma B.2

Proof. First, consider $\mathcal{M}_1$. We will derive an inequality that lower bounds N. We begin by asserting that the inequality of Equation (3) holds true for a reinforcement-learning algorithm $\mathcal{A} = (\mathcal{A}^{\mathrm{S}}, \mathcal{A}^{\mathrm{L}})$. That is:

$$\mathsf{P}_{\mathsf{T}}\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_1,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_1,\xi^{h_0}} - \epsilon\right) \geq 1 - \delta.$$

We expand the left-hand side by conditioning on $n(T) = 0$:

$$\begin{aligned} &\mathsf{P}_{\mathsf{T}}\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_1,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_1,\xi^{h_0}} - \epsilon \;\middle|\; n(T) = 0\right)\mathsf{P}_{\mathsf{T}}(n(T) = 0) + \\ &\mathsf{P}_{\mathsf{T}}\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_1,\xi^{h_0}} \geq V^{\pi^*}_{\mathcal{M}_1,\xi^{h_0}} - \epsilon \;\middle|\; n(T) > 0\right)(1 - \mathsf{P}_{\mathsf{T}}(n(T) = 0)) \\ &\qquad \geq 1 - \delta. \end{aligned}$$

Since $\mathsf{P}_{\mathsf{T}}\left(V^{\mathcal{A}^{\mathrm{L}}(T)}_{\mathcal{M}_1,\xi^{h_0}} \geq 1 - \epsilon \;\middle|\; n(T) > 0\right) \leq 1$, we may relax the inequality to:

$$(1 - \zeta_1)\mathsf{P}_{\mathsf{T}}(n(T) = 0) \leq \delta,$$

where we also plugged in our definition of ζ_i (see Lemma B.1). This relaxation optimistically assumes that a reinforcement-learning algorithm can learn an ϵ-optimal policy by observing at least one transition to h_0 or q_0.

Since there are at most N transitions initiating from the state g_l, and $n(T) = 0$ only occurs when all those transitions end up in g_k, we have $\mathsf{P}_{\mathsf{T}}(n\,(T) = 0) \geq (1-p)^N$. Incorporating this into the inequality we have:

$$(1-\zeta_1)\,(1-p)^N \leq \delta.$$

Symmetrically, for $\mathcal{M}_2$ we have:

$$(1-\zeta_2)\,(1-p)^N \leq \delta.$$

Since both inequalities need to be satisfied, we may combine them by using $\min$ to always choose the tighter inequality:

$$(1-\min\,(\zeta_1,\zeta_2))\,(1-p)^N \leq \delta.$$

By applying Lemma B.1, we finally remove the inequality's dependence on ζ_i, and get the desired lower bound of

$$N \geq \frac{\log(2\delta)}{\log\,(1-p)},$$

which completes the proof of Lemma B.2. □

E Uncommittable Words for non-$Finitary$ Formulas

In this section, we prove the following lemma:

Lemma E.1. *Any LTL formula $\phi \notin Guarantee$ has an uncommittable accepting word. Any LTL formula $\phi \notin Safety$ has an uncommittable rejecting word.*

E.1 Preliminaries

We will review some preliminaries to prepare for our proof of Lemma E.1.

We will use an automaton-based argument for our proof of Lemma E.1. To that end, we recall the following definitions for automatons.

Deterministic Finite Automaton A deterministic finite automaton (DFA) is a tuple $(S, A, P, s_0, s_{\text{acc}})$, where (S, A, P, s_0) is a deterministic MDP (i.e., P degenerated to a deterministic function $(S \times A) \to S$), and $s_{\text{acc}} \in S$ is an accepting state.

Deterministic Rabin Automaton A deterministic Rabin automaton (DRA) is a tuple (S, Π, T, s_0, Acc), where

- S is a finite set of states.
- Π is the atomic propositions of ϕ.
- T is a transition function $(S \times 2^{\Pi}) \to S$.
- $s_0 \in S$ is an initial state.
- Acc is a set of pairs of subsets of states $(B_i, G_i) \in (2^S)^2$.

An infinite length word w over the atomic propositions Π is accepted by the DRA, if there exists a run of the DRA such that there exists a $(B_i, G_i) \in Acc$ where the run visits all states in B_i finitely many times and visits some state(s) in G_i infinitely many times.

For any LTL formula ϕ, one can always construct an equivalent DRA that accepts the same set of infinite length words as ϕ (Safra 1988).

E.2 Proof of Lemma E.1 for $\phi \notin Guarantee$

Given an LTL formula ϕ, we first construct its equivalent DRA $\mathcal{R} = (S, \Pi, T, s_0, Acc)$ (Safra 1988).

A *path* in a DRA is a sequence of transitions in the DRA. A *cycle* in a DRA is a path that starts from some state and then returns to that state. A cycle is *accepting* if there exists a pair $(B_i, G_i) \in Acc$, such that the cycle does not visit states in B_i and visits some states in G_i. Conversely, a cycle is *rejecting* if it is not accepting. With the above definitions and to the end of proving Lemma E.1 for the case of $\phi \notin Guarantee$, we state and prove the following lemma.

Lemma E.2. *For any LTL formula $\phi \notin Guarantee$ and its equivalent DRA $\mathcal{R}$, it must be the case that $\mathcal{R}$ contains an accepting cycle that is reachable from the initial state and there exists a path from a state in the accepting cycle to a rejecting cycle.*

Proof. Suppose, for the sake of contradiction, there does not exist an accepting cycle that 1. is reachable from the initial state and 2. has a path to a rejecting cycle in the equivalent DRA $\mathcal{R}$. Then there are two scenarios:

- $\mathcal{R}$ does not have any accepting cycle that is reachable from the initial state.
- All accepting cycles reachable from the initial state do not have any path to any rejecting cycle.

For the first scenario, $\mathcal{R}$ must not accept any infinite length word. Therefore ϕ must be equal to F (i.e., the constant falsum). However, F is in the $Finitary$ LTL class, which is a subset of $Guarantee$, so this is a contradiction.

For the second scenario, consider any infinite length word w. Consider the induced infinite path $\mathcal{P} = (s_0, w[0], s_1, w[1], \dots)$ by w on the DRA starting from the initial state s_0.

If ϕ accepts the word w, the path $\mathcal{P}$ must reach some state in some accepting cycle.

Conversely, if ϕ rejects the word w, the path must not visit any state in any accepting cycle. This is because otherwise the path can no longer visit a rejecting cycle once it visits the accepting cycle, thereby causing the word to be accepted.

Therefore, ϕ accepts the word w as soon as the path $\mathcal{P}$ visits some state in some accepting cycle. This degenerates the DRA to a DFA, where the accepting states are all the states in the accepting cycles of the DRA. Then, an infinite length word w is accepted by ϕ if and only if there exists a prefix of w that is accepted by the DFA.

By the property of the $Guarantee$ class (see Appendix A), for $\phi \in Guarantee$, there exists a language of finite length words, L, such that $w \vDash \phi$ if L accepts a prefix of w (Manna and Pnueli 1987). Since a DFA recognizes a regular language, the formula must be in the $Guarantee$ LTL class. This is also a contradiction.

Therefore, there must exist an accepting cycle that is reachable from the initial state and has a path to a rejecting cycle in the equivalent DRA. □

We are now ready to give a construction of w_a, w_b, w_c and w_d that directly proves Lemma E.1 for $\phi \notin Guarantee$. Consider the equivalent DRA $\mathcal{R}$ of the LTL formula. By

Lemma E.2, $\mathcal{R}$ must contain an accepting cycle that is reachable from the initial state and has a path to a rejecting cycle. We can thus define the following paths and cycles:

- Let $\mathcal{P}_a$ be the path from the initial state to the accepting cycle.
- Let $\mathcal{P}_b$ be the accepting cycle.
- Let $\mathcal{P}_c$ be a path from the last state in the accepting cycle to the rejecting cycle.
- Let $\mathcal{P}_d$ be the rejecting cycle.

For a path $\mathcal{P} = (s_i, w[i], \dots s_j, w[j], s_{j+1})$, let $w(\mathcal{P})$ denote the finite length word consisting only of the characters in between every other state (i.e., each character is a tuple of truth values of the atomic propositions): $w(\mathcal{P}) = w[i] \dots w[j]$. Consider the assignments of $w_a = w(\mathcal{P}_a)$, $w_b = w(\mathcal{P}_b)$, $w_c = w(\mathcal{P}_c)$ and $w_d = w(\mathcal{P}_d)$. Notice that:

- The formula ϕ accepts the infinite length word $[w_a; w_b^\infty]$ because P^b is an accepting cycle.
- The formula ϕ rejects all infinite length words $[w_a; w_b^i; w_c; w_d^\infty]$ for all $i \in \mathbb{N}$ because P^d is a rejecting cycle.

By Definition 1, the infinite length word $[w_a; w_b^\infty]$ is an uncommittable accepting word. This construction proves Lemma E.1 for $\phi \notin Guarantee$.

E.3 Proof of Lemma E.1 for $\phi \notin Safety$

The proof for $\phi \notin Safety$ is symmetrical to $\phi \notin Guarantee$. For completeness, we give the proof below.

Given an LTL formula ϕ, we again first construct its equivalent DRA $\mathcal{R} = (S, \Pi, T, s_0, Acc)$.

To the end of proving Lemma E.1 for the case of $\phi \notin Safety$, we state and prove the following lemma.

Lemma E.3. *For any LTL formula $\phi \notin Safety$ and its equivalent DRA $\mathcal{R}$, it must be the case that $\mathcal{R}$ contains a rejecting cycle that is reachable from the initial state and has a path from any state in the rejecting cycle to an accepting cycle.*

Proof. Suppose, for the sake of contradiction, there does not exist a rejecting cycle that 1. is reachable from the initial state and 2. has a path to an accepting cycle in the equivalent DRA $\mathcal{R}$. Then there are two scenarios:

- $\mathcal{R}$ does not have any rejecting cycle that is reachable from the initial state.
- All rejecting cycles reachable from the initial state do not have any path to any accepting cycle.

For the first scenario, $\mathcal{R}$ must not reject any infinite length word. Therefore ϕ must be equal to T (i.e., the constant truth). However, T is in the $Finitary$ LTL class, which is a subset of $Safety$, so this is a contradiction.

For the second scenario, consider any infinite length word w. Consider the induced infinite path $\mathcal{P} = (s_0, w[0], s_1, w[1], \dots)$ by w on the DRA starting from the initial state s_0.

If ϕ rejects the word w, the path $\mathcal{P}$ must reach some state in some rejecting cycle.

Conversely, if ϕ accepts w, the path must not visit any state in any rejecting cycle. This is because otherwise the path can no longer visit a accepting cycle once it visits the rejecting cycle, thereby causing the word to be rejected.

Therefore, ϕ rejects the word w as soon as the path $\mathcal{P}$ visits some state in some rejecting cycle. This degenerates the DRA to a DFA, where the accepting states are all the states except those in the rejecting cycles of the DRA. Then, an infinite length word w is accepted by ϕ if and only if there all prefixes of w are accepted by the DFA.

By the property of the $Safety$ class (see Appendix A), for $\phi \in Safety$, there exists a language of finite length words, L, such that $w \vDash \phi$ if L accepts all prefixes of w (Manna and Pnueli 1987). Since a DFA recognizes a regular language, the formula must be in the $Safety$ LTL class. This is also a contradiction.

Therefore, there must exist a rejecting cycle that is reachable from the initial state and has a path to an accepting cycle in the equivalent DRA. □

We are now ready to give a construction of w_a, w_b, w_c and w_d that directly proves Lemma E.1 for $\phi \notin Safety$. Consider the equivalent DRA $\mathcal{R}$ of the LTL formula. By Lemma E.3, $\mathcal{R}$ must contain a rejecting cycle that is reachable from the initial state and has a path to an accepting cycle. We can thus define the following paths and cycles:

- Let $\mathcal{P}_a$ be the path from the initial state to the rejecting cycle.
- Let $\mathcal{P}_b$ be the rejecting cycle.
- Let $\mathcal{P}_c$ be a path from the last state in the rejecting cycle to the accepting cycle.
- Let $\mathcal{P}_d$ be the accepting cycle.

Consider the assignments of $w_a = w(\mathcal{P}_a)$, $w_b = w(\mathcal{P}_b)$, $w_c = w(\mathcal{P}_c)$ and $w_d = w(\mathcal{P}_d)$. Notice that:

- The formula ϕ rejects the infinite length word $[w_a; w_b^\infty]$ because P^b is a rejecting cycle.
- The formula ϕ accepts all infinite length words $[w_a; w_b^i; w_c; w_d^\infty]$ for all $i \in \mathbb{N}$ because P^d is an accepting cycle.

By Definition 1, the infinite length word $[w_a; w_b^\infty]$ is an uncommittable rejecting word. This construction proves Lemma E.1 for $\phi \notin Safety$.

F Proof of Theorem 1: the Reverse Direction

In this section, we give a proof sketch to the reverse direction of Theorem 1.

F.1 Proof Outline

- **Reduction to Infinite-horizon Cumulative Rewards.** First, given an LTL formula in $Finitary$ and an environment MDP, we will construct an *augmented MDP with rewards* similar to Giacomo et al. (2019); Camacho et al. (2019). We reduce the problem of finding the optimal non-stationary policy for satisfying the formula in the original MDP to the problem of finding the optimal stationary policy that maximizes the infinite-horizon (undiscounted) cumulative rewards in this augmented MDP.

- **Reduction to Finite-horizon Cumulative Rewards.** Next, we reduce the infinite-horizon cumulative rewards to a finite-horizon cumulative rewards, using the fact that the formula is finitary.
- **Sample Complexity Upper Bound.** Lastly, Dann and Brunskill (2015) have derived an upper bound on the sample complexity for a reinforcement-learning algorithm for finite-horizon MDPs. We thus specialize this known upper bound to our problem setup of the augmented MDP and conclude that any finitary formula is PAC-learnable.

F.2 Proof

Reduction to Infinite-horizon Cumulative Rewards Given an LTL formula ϕ in $Finitary$ with atomic propositions Π, one can compile ϕ into a DFA $\bar{\mathcal{M}} = (\bar{S}, 2^{\Pi}, \bar{P}, \bar{s_0}, \bar{s_{\text{acc}}})$ that decides the satisfaction of ϕ (Latvala 2003). In particular, for a given sample path w of DTMC induced by a policy and the environment MDP, $\mathcal{L}(w)$ satisfies ϕ if and only if the DFA, upon consuming $\mathcal{L}(w)$, eventually reaches the accept state s_{acc}. Here, the DFA has a size (in the worst case) doubly exponential to the size of the formula: $|\bar{S}| = \mathcal{O}(2\text{exp}(|\phi|))$ (Kupferman and Vardi 1999).

We then use the following product construction to form an augmented MDP with rewards $\hat{\mathcal{M}} = (\hat{S}, \hat{A}, \hat{P}, \hat{s_0}, \hat{R})$. Specifically,

- The states and actions are: $\hat{S} = S \times \bar{S}$ and $\hat{A} = A$.
- The transitions follow the transitions in the environment MDP and the DFA simultaneously, where the action of the DFA come from labeling the current state of the environment MDP: $\hat{P}((s, \bar{s}), a, (s', \bar{s}')) = P(s, a, s')$ and $\bar{s}' = \bar{P}(\bar{s}, \mathcal{L}(s))$.
- The reward function gives a reward of one to any transition from a non-accepting state that reaches s_{acc} in the DFA, and zero otherwise: $R((s, \hat{s}), a, (s', \hat{s}')) = \mathbb{1}_{s \neq s_{\text{acc}} \wedge \hat{s}' = s_{\text{acc}}}$.

By construction, each run of the augmented MDP gives a reward of 1 iff the run satisfies the finitary formula ϕ. The expected (undiscounted) infinite-horizon cumulative rewards thus equals the satisfaction probability of the formula. Therefore, maximizing the infinite-horizon cumulative rewards in the augmented MDP is equivalent to maximizing the satisfaction probability of ϕ in the environment MDP.

Reduction to Finite-horizon Cumulative Rewards By the property of LTL hierarchy (Manna and Pnueli 1987), for any LTL formula ϕ in $Finitary$ and an infinite length word w, one can decide if ϕ accepts w by inspecting a length-H prefix of w. Here, H is a constant that is computable from ϕ. In particular, H equals the longest distance from the start state to a terminal state in the DFA in the above construction[1]. Thus, the infinite-horizon cumulative rewards is further equivalent to the finite-horizon (of length H) cumulative rewards, since the product construction gives no reward after the horizon H. This implies that finding the optimal policy for ϕ is further equivalent to finding the optimal policy that maximizes the cumulative rewards for a finite horizon H in this augmented MDP.

Sample Complexity Upper Bound Lastly, Dann and Brunskill (2015) derived that with at most $\tilde{\mathcal{O}}\left(\frac{|S|^2|A||H^3}{\epsilon^2}\log\frac{1}{\delta}\right)$ number of state-action samples[2], a particular reinforcement-learning algorithm called UCFH is sample efficiently PAC[3]. Incorporating the fact that the augmented MDP has $|\hat{S}| = |S| \cdot \mathcal{O}(2\text{exp}(|\phi|))$ number of states, we obtain a sample complexity upper bound of $\tilde{\mathcal{O}}\left(\frac{|S|^2\, 2\text{exp}(|\phi|)^2|A|H^3}{\epsilon^2}\log\frac{1}{\delta}\right)$ for the overall reinforcement-learning algorithm.

Since for any finitary formula, we have constructed a reinforcement-learning algorithm that is sample efficiently LTL-PAC for all environment MDPs, this concludes our proof that any formula in $Finitary$ is LTL-PAC-learnable.

G 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 work 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 H. 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 its counterexample MDP as in Figure 2. The second is a formula and an MDP adapted from a case study in prior work (Sadigh et al. 2014). In this section, we focus on the first pair and defer the complete methodology and results to Appendix H.

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

[1] Note that since ϕ is finitary, the DFA does not have any cycles except at the terminal states (Duret-Lutz et al. 2016).

[2] The notation $\tilde{\mathcal{O}}(.)$ is the same as $\mathcal{O}(.)$, but ignores any log-terms.

[3] The bound given in Dann and Brunskill (2015) is $\tilde{\mathcal{O}}\left(\frac{|S|^2|A||H^2}{\epsilon^2}\log\frac{1}{\delta}\right)$, and it is a upper bound on the number of episodes. To make it consistent with our lower bound, which is a bound on the number of sampled transitions, we multiply it by an additional H term.

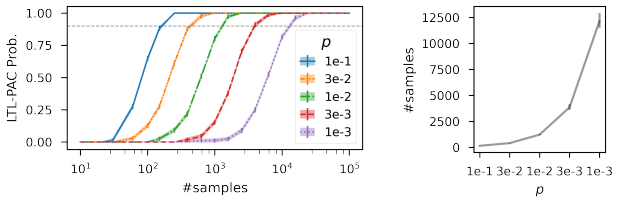


Figure G.1: Left: LTL-PAC probabilities vs. number of samples, varying parameters p. Right: number of samples needed to reach 0.9 LTL-PAC probability vs. parameter p.

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.

Results Figure G.1 presents the results for the algorithm in Bozkurt et al. (2020). Appendix H 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 G.1, 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 G.1, 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 in this case) 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.

H Empirical Experiment Details

H.1 Details of Methodology

Chosen Algorithms We consider a set of state-of-the-art reinforcement-learning algorithms for LTL objectives implemented in the Mongujerrie toolbox (Hahn et al. 2021).

A common pattern in these previous works (Sadigh et al. 2014; Hahn et al. 2019; Bozkurt et al. 2020) is that each work constructs a product MDP with rewards (i.e., an MDP with a reward function on that MDP) from an LTL formula and an environment MDP. Moreover, these works permit the use of any standard reinforcement-learning algorithm, such as Q-learning or SARSA(λ), to solve the constructed product MDP with the specified reward function to obtain the product MDP's optimal policy. Finally, these works cast the optimal policy back to a non-stationary policy of the environment MDP, which becomes the algorithm's output policy.

Following Hahn et al. (2021), we call each specific construction of a product MDP with rewards as a *reward-scheme*. We then characterize each reinforcement-learning algorithm as a "reward-scheme" and "learning-algorithm"

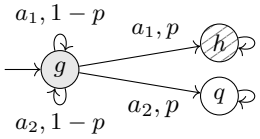


Figure H.1: Environment MDP used in the experiments.

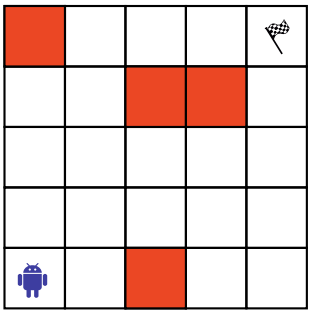


Figure H.2: Gridworld environment MDP from Sadigh et al. (2014) with a customized transition dynamics. The agent starts from the lower left corner. At each time step, the agent can choose to move up, down, left or right. The white cells are sticky: the agent moves towards the intended direction with probability $1-p$ (or stays stationary if it will move off the grid), and stays stationary with probability p. The red cells are trapping: once the agent steps on a red cell, it stays there forever.

pair. We consider a total of five reward-schemes [4]: Reward-on-acc (Sadigh et al. 2014), Multi-discount (Bozkurt et al. 2020), Zeta-reach (Hahn et al. 2019), Zeta-acc (Hahn et al. 2020), and Zeta-discount (Hahn et al. 2020). We consider a total of three learning-algorithms: Q-learning (Watkins and Dayan 1992), Double Q-learning (Hasselt 2010), and SARSA(λ) (Sutton 1988). This yields a total of 15 reinforcement-learning algorithms for LTL objectives.

Algorithm Parameters Each reinforcement-learning algorithm in Mungojerrie accepts a set of hyper-parameters. For the majority of the hyper-parameters, we use their default values as in Mungojerrie Version 1.0 (Hahn et al. 2021). We present the hyper-parameters that differ from the default values in Table H.1. For each of the hyper-parameters in Table H.1, we use a different value from the default value because it allow all the algorithms that we consider to converge within 10^5 steps (i.e., the maximum learning steps that we allow). For SARSA(λ), we use $\lambda = 0$.

Objectives and Environment MDPs We consider two pairs of LTL formulas and environment MDPs. The first pair is the formula $\mathsf{F}\, h$ and the counterexample MDP constructed according to Appendix B.1, shown in Figure H.1. The second pair is the formula $\mathsf{F}\, \mathit{goal}$ and a gridworld environment MDP from from Sadigh et al. (2014) with a customized transition dynamics, shown in Figure H.2.

[4] We use the same naming of each reward-scheme as in the Mungojerrie toolbox (Hahn et al. 2021)

Reinforcement-learning-algorithm	Learning Rate	Exploration	Reset Episode Every Steps
Q-learning	$\frac{10}{10+t}$	Linear decay from 1.0 to 10^{-1}	10
Double Q-learning	$\frac{30}{30+t}$	Linear decay from 1.0 to 10^{-1}	10
SARSA(λ)	$\frac{10}{10+t}$	Linear decay from 1.0 to 10^{-3}	10

Table H.1: Non-default hyper-parameters used for each learning-algorithm

Software and Platform We use a custom version of Mungojerrie. Our modifications are:

- Modification to allow parallel Monte Carlo estimation of the LTL-PAC probability.
- Modification to allow the reinforcement-learning algorithms to have a non-linear learning rate decay. In particular, we use a learning rate of $\frac{k}{k+t}$ at every learning step t, where k is a hyper-parameter (see Table H.1 for the value of k for each algorithm). This modification is necessary for ensuring Q-learning's convergence (Watkins and Dayan 1992).

We run all experiments on a machine with 2.9 GHz 6-Core CPU and 32 GB of RAM.

H.2 Complete Results

The plot we presented in the body of the paper (Figure G.1) corresponds to the setting of Multi-discount with Q-learning, under the environment MDP shown in Figure H.1. We present the complete results for the two pairs of LTL formulas and environment MDPs in Figures H.3 and H.4.

I Classification of Prior Works

In Section 5, we discussed several categories of approaches to work around the hardness of reinforcement learning with LTL objectives and classified approaches in prior works belonging to each category. In this section, we explain the rationale for each classification.

Use a Finitary Objective Henriques et al. (2012) introduced a variant of LTL called *Bounded LTL* and used Bounded LTL objective for reinforcement learning. Every Bounded LTL formula is decidable by a bounded length prefix of the input word. Therefore, each Bounded LTL formula is equivalent to an finitary LTL formula. Therefore, we classified this approach as using a Finitary objective.

Jothimurugan, Alur, and Bastani (2019) introduced a task specification language over finite length words. Further, their definition of an MDP contains an additional finite time horizon H. Each sample path of the MDP is then a length-H finite length word and is evaluated by a formula of the task specification language. [5] Each formula of the task specification language with a fixed finite horizon H is equivalent to an LTL formula in the Finitary class. Therefore, we classified this approach as using a Finitary objective.

Best-Effort Guarantee We classified Ashok, Křetínský, and Weininger (2019) to this category and explained our rationale of this classification in Section 5.

Know More About the Environment We classified Fu and Topcu (2014); Brázdil et al. (2014) to this category and explained our rationale of this classification in Section 5.

Use an LTL-like Objective Littman et al. (2017) introduced a discounted variant of LTL called *Geometric LTL* (GLTL). A temporal operator in a GLTL formula expires within a time window whose length follows a geometric distribution. For example, a GLTL formula $\mathsf{F}_{0.1}$*goal* is satisfied if the sample path reaches the *goal* within a time horizon H, where H follows Geometric(0.1), the geometric distribution with the success parameter 0.1. Since GLTL's semantics is different from LTL's semantics, we classified this approach as using an LTL-like objective.

Li, Vasile, and Belta (2017) introduced a variant of LTL called *Truncated-LTL* (TLTL). A formula in TLTL, similar to a formula in Bounded LTL (Henriques et al. 2012), is decidable by a bounded length prefix of the input word. Moreover, TLTL has a *qualitative semantics*, in addition to the standard Boolean semantics of LTL. In particular, the qualitative semantics of a TLTL formula maps a sample path of the environment MDP to a real number that indicates the degree of satisfaction for the TLTL formula. Therefore, we classified this approach as using an LTL-like objective.

Giacomo et al. (2019) introduced *Restraining Bolts*. A Restraining Bolts specification is a set of pairs (ϕ_i, r_i), where each ϕ_i is an LTLf/LDLf formula, and r_i is a scalar reward. An LTLf formula is similar to an LTL formula but interpreted over finite length words instead of infinite length words. LDLf is an extension of LTLf and is also interpreted over finite length words. [6] Given an environment MDP, the approach checks each finite length prefix of a sample path of the MDP against each ϕ_i, and if a prefix satisfies ϕ_i, the approach gives the corresponding reward r_i to the agent. The objective in Giacomo et al. (2019) is to maxi-

[5] There are two possible interpretations of the finite horizon in Jothimurugan, Alur, and Bastani (2019). The first interpretation is that the environment MDP inherently terminates and produces length-H sample paths. The second interpretation is that the finite horizon H is part of the specification given by a user of their approach. Thus, we use the second interpretation to classify their approach. The difference between the two interpretations is only conceptual — if the environment inherently terminates with a fixed finite horizon H, it would be equivalent to imposing a finite horizon H in the task specification.

[6] LDLf is more expressive than LTLf (De Giacomo and Vardi 2013). In particular, LTLf is equivalent to the star-free subset of regular languages while LDLf is equivalent to the full set of regular languages.

mize the discounted cumulative sum of rewards produced by the Restraining Bolts specification. To the best of our knowledge, this objective is not equivalent to maximizing the satisfaction of an LTL formula. Nonetheless, a Restraining Bolts specification can be seen as an LTL-like specification. Therefore, we classified this approach as using an LTL-like objective.

Camacho et al. (2019) introduced *reward machine*. A reward machine specification is a deterministic finite automaton equipped with a reward for each transition. The objective in Camacho et al. (2019) is to maximize the discounted cumulative rewards produced by the reward machine specification. Camacho et al. (2019) showed that LTL objectives formulas in the Guarantee or Safety class are reducible to reward machine objectives without discount factors. However, since the approach maximizes discounted cumulative rewards in practice, it does not directly optimize for the LTL objectives in the Guarantee or Safety classes. Therefore, we classified this approach as using an LTL-like objective.

We also classified Sadigh et al. (2014); Hahn et al. (2019); Hasanbeig et al. (2019); Bozkurt et al. (2020) as using LTL-like objectives, and explained our rationale of these classifications in Section 5.

J Concurrent Work

Concurrent to this work, Alur et al. (2021) developed a framework to study reductions between reinforcement-learning task specifications. They looked at various task specifications, including cumulative discounted rewards, infinite-horizon average-rewards, reachability, safety, and LTL. They thoroughly review previous work concerning reinforcement learning for LTL objectives, which we also cite. Moreover, Alur et al. (2021, Theorem 8) states a seemingly similar result as the forward direction of our Theorem 1:

> There does not exist a PAC-MDP algorithm for the class of safety specifications.

Despite the parallels, we clarify one crucial difference and two nuances between our work and theirs.

Firstly and most importantly, their theorem is equivalent to "there exists a safety specification that is not PAC-learnable.", whereas our Theorem 1 works pointwise for each LTL formula, asserting "all non-finitary specifications are not PAC-learnable." The proof of their theorem gives one safety specification and shows that it is not PAC-learnable.[7] On the other hand, the proof of the forward direction of our Theorem 1 constructs a counterexample for each non-finitary formula. This point is crucial since it allows us to precisely carve out the PAC-learnable subset, namely the finitary formulas, from the LTL hierarchy.

Secondly, their definition of safety specification is equivalent to a strict subset of the safety class in the LTL hierarchy that we consider. In particular, their safety specification is equivalent to LTL formulas of the form $\mathsf{G}\,(a_1 \vee a_2 \vee \cdots \vee a_n)$, where each $a_i \in \Pi$ is an atomic proposition, with $n = 0$ degenerating the specification to T (the constant true).

[7] Their result is similar to what we showed in our Appendix B.2, where we consider the particular guarantee formula $\mathsf{F}\,h_0$ and show that it is not PAC-learnable.

Lastly, they consider reinforcement-learning algorithms,[8] whereas we consider the slightly more general planning-with-generative-model algorithms. We believe their theorem and proof can be modified to accommodate our more general algorithm definition.

[8] with an additional operation that resets to the initial state

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*.

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*.

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*.

Dann, C.; and Brunskill, E. 2015. Sample Complexity of Episodic Fixed-Horizon Reinforcement Learning. In *Neural Information Processing Systems*.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *International Joint Conference on Artificial Intelligence*.

Duret-Lutz, A.; Lewkowicz, A.; Fauchille, A.; Michaud, T.; Renault, E.; and Xu, L. 2016. Spot 2.0 - A Framework for LTL and ω -Automata Manipulation. In *ATVA*.

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

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*.

Hahn, E.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2020. Faithful and Effective Reward Schemes for Model-Free Reinforcement Learning of Omega-Regular Objectives. In *Automated Technology for Verification and Analysis*.

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*.

Hasselt, H. V. 2010. Double Q-learning. In *Neural Information Processing Systems*.

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*.

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

Kupferman, O.; and Vardi, M. 1999. Model Checking of Safety Properties. *Formal Methods in System Design*, 19.

Latvala, T. 2003. Efficient Model Checking of Safety Properties. In *Model Checking Software*.

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*.

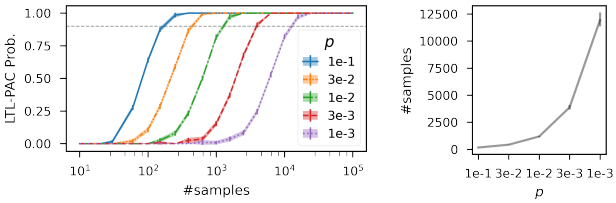
Manna, Z.; and Pnueli, A. 1987. A Hierarchy of Temporal Properties. In *Symposium on Principles of Distributed Computing*.

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*.

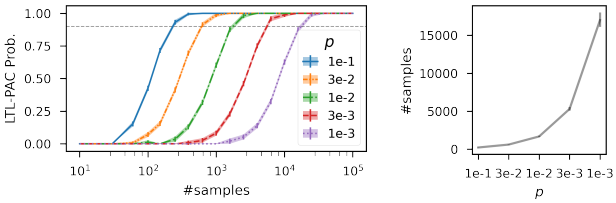
Safra, S. 1988. On the Complexity of ω-Automata. In *Symposium on Foundations of Computer Science*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1).

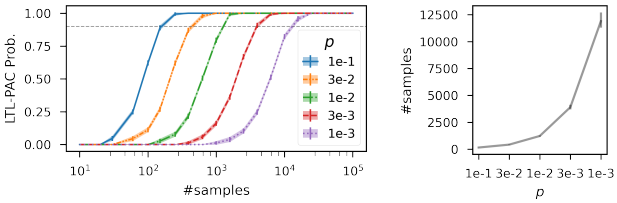
Watkins, C. J. C. H.; and Dayan, P. 1992. Q-learning. *Machine Learning*, 8(3).



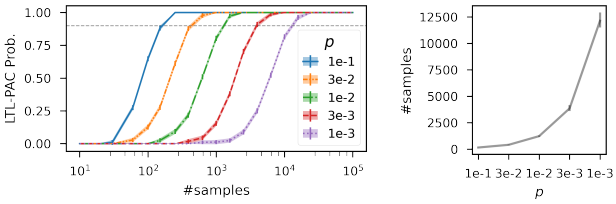
(a) Reward-on-acc with Q-learning



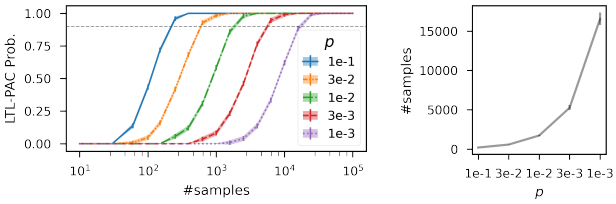
(b) Reward-on-acc with Double Q-learning



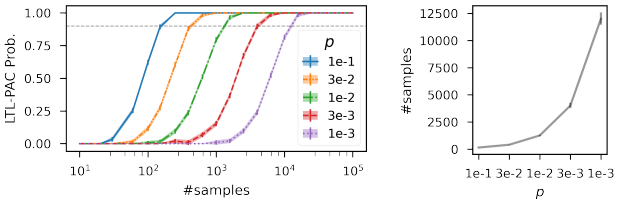
(c) Reward-on-acc with SARSA(λ)



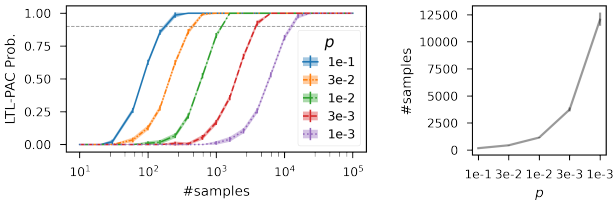
(d) Multi-discount with Q-learning



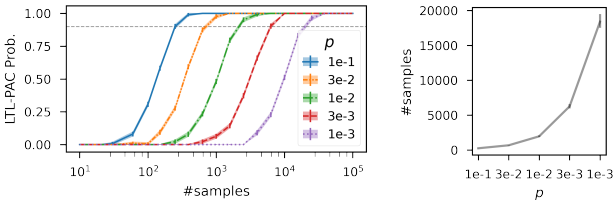
(e) Multi-discount with Double Q-learning



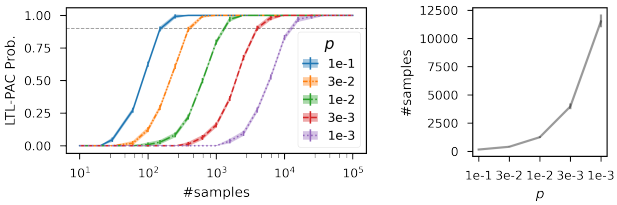
(f) Multi-discount with SARSA(λ)



(g) Zeta-reach with Q-learning

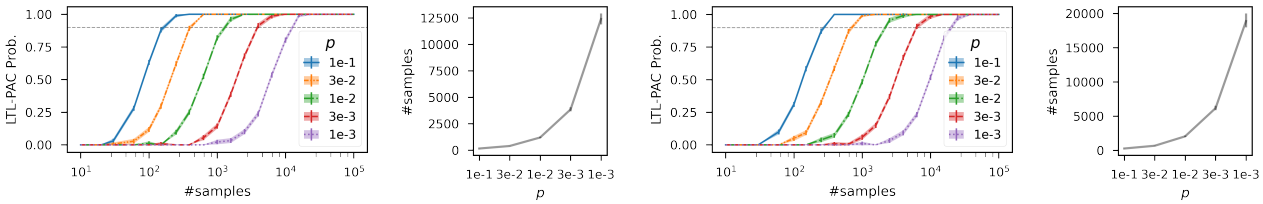


(h) Zeta-reach with Double Q-learning



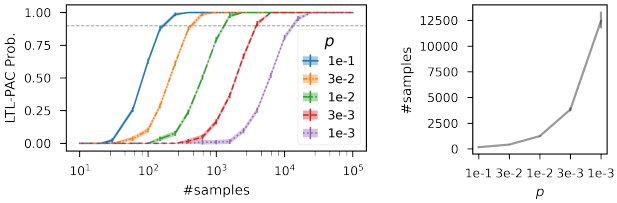
(i) Zeta-reach with SARSA(λ)

Figure H.3: Empirical Results (continued on next page)

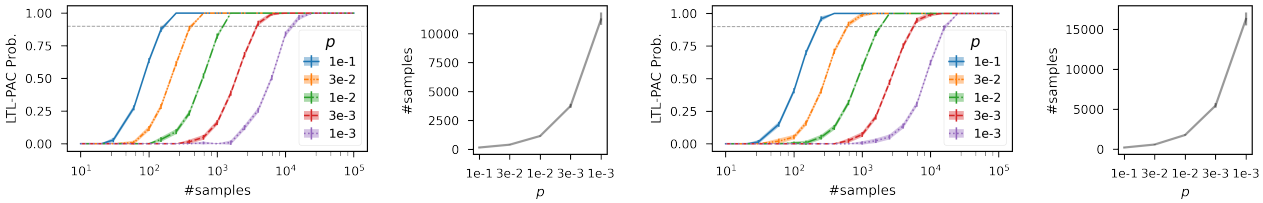


(j) Zeta-acc with Q-learning

(k) Zeta-acc with Double Q-learning

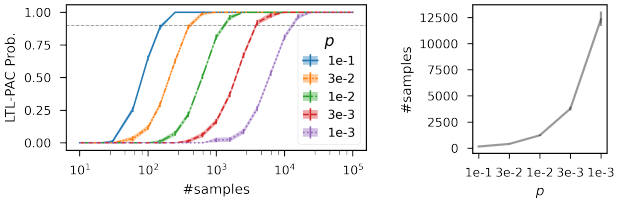


(l) Zeta-acc with SARSA(λ)



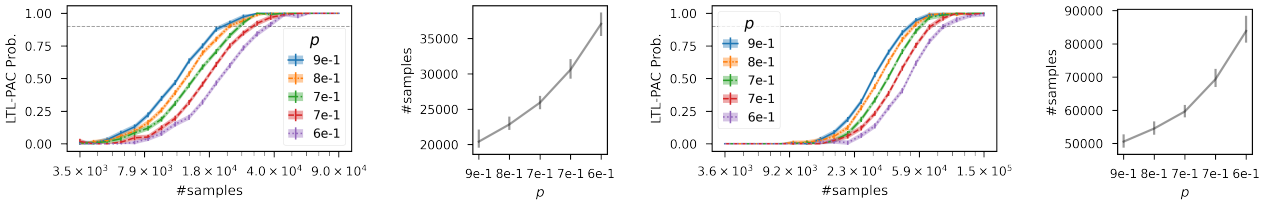
(m) Zeta-discount with Q-learning

(n) Zeta-discount with Double Q-learning



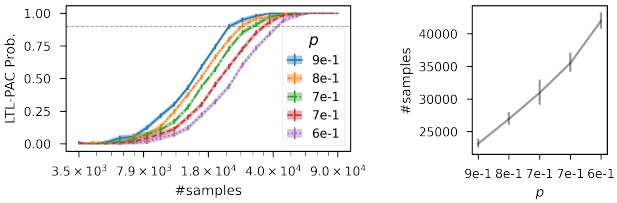
(o) Zeta-discount with SARSA(λ)

Figure H.3: Empirical Results (continued). Each sub-figure corresponds to a specific reward-scheme and learning-algorithm pair. For each sub-figure, on the left: LTL-PAC probabilities vs. number of samples, for varying parameters p; on the right: number of samples needed to reach 0.9 LTL-PAC probability vs. parameters p.

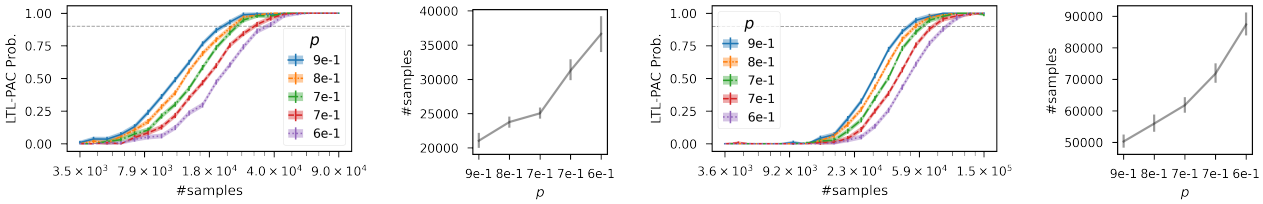


(a) Reward-on-acc with Q-learning

(b) Reward-on-acc with Double Q-learning

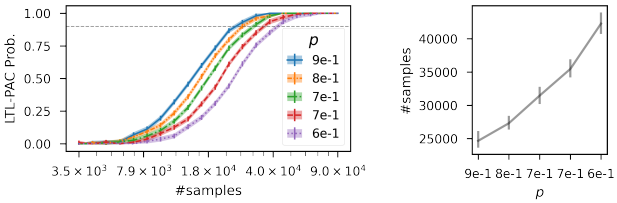


(c) Reward-on-acc with SARSA(λ)

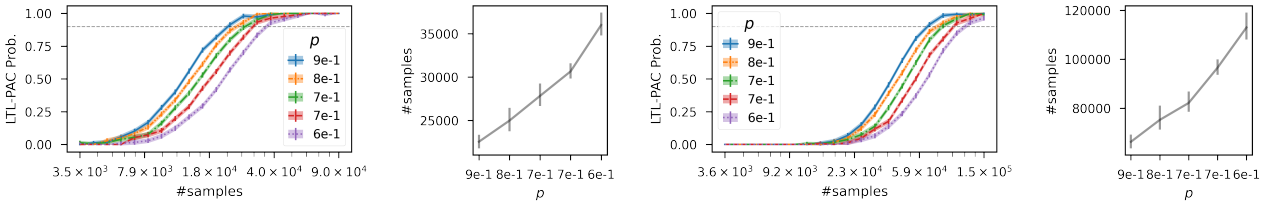


(d) Multi-discount with Q-learning

(e) Multi-discount with Double Q-learning

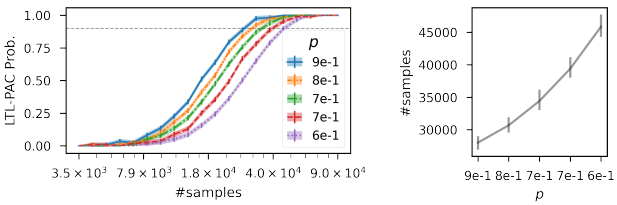


(f) Multi-discount with SARSA(λ)



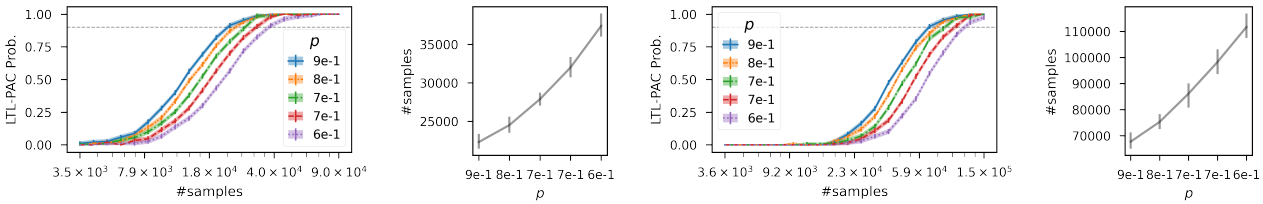
(g) Zeta-reach with Q-learning

(h) Zeta-reach with Double Q-learning



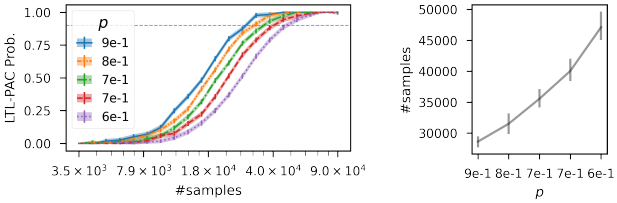
(i) Zeta-reach with SARSA(λ)

Figure H.4: Empirical Results (continued on next page)

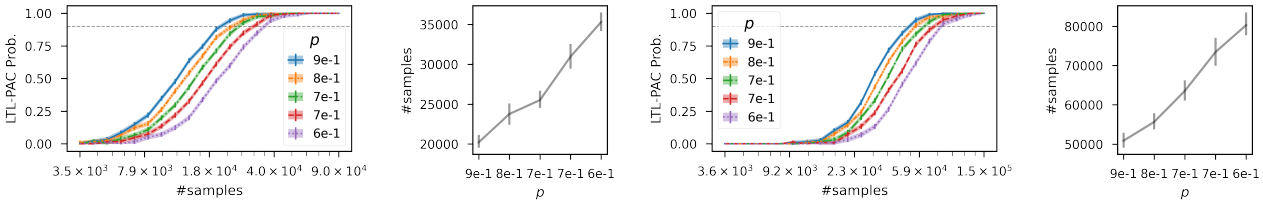


(j) Zeta-acc with Q-learning

(k) Zeta-acc with Double Q-learning

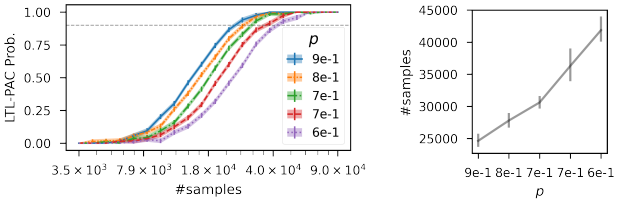


(l) Zeta-acc with SARSA(λ)



(m) Zeta-discount with Q-learning

(n) Zeta-discount with Double Q-learning



(o) Zeta-discount with SARSA(λ)

Figure H.4: Empirical Results (continued). Each sub-figure corresponds to a specific reward-scheme and learning-algorithm pair. For each sub-figure, on the left: LTL-PAC probabilities vs. number of samples, for varying parameters p; on the right: number of samples needed to reach 0.9 LTL-PAC probability vs. parameters p.