Sample Efficient Model-free Reinforcement Learning from LTL Specifications with Optimality Guarantees

Daqian Shao and **Marta Kwiatkowska**

Department of Computer Science, University of Oxford, UK

{daqian.shao, marta.kwiatkowska}@cs.ox.ac.uk

Abstract

Linear Temporal Logic (LTL) is widely used to specify high-level objectives for system policies, and it is highly desirable for autonomous systems to learn the optimal policy with respect to such specifications. However, learning the optimal policy from LTL specifications is not trivial. We present a model-free Reinforcement Learning (RL) approach that efficiently learns an optimal policy for an unknown stochastic system, modelled using Markov Decision Processes (MDPs). We propose a novel and more general product MDP, reward structure and discounting mechanism that, when applied in conjunction with off-the-shelf model-free RL algorithms, efficiently learn the optimal policy that maximizes the probability of satisfying a given LTL specification with optimality guarantees. We also provide improved theoretical results on choosing the key parameters in RL to ensure optimality. To directly evaluate the learned policy, we adopt probabilistic model checker PRISM to compute the probability of the policy satisfying such specifications. Several experiments on various tabular MDP environments across different LTL tasks demonstrate the improved sample efficiency and optimal policy convergence.

1 Introduction

Linear Temporal Logic (LTL) is a temporal logic language that can encode formulae regarding properties of an infinite sequence of logic propositions. LTL is widely used for the formal specification of high-level objectives for robotics and multi-agent systems, and it is desirable for a system or an agent in the system to learn policies with respect to these high-level specifications. Such systems are modelled as Markov Decision Processes (MDPs), where classic policy synthesis techniques can be adopted if the states and transitions of the MDP are known. However, when the transitions are not known *a priori*, the optimal policy needs to be learned through interactions with the MDP.

Model-free Reinforcement Learning [Sutton and Barto, 2018], a powerful method to train an agent to choose actions in order to maximize rewards over time in an unknown environment, is a perfect candidate for LTL specification policy learning. However, it is not straightforward to utilize reward-based RL to learn the optimal policy that maximizes the probability of satisfying LTL specifications [Alur *et al.*, 2021] due to the difficulty of deciding when, where, and how much reward to give to the agent. To this end, most works adopt a method to first transform the LTL specification into an automaton, then build a product MDP using the original environment MDP and the automaton, on which model-free RL algorithms are applied. However, one crucial obstacle still remains, and that is how to properly define the reward that leads an agent to the optimal satisfaction of LTL specification. Several algorithms have been proposed [Hahn *et al.*, 2019; Bozkurt *et al.*, 2019; Hasanbeig *et al.*, 2019] for learning LTL specifications, where, in order to ensure optimality, the key parameters are chosen depending on assumptions or knowledge of the environment MDP. These works normally only prove the existence of the optimality guarantee parameters or provide unnecessarily harsh bounds for them, which might lead to inefficient learning. In addition, it is unclear how to explicitly choose these parameters even with certain knowledge of the environment MDP. Furthermore, the assumed parameters are evaluated in experiments indirectly, either through inspection of the value function or comparison of the expected reward gained, making it difficult to tune the optimality parameters for LTL learning in general MDPs.

In this work, we propose a novel and more general product MDP, reward structure and discounting mechanism that, by leveraging model-free reinforcement learning algorithms, efficiently learns the optimal policy that maximizes the probability of satisfying the LTL specification with guarantees. We demonstrate improved theoretical results on the optimality of our product MDP and the reward structure, with more stringent analysis that yields better bounds on the optimality parameters. Moreover, this analysis sheds light on how to explicitly choose the optimality parameters based on the environment MDP. We also adopt counterfactual imagining that exploits the known high-level LTL specification to further improve the performance of our algorithm. Last but not least, we propose to use the PRISM model checker [Kwiatkowska *et al.*, 2011] to directly evaluate the satisfaction probability of the learned policies, providing a platform to directly compare algorithms and tune key parameters. We conduct exper-

iments on several common MDP environments with various challenging LTL tasks, and demonstrate the improved sample efficiency and convergence of our methods.

Our contributions include: (i) a novel product MDP design that incorporates an accepting states counter with a generalized reward structure; (ii) a novel reinforcement learning algorithm that converges to the optimal policy for satisfying LTL specifications, with theoretical optimality guarantees and theoretical analysis results on choosing the key parameters; (iii) the use of counterfactual imagining, a method to exploit the known structure of the LTL specification by creating imagination experiences through counterfactual reasoning; and (iv) direct evaluation of the proposed algorithms through a novel integration of probabilistic model checkers within the evaluation pipeline, with strong empirical results demonstrating better sample efficiency and training convergence.

1.1 Related Work

Most works on LTL learning with reward-based RL utilize a product MDP: a product of the environment MDP and an automaton translated from the LTL specification. Sadigh *et al.* [2014] first used deterministic Rabin automata to create this product with a discounted reward design to learn LTL, while later works adopted a new automaton design, limit-deterministic Büchi automata (LDBA) [Sickert *et al.*, 2016]. Hahn *et al.* [2019] adopted a product MDP with LDBA and augmented it with sink states to reduce the LTL satisfaction problem into a limit-average reward problem with optimality guarantees. Hahn *et al.* [2020] later modified this approach by including two discount factors with similar optimality guarantee results. Bozkurt *et al.* [2019] proposed a discounted reward learning algorithm on the product MDP with optimality guarantees, where the discount factor is chosen based on certain assumptions about the unknown environment MDP. To the best of our knowledge, these approaches are the only available to provide optimality guarantees for the full infinite-horizon LTL learning. However, many methods have nevertheless demonstrated empirical results for learning LTL. Hasanbeig *et al.* [2020; 2019] proposed an accepting frontier function as the reward for the product MDP, while Cai *et al.* [2021] extended this reward frontier to continuous control tasks.

Due to the difficulty of learning full LTL, many approaches focus on learning restricted finite LTL variants. Giacomo *et al.* [2013; 2019] proposed the LTLf variant and a corresponding reinforcement learning algorithm; Littman *et al.* [2017] formulated a learning algorithm for the GLTL variant; Aksaray *et al.* [2016] proposed to learn Signal Temporal Logic and Li *et al.* [2016] a truncated LTL variant for robotics applications.

Another related line of work leverages automata to learn non-Markovian rewards. Toro Icarte *et al.* [2022; 2018] defined a reward machine automaton to represent high-level non-Markovian rewards, while Camacho *et al.* [2019] introduced a method to learn finite LTL specifications by transforming them into reward machines. However, the expressiveness of reward machines is strictly weaker than that of LTL. Lastly, there are works that exploit other high-level logic

specifications to facilitate learning [Andreas *et al.*, 2016; Jiang *et al.*, 2021; Jothimurugan *et al.*, 2020; Jothimurugan *et al.*, 2021], but they are less relevant to reinforcement learning from LTL.

2 Preliminaries

Before formulating our problem, we provide preliminary background on Markov decision processes, linear temporal logic, and reinforcement learning.

2.1 Markov Decision Processes

Definition 1 (Markov Decision Process [Littman, 2001]). *A Markov decision process (MDP) $\mathcal{M}$ is a tuple $(S, s_0, A, T, \mathcal{AP}, L, r, \gamma)$, where S is a finite set of states, $s_0 \in S$ is the initial state, A is a finite set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is the probabilistic transition function, $\mathcal{AP}$ is the set of atomic propositions, $L : S \rightarrow 2^{\mathcal{AP}}$ is the proposition labeling function, $r : S \times A \times S \rightarrow \mathbb{R}$ is a reward function and $\gamma : S \rightarrow (0, 1]$ is a discount function. Let $A(s)$ denote the set of available actions at state s, then, for all $s \in S$, it holds that $\sum_{s' \in S} T(s, a, s') = 1$ if $a \in A(s)$ and 0 otherwise.*

An infinite path is a sequence of states $\sigma = s_0, s_1, s_2...,$ where there exist $a_{i+1} \in A(s_i)$ such that $T(s_i, a_{i+1}, s_{i+1}) > 0$ for all $i \geq 0$, and a finite path is a finite such sequence. We denote the set of infinite and finite paths of the MDP $\mathcal{M}$ as $Paths^{\mathcal{M}}$ and $FPaths^{\mathcal{M}}$, respectively. We use $\sigma[i]$ to denote s_i, and $\sigma[: i]$ and $\sigma[i :]$ to denote the prefix and suffix of the path, respectively. Furthermore, we assume self-loops: if $A(s) = \varnothing$ for some state s, we let $T(s, a, s) = 1$ for some $a \in A$ and $A(s) = a$ such that all finite paths can be extended to an infinite one.

A finite-memory policy π for $\mathcal{M}$ is a function $\pi : FPaths^{\mathcal{M}} \rightarrow D(A)$ such that $supp(\pi(\sigma)) \subseteq A(\sigma[-1])$, where $D(A)$ denotes a distribution over A, $supp(d)$ denotes the support of the distribution and $\sigma[-1]$ is the last state of a finite path σ. A policy π is memoryless if it only depends on the current state, *i.e.*, $\sigma[-1] = \sigma'[-1]$ implies $\pi(\sigma) = \pi(\sigma')$, and a policy is deterministic if $\pi(\sigma)$ is a point distribution for all $\sigma \in FPaths^{\mathcal{M}}$. For a deterministic memoryless policy, we let $\pi(s)$ represent $\pi(\sigma)$ where $\sigma[-1] = s$.

Let $Paths^{\mathcal{M}}_{\pi} \subseteq Paths^{\mathcal{M}}$ denote the subset of infinite paths that follow policy π and we define the probability space $(Paths^{\mathcal{M}}_{\pi}, \mathcal{F}_{Paths^{\mathcal{M}}_{\pi}}, \mathbb{P}_{\pi})$ over $Paths^{\mathcal{M}}_{\pi}$ in the standard way. Then, for any function $f : Paths^{\mathcal{M}}_{\pi} \rightarrow \mathbb{R}$, let $\mathbb{E}_{\pi}[f]$ be the expectation of f over the infinite paths of $\mathcal{M}$ following π.

A Markov chain (MC) induced by $\mathcal{M}$ and deterministic memoryless policy π is a tuple $\mathcal{M}_{\pi} = (S, s_0, T_{\pi}, \mathcal{AP}, L)$, where $T_{\pi}(s, s') = T(s, \pi(s), s')$. A sink (bottom) strongly connected component (BSCC) of a MC is a set of states $C \subseteq S$ such that, for all pairs $s_1, s_2 \in C$, there exists a path from s_1 to s_2 following the transition function T_{π} (strongly connected), and there exists no state $s' \in S \setminus C$ such that $T_{\pi}(s, s') > 0$ for all $s \in C$ (sink).

2.2 Linear Temporal Logic

Linear Temporal Logic (LTL) provides a high-level description for specifications of a system. LTL is very expressive and can describe specifications with infinite horizon.

Definition 2 ([Baier and Katoen, 2008]). *An LTL formula over atomic propositions $\mathcal{AP}$ is defined by the grammar:*

$$\varphi ::= true \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathsf{X}\,\varphi \mid \varphi_1 \mathsf{U}\,\varphi_2, \quad p \in \mathcal{AP},$$

where X represents next and U represents until. Other Boolean and temporal operators are derived as follows: or: $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$; implies: $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$; eventually: $\mathsf{F}\,\varphi = true\mathsf{U}\,\varphi$; and always: $\mathsf{G}\,\varphi = \neg(\mathsf{F}\,\neg\varphi)$.

The satisfaction of an LTL formula φ by an infinite path $\sigma \in Paths^{\mathcal{M}}$ is denoted by $\sigma \models \varphi$, and is defined by induction on the structure of φ:

$$\sigma \text{ satisfies } \varphi \text{ if } \quad p \in L(\sigma[0]) \quad \text{for } p \in \mathcal{AP};$$
$$\sigma[1:] \models \varphi \quad \text{for } \mathsf{X}\,\varphi;$$
$$\exists i, \sigma[i] \models \varphi_2 \text{ and } \forall j < i, \sigma[j] \models \varphi_1 \quad \text{for } \varphi_1 \mathsf{U}\,\varphi_2,$$

with the satisfaction of Boolean operators defined by their default meaning.

2.3 Reinforcement Learning

Reinforcement learning [Sutton and Barto, 2018] teaches an agent in an unknown environment to select an action from its action space, in order to maximize rewards over time. In most cases the environment is modelled as an MDP $\mathcal{M} = (S, s_0, A, T, \mathcal{AP}, L, r, \gamma)$. Given a deterministic memoryless policy π, at each time step t, let the agent's current state be s_t, then the action $a = \pi(s_t)$ is chosen and the next state $s_{t+1} \sim T(s_t, a, \cdot)$ together with the immediate reward $r(s_t, a, s_{t+1})$ is received from the environment. Then, starting at $s \in S$ and time step t, the expected discounted reward following π is

$$G_t^{\pi}(s) = \mathbb{E}_{\pi}\Big[\sum_{i=t}^{\infty}\Big(\prod_{j=t}^{i-1}\gamma(s_j)\Big) \cdot r(s_i, a_i, s_{i+1}) \mid s_t = s\Big], \quad (1)$$

where $\prod_{j=t}^{t-1} := 1$. The agent's goal is to learn the optimal policy that maximizes the expected discounted reward. Note that we defined a discount function instead of a constant discount factor because it is essential for our proposed LTL learning algorithm to discount the reward depending on the current MDP state.

Q-learning [Watkins and Dayan, 1992] is a widely used approach for model-free RL. It utilizes the idea of the Q function $Q^{\pi}(s, a)$, which is the expected discounted reward of taking action a at state s and following policy π after that. The Q function for all optimal policies satisfies the Bellman optimality equations:

$$Q^*(s, a) = \sum_{s' \in S} T(s, a, s')\big(r(s, a, s') \quad (2)$$
$$+ \gamma(s) \max_{a' \in A} Q^*(s', a')\big) \,\forall a \in A, s \in S.$$

At each iteration of the Q-learning algorithm, the agent's experiences, i.e, the next state s' and immediate reward $r(s, a, s')$, are used to update the Q function:

$$Q(s, a) \xleftarrow{\alpha} r(s, a, s') + \gamma(s) \max_{a' \in A} Q(s', a'), \quad (3)$$

where α is the learning rate and $x \xleftarrow{\alpha} y$ represents $x \leftarrow x + \alpha(y - x)$. In addition, the optimal policy can be recovered from the optimal Q function $Q^*(s, a)$ by selecting the action a with the highest state-action pair value in each state s. Q-learning converges to the optimal Q function in the limit given that each state-action pair is visited infinitely often [Watkins and Dayan, 1992], and thus learns the optimal policy.

3 Our Method

Our goal is to formulate a model-free reinforcement learning approach to efficiently learn the optimal policy that maximizes the probability of satisfying an LTL specification with guarantees. We now give an overview of our method. We first transform the LTL objective φ into a limit-deterministic Büchi automaton. Then, we introduce a novel product MDP and define a generalized reward structure on it. With this reward structure, we propose a Q-learning algorithm that adopts a collapsed Q function to learn the optimal policy with optimality guarantees. Lastly, we enhance our algorithm with counterfactual imagining that exploits the automaton structure to improve performance while maintaining optimality.

3.1 Problem Formulation

Given an MDP $\mathcal{M} = (S, s_0, A, T, \mathcal{AP}, L, r, \gamma)$ with unknown states and transitions and an LTL objective φ, for any policy π of the MDP $\mathcal{M}$, let $\mathbb{P}_{\pi}(s \models \varphi)$ denote the probability of paths from state s following π satisfying the LTL formula φ:

$$\mathbb{P}_{\pi}(s \models \varphi) = \mathbb{P}_{\pi}\{\sigma \in Paths_{\pi}^{\mathcal{M}} \mid \sigma[0] = s, \sigma \models \varphi\}. \quad (4)$$

Then, we would like to design a model-free RL algorithm that learns a deterministic memoryless optimal policy π_{φ} that maximizes the probability of $\mathcal{M}$ satisfying φ:

$$\mathbb{P}_{\pi_{\varphi}}(s \models \varphi) = \max_{\pi} \mathbb{P}_{\pi}(s \models \varphi) \quad \forall s \in S. \quad (5)$$

3.2 Limit-Deterministic Büchi Automata

We first transform the LTL specifications into automata. The common choices of automata include deterministic Rabin automata and non-deterministic Büchi automata. In this work, we adopt a Büchi automata variant called limit-deterministic Büchi automata (LDBA) [Sickert *et al.*, 2016].

Definition 3. *A non-deterministic Büchi automaton is an automaton $\mathcal{A} = (\mathcal{AP}, \mathcal{Q}, q_0, \Delta, \mathcal{F})$, where $\mathcal{AP}$ is the set of atomic propositions, $\mathcal{Q}$ is a finite set of states, $q_0 \in \mathcal{Q}$ is the initial state and $\mathcal{F} \subseteq \mathcal{Q}$ is the set of accepting states. Let $\Sigma = 2^{\mathcal{AP}} \cup \{\epsilon\}$ be a finite alphabet, then the transition function is given by $\Delta : \mathcal{Q} \times \Sigma \rightarrow 2^{\mathcal{Q}}$.*

Definition 4 (LDBA). *A Büchi automaton is limit-deterministic if $\mathcal{Q}$ can be partitioned into a deterministic set and a non-deterministic set, that is, $\mathcal{Q} = \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{D}}$, where $\mathcal{Q}_{\mathcal{N}} \cap \mathcal{Q}_{\mathcal{D}} = \varnothing$, such that*

1. *$\mathcal{F} \subseteq \mathcal{Q}_{\mathcal{D}}$ and $q_0 \in \mathcal{Q}_{\mathcal{N}}$;*
2. *$|\Delta(q, \nu)| \leq 1$ for all $q \in \mathcal{Q}_{\mathcal{N}}$ and $\nu \neq \epsilon \in \Sigma$;*
3. *$\Delta(q, \nu) \subseteq \mathcal{Q}_{\mathcal{D}}$ and $|\Delta(q, \nu)| \leq 1$ for all $q \in \mathcal{Q}_{\mathcal{D}}$ and $\nu \in \Sigma$;*

An LDBA is a Büchi automaton where the non-determinism is limited in the initial component $\mathcal{Q}_\mathcal{N}$ of the automaton. An LDBA starts in a non-deterministic initial component and then transitions into a deterministic accepting component $\mathcal{Q}_\mathcal{D}$ through ϵ-transitions after reaching an accepting state, where all transitions after this point are deterministic. We follow the formulation of Bozkurt *et al.* [2019] to extend the alphabets with an ϵ-transition that handles all the non-determinism, meaning only ϵ can transition the automaton state to more then 1 states: $|\Delta(q, \epsilon)| > 1$. This allows the MDP to synchronise with the automaton, which we will discuss in detail in Section 3.3.

An infinite word $\mathcal{W} \in \Sigma^\omega$, where Σ^ω is the set of all infinite words over the alphabet Σ, is accepted by a Büchi automaton $\mathcal{A}$ if there exists an infinite automaton run $\theta \in \mathcal{Q}^\omega$ from q_0, where $\theta[t+1] \in \Delta(\theta[t], \mathcal{W}[t]), \forall t \geq 0$, such that $\inf(\theta) \cap \mathcal{F} \neq \varnothing$, where $\inf(\theta)$ is the set of automaton states that are visited infinitely often in the run θ.

LDBAs are as expressive as the LTL language, and the satisfaction of any given LTL specification φ can be evaluated on the LDBA derived from φ. We use Rabinizer 4 [Křetínský *et al.*, 2018] to transform LTL formulae into LDBAs. In Figure 1 (left) we give an example of the LDBA derived from the LTL formula "FG a & G !c", where state 1 is the accepting state. LDBAs are different from reward machines [Toro Icarte *et al.*, 2022] because they can express properties satisfiable by infinite paths, which is strictly more expressive than reward machines, and they have different accepting conditions.

3.3 Product MDP

In this section, we propose a novel product MDP of the environment MDP, an LDBA and an integer counter, where the transitions for each component are synchronised. Contrary to the standard product MDP used in the literature [Bozkurt *et al.*, 2019; Hahn *et al.*, 2019; Hasanbeig *et al.*, 2020], this novel product MDP incorporates a counter that counts the number of accepting states visited by paths starting at the initial state.

Definition 5 (Product MDP). *Given an MDP $\mathcal{M} = (S, s_0, A, T, \mathcal{AP}, L, r, \gamma)$, an LDBA $\mathcal{A} = (\mathcal{AP}, \mathcal{Q}, q_0, \Delta, \mathcal{F})$ and $K \in \mathbb{N}$, we construct the product MDP as follows:*

$$\mathcal{M}^\times = \mathcal{M} \times \mathcal{A} \times [0..K] = (S^\times, s_0^\times, A^\times, T^\times, \mathcal{F}^\times, r^\times, \gamma^\times),$$

where the product states $S^\times = S \times \mathcal{Q} \times [0..K]$, the initial state $s_0^\times = (s_0, q_0, 0)$, the product actions $A^\times = A \cup \{\epsilon_q \mid q \in \mathcal{Q}\}$, the accepting set $\mathcal{F}^\times = S \times \mathcal{F} \times [0..K]$, and the product transitions $T^\times : S^\times \times A^\times \times S^\times \to [0, 1]$, which are defined as:

$$T^\times((s, q, n), a, (\hat{s}, \hat{q}, n)) \tag{6}$$

$$= \begin{cases} T(s, a, \hat{s}) & \text{if } a \in A \text{ and } \hat{q} \in \Delta(q, L(s)) \setminus \mathcal{F}; \\ 1 & \text{if } a = \epsilon_{\hat{q}}, \hat{q} \in \Delta(q, \epsilon) \text{ and } \hat{s} = s; \\ 0 & \text{otherwise} . \end{cases}$$

$$T^\times((s, q, n), a, (\hat{s}, \hat{q}, \min(n+1, K))) \tag{7}$$

$$= \begin{cases} T(s, a, \hat{s}) & \text{if } a \in A \text{ and } \hat{q} \in \Delta(q, L(s)) \cap \mathcal{F}; \\ 0 & \text{otherwise} . \end{cases}$$

where all other transitions are equal to 0. The product reward $r^\times : S^\times \times A^\times \times S^\times \to \mathbb{R}$ and the product discount function $\gamma^\times : S^\times \to (0, 1]$ that are suitable for LTL learning are defined later in Definition 6.

Furthermore, an infinite path σ of $\mathcal{M}^\times$ satisfies the Büchi condition $\varphi_\mathcal{F}$ if $\inf(\sigma) \cap \mathcal{F}^\times \neq \varnothing$. With a slight abuse of notation we denote this condition in LTL language as $\sigma \models$ GF $\varphi_\mathcal{F}$, meaning for all $M \in \mathbb{N}$, there always exists $s^\times \in \mathcal{F}^\times$ that will be visited in $\sigma[M :]$.

When an MDP action $a \in A$ is taken in the product MDP $\mathcal{M}^\times$, the alphabet used to transition the LDBA is deduced by applying the label function to the current environment MDP state: $L(s) \in 2^{\mathcal{AP}}$. In this case, the LDBA transition $\delta(q, L(s))$ is deterministic. Otherwise, if an ϵ-action $\epsilon_{\hat{q}} \in \{\epsilon_q \mid q \in \mathcal{Q}\}$ is taken, LDBA is transitioned with an ϵ-transition, and the non-determinism of $\delta(q, \epsilon)$ is resolved by transitioning the automaton state to $\hat{q}$. The K counter value is equal to 0 in the initial state, and each time an accepting state is reached, the counter value increases by one until it is capped at K.

Example 1. *We motivate our product MDP structure of Definition 5 through an example. In Figure 1 (right), we have a grid environment where the agent can decide to go up, down, left or right. The task is to visit states labeled "a" infinitely often without visiting "c" as described by the LDBA in Figure 1 (left). The MDP starts at (1,0), with walls denoted by solid gray squares. The states in the first row only allow action right as denoted by the right pointing triangle, which leads to a sink at (0,9). There is also a probabilistic gate at (2,0) that transitions the agent randomly to go down or right, and if the agent reaches (2,1), the accepting sink at (2,9) is reachable. Therefore, the optimal policy is to go down from the start and stay in (2,9) if the probabilistic gate at (2,0) transitions the agent to the right. The probability of satisfying this task is the probability of the gate sending you to the right. Intuitively, this environment has some initial accepting states that are easy to explore, but lead to non-accepting sinks, whereas the true optimal path requires more exploration. If we set $K = 10$ in the product MDP for this task, we can assign very small rewards for the initially visited accepting states and gradually increase the reward as more accepting states are visited to encourage exploration and guide the agent to the optimal policy.*

Next, we provide a theorem, which states that the product MDP with Büchi condition $\varphi_\mathcal{F}$ is equivalent, in terms of the optimal policy, to the original MDP with LTL specification φ. The proof of this theorem is provided in Appendix A.1[1].

Theorem 1 (Satisfiability Equivalence). *For any product MDP $\mathcal{M}^\times$ that is induced from LTL formula φ, we have that*

$$\sup_\pi \mathbb{P}_\pi(s_0 \models \varphi) = \sup_{\pi^\times} \mathbb{P}_{\pi^\times}(s_0^\times \models \text{GF } \varphi_\mathcal{F}). \tag{8}$$

Furthermore, a deterministic memoryless policy that maximizes the probability of satisfying the Büchi condition $\varphi_\mathcal{F}$ on the product MDP $\mathcal{M}^\times$, starting from the initial state, induces

---

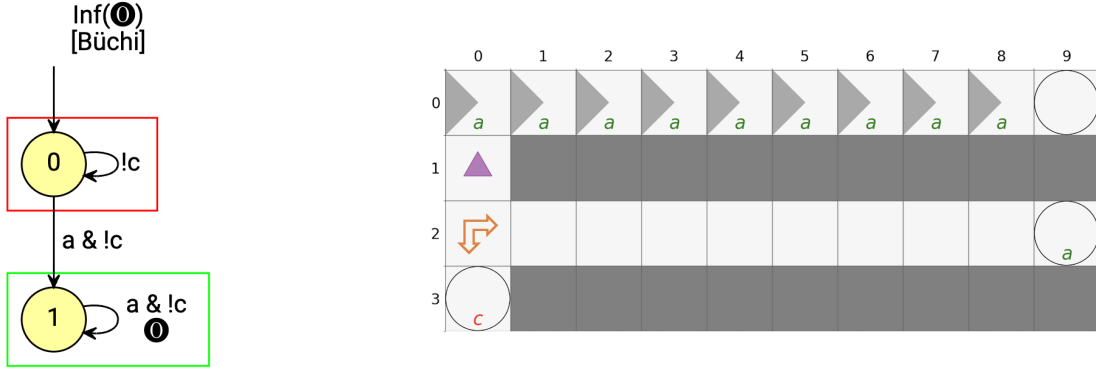[1]See Appendix in the full version of this paper on arXiv: https://arxiv.org/abs/2305.01381 [Shao and Kwiatkowska, 2023]

Figure 1: An LDBA for "FG a & G !c" (left), and a probabilistic gate MDP (right) motivating the K counter (see Example 1).

a deterministic finite-memory optimal policy that maximizes the probability of satisfying φ on the original MDP $\mathcal{M}$ from the initial state.

Reward Structure for LTL Learning

We first define a generalized reward structure in $\mathcal{M}^\times$ for LTL learning, and then prove the equivalence between acquiring the highest expected discounted reward and achieving the highest probability of satisfying $\varphi_\mathcal{F}$ under this reward structure.

Definition 6 (Reward Structure). *Given a product MDP $\mathcal{M}^\times$ and a policy π, the product reward function $r^\times : S^\times \times A^\times \times S^\times \to \mathbb{R}$ is suitable for LTL learning if*

$$r^\times((s, q, n), a^\times, (s', q', n')) = \begin{cases} R_n & \text{if } q' \in \mathcal{F}; \\ 0 & \text{otherwise}, \end{cases} \quad (9)$$

where $R_n \in (0, U]$ are constants for $n \in [0..K]$ and $U \in (0, 1]$ is an upper bound on the rewards. The rewards are non-zero only for accepting automaton states, and depend on the value of the K counter.

Then, given a discount factor $\gamma \in (0, 1]$, we define the product discount function $\gamma^\times : S^\times \to (0, 1]$ as

$$\gamma^\times(s_j^\times) = \begin{cases} 1 - r_j^\times & \text{if } r_j^\times > 0; \\ \gamma & \text{otherwise}, \end{cases}$$

and the expected discounted reward following policy π starting at $s^\times$ and time step t is

$$G_t^\pi(s^\times) = \mathbb{E}_\pi\left[\sum_{i=t}^{\infty}\left(\prod_{j=t}^{i-1} \gamma^\times(s_j^\times)\right) \cdot r^\times(s_i^\times, a_i^\times, s_{i+1}^\times) \mid s_t^\times = s^\times\right]. \quad (10)$$

The highest K value reached in a path (i.e., the number of accepting states visited in the path) acts as a measure of how promising that path is for satisfying $\varphi_\mathcal{F}$. By exploiting it, we can assign varying rewards to accepting states to guide the agent, as discussed in the motivating example in Section 3.3. Next, we provide a lemma stating the properties of the product MDP regarding the satisfaction of the Büchi condition $\varphi_\mathcal{F}$.

Lemma 1. *Given a product MDP $\mathcal{M}^\times$ with its corresponding LTL formula φ and a policy π, we write $\mathcal{M}_\pi^\times$ for the induced Markov chain from π. Let $B_\mathcal{F}^\times$ denote the set of states*

that belong to accepting BSCCs of $\mathcal{M}_\pi^\times$, and $B_\varnothing^\times$ denote the set of states that belong to rejecting BSCCs:

$$B_\mathcal{F}^\times := \{s^\times \mid s^\times \in B \in BSCC(\mathcal{M}_\pi^\times), B \cap \mathcal{F}^\times \neq \varnothing\}; \quad (11)$$

$$B_\varnothing^\times := \{s^\times \mid s^\times \in B \in BSCC(\mathcal{M}_\pi^\times), B \cap \mathcal{F}^\times = \varnothing\}, \quad (12)$$

where $BSCC(\mathcal{M}_\pi^\times)$ is the set of all BSCCs of $\mathcal{M}_\pi^\times$. We further define more general accepting and rejecting sets:

$$B_\mathcal{F} := \{(s, q, n) \mid \exists n' \in [0..K] : (s, q, n') \in B_\mathcal{F}^\times\}; \quad (13)$$

$$B_\varnothing := \{(s, q, n) \mid \exists n' \in [0..K] : (s, q, n') \in B_\varnothing^\times\}. \quad (14)$$

We then have that $\mathbb{P}_\pi(s^\times \models \mathsf{GF}\ \varphi_\mathcal{F}) = 1\ \forall s^\times \in B_\mathcal{F}$, $\mathbb{P}_\pi(s^\times \models \mathsf{GF}\ \varphi_\mathcal{F}) = 0\ \forall s^\times \in B_\varnothing$ and $B_\varnothing \cap \mathcal{F}^\times = \varnothing$. Furthermore, $B_\mathcal{F}$ and $B_\varnothing$ are sink sets, meaning once the set is reached, no states outside the set can be reached.

The proof of this lemma is provided in Appendix A.2. Using this lemma, we can now state and proof the main theorem of this paper.

Theorem 2 (Optimality Guarantee). *Given an LTL formula φ and a product MDP $\mathcal{M}^\times$, there exists an upper bound $U \in (0, 1]$ for rewards and a discount factor $\gamma \in (0, 1]$ such that for all product rewards $r^\times$ and product discount functions $\gamma^\times$ satisfying Definition 6, the optimal deterministic memoryless policy π_r that maximizes the expected discounted reward $G_0^{\pi_r}(s_0^\times)$ is also an optimal policy π_φ that maximizes the probability of satisfying the Büchi condition $\mathbb{P}_{\pi_\varphi}(s_0^\times \models \mathsf{GF}\ \varphi_\mathcal{F})$ on the product MDP $\mathcal{M}^\times$.*

Proof sketch. We now present a sketch of the proof to provide intuition for the main steps and the selection of key parameters. The full proof is provided in Appendix A.3.

To ensure optimality, given a policy π with the product MDP $\mathcal{M}^\times$ and the LTL formula φ, we want to demonstrate a tight bound between the expected discounted reward following π and the probability of π satisfying φ, such that maximizing one quantity is equivalent to maximizing the other.

At a high level, we want to select the two key parameters, the reward upper bound $U \in (0, 1]$ and the discount factor $\gamma \in (0, 1]$, to adequately bound: (i) the rewards given for paths that eventually reach rejecting BSCCs (thus not satisfying the LTL specification); and (ii) the discount of rewards received from rejecting states for paths that eventually reach accepting BSCCs.

Algorithm 1: KC Q-learning from LTL

Input: environment MDP $\mathcal{M}$, LTL formula φ
1: translate φ into an LBDA $\mathcal{A}$
2: construct product MDP $\mathcal{M}^\times$ using $\mathcal{M}$ and $\mathcal{A}$
3: initialize Q for each s and q pair
4: **for** $l \leftarrow 0$ *to max_episode* **do**
5:     initialize $(s, q, n) \leftarrow (s_0, q_0, 0)$
6:     **for** $t \leftarrow 0$ *to max_timestep* **do**
7:         get policy π derived from Q (*e.g.*, ϵ-greedy)
8:         take action $a^\times \leftarrow \pi((s, q, n))$ in $\mathcal{M}^\times$
9:         get next product state (s', q', n')
10:        $r \leftarrow r^\times((s, q, n), a^\times, (s', q', n'))$
11:        $\gamma \leftarrow \gamma'(s, q, n)$
12:        $Q((s, q), a^\times) \overset{\alpha}{\Leftarrow} r + \gamma \max_{a \in A^\times} Q((s', q'), a)$
13:        update $(s, q, n) \leftarrow (s', q', n')$
14: gets greedy policy π_φ from Q
15: **return** induced policy on $\mathcal{M}$ by removing ϵ-actions

Algorithm 2: CF+KC Q-learning from LTL

Input: environment MDP $\mathcal{M}$, LTL formula φ
1: translate φ into an LBDA $\mathcal{A}$
2: construct product MDP $\mathcal{M}^\times$ using $\mathcal{M}$ and $\mathcal{A}$
3: initialize Q for each s and q pair
4: **for** $l \leftarrow 0$ *to max_episode* **do**
5:     initialize $(s, q, n) \leftarrow (s_0, q_0, 0)$
6:     **for** $t \leftarrow 0$ *to max_timestep* **do**
7:         get policy π derived from Q (e.g., ϵ-greedy)
8:         get action $a^\times \leftarrow \pi((s, q, n))$ in $\mathcal{M}^\times$
9:         **for** $\bar{q} \in \mathcal{Q}$ **do**
10:            get counterfactual imagination $(s', \bar{q}', n')$
                 by taking action $a^\times$ at $(s, \bar{q}, n)$
11:            $r \leftarrow r^\times((s, \bar{q}, n), a^\times, (s', \bar{q}', , n'))$
12:            $\gamma \leftarrow \gamma'(s, \bar{q}, n)$
13:            $Q((s, \bar{q}), a^\times) \overset{\alpha}{\Leftarrow}$
                 $r + \gamma \max_{a \in A^\times} Q((s', \bar{q}'), a)$
14:        obtain (s', q', n') using action $a^\times$ at (s, q, n)
15:        update $(s, q, n) \leftarrow (s', q', n')$
16: gets greedy policy π_φ from Q
17: **return** induced policy on $\mathcal{M}$ by removing ϵ-actions

We informally denote by $C_\varnothing^\pi$ the expected number of visits to accepting states before reaching a rejecting BSCC, and (i) can be sufficiently bounded by selecting $U = 1/C_\varnothing^\pi$. Next, we informally write $C_\mathcal{F}^\pi$ for the expected number of rejecting states visited before reaching an accepting BSCC, and denote by N^π the expected steps between visits of accepting states in the accepting BSCC. Intuitively, for (ii), we bound the amount of discount before reaching the accepting BSCC using $C_\mathcal{F}^\pi$, and we bound the discount after reaching the BSCC using N^π, yielding $\gamma = 1 - 1/(C_\varnothing^\pi * N^\pi + C_\mathcal{F}^\pi)$.

In practice, using upper bounds of $C_\varnothing^\pi, C_\mathcal{F}^\pi$ and N^π instead also ensures optimality, and those bounds can be deduced from assumptions about, or knowledge of, the MDP. $\square$

As shown in the proof sketch, selecting $U = 1/C_\varnothing^\pi$ and $\gamma = 1 - 1/(C_\varnothing^\pi * N^\pi + C_\mathcal{F}^\pi)$ is sufficient to ensure optimality. Using the example of the probabilistic gate MDP in Figure 1 (right), we have that $C_\varnothing^\pi \approx C_\mathcal{F}^\pi \leq 10$ and $N^\pi = 1$, so choosing $U = 0.1$ and $\gamma = 0.95$ is sufficient to guarantee optimality. For more general MDPs, under the common assumption that the number of states $|S|$ and the minimum non-zero transition probability $p_{\min} := \min_{s,a,s'}\{T(s, a, s') > 0\}$ are known, $C_\varnothing^\pi$ and $C_\mathcal{F}^\pi$ can be upper bounded by $|S|/p_{\min}$, while N^π can be upper bounded by $|S|$.

3.4 LTL Learning With Q-Learning

Employing this product MDP $\mathcal{M}^\times$ and its reward structure, we present Algorithm 1 (KC), a model-free Q-learning algorithm for LTL specifications utilizing the K counter product MDP. The product MDP is constructed on the fly as we explore: for action $a^\times \in A$, observe the next environment state s' by taking action $a^\times$ in environment state s. Then, we compute the next automaton state q' using transition function $\Delta(q, L(s))$ and counter n depending on whether $n \leq K$ and $q' \in \mathcal{F}$. If $a^\times \in \{\epsilon_q \mid q \in \mathcal{Q}\}$, update q' using the ϵ-transition and leave environment state s and counter n unchanged. However, directly adopting Q-learning on this product MDP yields a Q function defined on the whole product

state space $S^\times$, meaning the agent needs to learn the Q function for each K value. To improve efficiency, we propose to define the Q function on the environment states S and automaton states $\mathcal{Q}$ only, and for a path σ of $\mathcal{M}^\times$, the update rule for the Q function at time step t is:

$$Q_{t+1}((s_t, q_t), a_t^\times) \overset{\alpha}{\Leftarrow} r^\times(s_t^\times, a_t^\times, s_{t+1}^\times) \tag{15}$$
$$+ \gamma'(s_t^\times) \max_{a^\times \in A^\times} Q_t((s_{t+1}, q_{t+1}), a^\times),$$

where $s_t^\times = (s_t, q_t, n_t)$, $a_t^\times$ is the action taken at time step t, $s_{t+1}^\times = (s_{t+1}, q_{t+1}, n_{t+1})$ is the next product state, and α is the learning rate. We claim that, with this collapsed Q function, the algorithm returns the optimal policy for satisfying φ because the optimal policy is independent from the K counter, with the proof provided in Appendix A.4.

Theorem 3. *Given an environment MDP $\mathcal{M}$ and an LTL specification φ with appropriate discount factor γ and reward function $r^\times$ satisfying Theorem 2, Q-learning for LTL described in Algorithm 1 converges to an optimal policy π_φ that maximizes the probability of satisfying φ on $\mathcal{M}$.*

3.5 Counterfactual Imagining

Additionally, we propose a method to exploit the structure of the product MDP, specifically the LDBA, to facilitate learning. We use counterfactual reasoning to generate synthetic imaginations: from one state in the environment MDP, imagine we are at each of the automaton states while taking the same actions.

If the agent is at product state (s, q, n) and an action $a^\times \in A$ is chosen, for each $\bar{q} \in \mathcal{Q}$, the next state $(s', \bar{q}', n')$ by taking action $a^\times$ from $(s, \bar{q}, n)$ can be computed by first taking the action in environment state, and then computing the next automaton state $\bar{q}' = \Delta(\bar{q}, L(s))$ and the next K
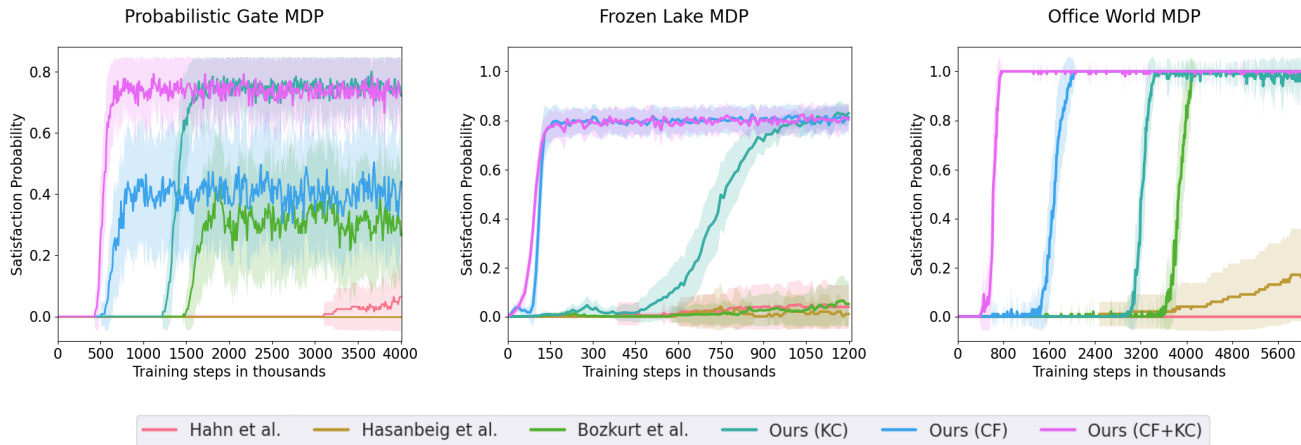
Figure 2: The experimental results on various MDPs and tasks.

value n'. The reward for the agent is $r^\times(s, \bar{q}, \bar{n})$, and we can therefore update the Q function with this enriched set of experiences. These experiences produced by counterfactual imagining are still sampled from $\mathcal{M}^\times$ following the transition function $T^\times$, and hence, when used in conjunction with any off-policy learning algorithms like Q-learning, the optimality guarantees of the algorithm are preserved.

As shown in Algorithm 2 (CF-KC), counterfactual imagining (CF) can be incorporated into our KC Q-learning algorithm by altering a few lines (line 9-12 in Algorithm 2) of code, and it can also be used in combination with other automata product MDP RL algorithms for LTL. Note that the idea of counterfactual imagining is similar to that proposed by Toro Icarte *et al.* [2022], but our approach has adopted LDBAs in the product MDPs for LTL specification learning.

4 Experimental Results

We evaluate our algorithms on various MDP environments, including the more realistic and challenging stochastic MDP environments[2]. We propose a method to directly evaluate the probability of satisfying LTL specifications by employing probabilistic model checker PRISM [Kwiatkowska *et al.*, 2011]. We build the induced MC $\mathcal{M}_\pi$ from the environment MDP and the policy in PRISM format, and adopt PRISM to compute the exact satisfaction probability of the given LTL specification. We utilize tabular Q-learning as the core off-policy learning method to implement our three algorithms: Q-learning with K counter reward structure (KC), Q-learning with K counter reward structure and counterfactual imagining (CF+KC), and Q-learning with only counterfactual imagining (CF), in which we set $K = 0$. We compare the performance of our methods against the methods proposed by Bozkurt *et al.* [2019], Hahn *et al.* [2019] and Hasanbeig *et al.* [2020]. Note that our KC algorithm, in the special case that $K = 0$ with no counterfactual imagining, is algorithmically equivalent to Bozkurt *et al.* [2019] when setting their parameter $\gamma_B = 1 - U$. Our methods differ from all other

existing methods to the best of our knowledge. The details and setup of the experiments are given in Appendix B.

We set the learning rate $\alpha = 0.1$ and $\epsilon = 0.1$ for exploration. We also set a relatively loose upper bound on rewards $U = 0.1$ and discount factor $\gamma = 0.99$ for all experiments to ensure optimality. Note that the optimality of our algorithms holds for a family of reward structures defined in Definition 6, and for experiments we opt for a specific reward function that linearly increases the reward for accepting states as the value of K increases, namely $r_n = U \cdot n/K \; \forall n \in [0..K]$, to facilitate training and exploration. The Q function is optimistically initialized by setting the Q value for all available state-action pairs to $2U$. All experiments are run 100 times, where we plot the average satisfaction probability with half standard deviation in the shaded area.

First, we conduct experiments on the probabilistic gate MDP described in Example 1 with task "FG a & G !c", which means reaching only states labeled "a" in the future while never reaching "c" labeled states. We set $K = 10$ for this task, and in Figure 2 (left), compared to the other three methods, our method KC achieved better sample efficiency and convergence and CF demonstrates better sample efficiency while still lacking training stability. The best performance is achieved by CF+KC, while other methods either exhibit slower convergence (Bozkurt *et al.* [2019] and Hahn *et al.* [2019]) or fail to converge (Hasanbeig *et al.* [2020]) due to the lack of theoretical optimality guarantees.

The second MDP environment is the 8×8 frozen lake environment from OpenAI Gym [Brockman *et al.*, 2016]. This environment consists of frozen lake tiles, where the agent has 1/3 chance of moving in the intended direction and 1/3 of going sideways each, with details provided in Appendix B.2. The task is "(GF a | GF b) & G !h", meaning to always reach lake camp "a" or lake camp "b" while never falling into holes "h". We set $K = 10$ for this task, and in Figure 2 (middle), we observe significantly better sample efficiency for all our methods, especially for CF+KC and CF, which converge to the optimal policy at around 150k training steps. The other three methods, on the other hand, barely start to converge at 1200k training steps. CF performs especially well in this task

---

[2]The implementation of our algorithms and experiments can be found on GitHub: https://github.com/shaodaqian/rl-from-ltl
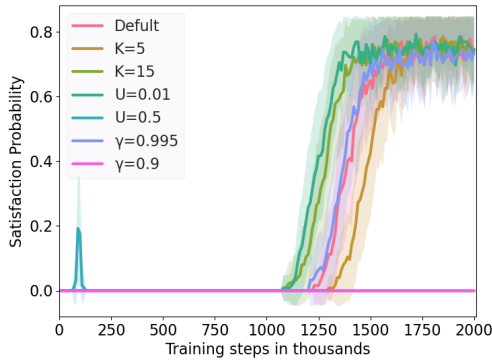
Figure 3: Sensitivity analysis on key parameters.

because the choice of always reaching "a" or "b" can be considered simultaneously during each time step, reducing the sample complexity to explore the environment.

Lastly, we experiment on a slight modification of the more challenging office world environment proposed by Toro Icarte *et al.* [2022], with details provided in Appendix B.3. We include patches of icy surfaces in the office world, with the task to either patrol in the corridor between "a" and "b", or write letters at "l" and then patrol between getting tea "t" and workplace "w", while never hitting obstacles "o". $K = 5$ is set for this task for a steeper increase in reward, since the long distance between patrolling states makes visiting many accepting states in each episode time consuming. Figure 2 (right) presents again the performance benefit of our methods, with CF+KC performing the best and CF and KC second and third, respectively. For this challenging task in a large environment, the method of Hahn *et al.* [2019] requires the highest number of training steps to converge.

Overall, the results demonstrate KC improves both sample efficiency and training stability, especially for challenging tasks. In addition, CF greatly improves sample efficiency, which, combined with KC, achieves the best results.

4.1 Runtime Analysis and Sensitivity Analysis

It is worth mentioning that, for counterfactual imagining, multiple updates to the Q function are performed at each step in the environment. This increases the computational complexity, but the additional inner loop updates on the Q function will only marginally affect the overall computation time if the environment steps are computationally expensive. Taking the office world task as an example, the average time to perform 6 million training steps are 170.9s, 206.3s and 236.1s for KC, CF and CF+KC, respectively. However, the time until convergence to the optimal policies are 96.8s, 70.2s and 27.5s for KC, CF and CF+KC, respectively.

For sensitivity analysis on the key parameters, we run experiments on the probabilistic gate MDP task with different parameters against the default values of $U = 0.1, \gamma = 0.99$ and $K = 10$. As shown in Figure 3, if $U(= 0.5)$ is chosen too high or $\gamma(= 0.9)$ is chosen too low, the algorithm does not converge to the optimal policy as expected. However, looser parameters $U = 0.01$ and $\gamma = 0.995$ do not harm the performance, which means that, even with limited knowledge of the underlying MDP, our algorithm still performs well with

loose parameters. Optimality is not affected by the K value, while the performance is only mildly affected by different K values.

5 Conclusion

We presented a novel model-free reinforcement learning algorithm to learn the optimal policy of satisfying LTL specifications in an unknown stochastic MDP with optimality guarantees. We proposed a novel product MDP, a generalized reward structure and a RL algorithm that ensures convergence to the optimal policy with the appropriate parameters. Furthermore, we incorporated counterfactual imagining, which exploits the LTL specification to create imagination experiences. Lastly, utilizing PRISM [Kwiatkowska *et al.*, 2011], we directly evaluated the performance of our methods and demonstrated superior performance on various MDP environments and LTL tasks.

Future works include exploring other specific reward functions under our generalized reward structure framework, tackling the open problem [Alur *et al.*, 2021] of dropping all assumptions regarding the underlying MDP, and extending the theoretical framework to continuous states and action environment MDPs, which might be addressed through an abstraction of the state space. In addition, utilizing potential-based reward shaping [Ng *et al.*, 1999][Devlin and Kudenko, 2012] to exploit the semantic class structures of LTL specifications and transferring similar temporal logic knowledge of the agent [Xu and Topcu, 2019] between environments could also be interesting.

References

[Aksaray *et al.*, 2016] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 6565–6570, 9 2016.

[Alur *et al.*, 2021] Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. A framework for transforming specifications in reinforcement learning. *arXiv:2111.00272*, 2021.

[Andreas *et al.*, 2016] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *34th International Conference on Machine Learning, ICML 2017*, 1:229–239, 11 2016.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles Of Model Checking*, volume 950. MIT Press, 2008.

[Bozkurt *et al.*, 2019] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 10349–10355, 2019.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba Openai. OpenAI Gym. *arXiv:1606.01540*, 6 2016.

[Cai *et al.*, 2021] Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics and Automation Letters*, 6(4):7973–7980, 2021.

[Camacho *et al.*, 2019] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-August:6065–6073, 2019.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2013.

[Devlin and Kudenko, 2012] Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[Giuseppe De Giacomo *et al.*, 2019] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, 2019.

[Hahn *et al.*, 2019] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 11427 LNCS:395–412, 2019.

[Hahn *et al.*, 2020] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives. *Automated Technology for Verification and Analysis*, 12302 LNCS:108–124, 2020.

[Hasanbeig *et al.*, 2019] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. *Proceedings of the IEEE Conference on Decision and Control*, 2019-December:5338–5343, 9 2019.

[Hasanbeig *et al.*, 2020] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. *Proceedings of*

the International Conference on Formal Modeling and Analysis of Timed Systems*, 12288 LNCS:1–22, 2020.

[Jiang *et al.*, 2021] Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7995–8003, 5 2021.

[Jothimurugan *et al.*, 2020] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 32, 8 2020.

[Jothimurugan *et al.*, 2021] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 12:10026–10039, 6 2021.

[Křetínský *et al.*, 2018] Jan Křetínský, Tobias Meggendorfer, Salomon Sickert, and Christopher Ziegler. Rabinizer 4: From LTL to your favourite deterministic automaton. *Proceedings of the International Conference on Computer Aided Verification*, 10981 LNCS:567–577, 2018.

[Kwiatkowska *et al.*, 2011] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. *Computer Aided Verification*, 6806 LNCS:585–591, 2011.

[Li *et al.*, 2016] Xiao Li, Cristian Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. *IEEE International Conference on Intelligent Robots and Systems*, 2017-September:3834–3839, 12 2016.

[Littman *et al.*, 2017] Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via GLTL. *arXiv:1704.04341*, 4 2017.

[Littman, 2001] M.L. Littman. Markov decision processes. *International Encyclopedia of the Social & Behavioral Sciences*, pages 9240–9242, 2001.

[Ng *et al.*, 1999] Andrew Y. Ng, Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

[Sadigh *et al.*, 2014] Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. *Proceedings of the IEEE Conference on Decision and Control*, 2015-February(February):1091–1096, 2014.

[Shao and Kwiatkowska, 2023] Daqian Shao and Marta Kwiatkowska. Sample efficient model-free reinforcement learning from ltl specifications with optimality guarantees. *arXiv:2305.01381*, 5 2023.

[Sickert *et al.*, 2016] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínský. Limit-deterministic Büchi

automata for linear temporal logic. *Computer Aided Verification*, 9780:312–332, 2016.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An Introduction*, volume 3. MIT Press, 2018.

[Toro Icarte *et al.*, 2018] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A Mcllraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[Toro Icarte *et al.*, 2022] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

[Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning 1992 8:3*, 8(3):279–292, 5 1992.

[Xu and Topcu, 2019] Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.