Accelerated Learning with Linear Temporal Logic using Differentiable Simulation

Anonymous Author(s)
Affiliation
Address
email

Abstract

To ensure learned controllers comply with safety and reliability requirements for reinforcement learning in real-world settings remains challenging. Traditional safety assurance approaches, such as state avoidance and constrained Markov decision processes, often inadequately capture trajectory requirements or may result in overly conservative behaviors. To address these limitations, recent studies advocate the use of formal specification languages such as linear temporal logic (LTL), enabling the derivation of correct-by-construction learning objectives from the specified requirements. However, the sparse rewards associated with LTL specifications make learning extremely difficult, whereas dense heuristic-based rewards risk compromising correctness. In this work, we propose the first method, to our knowledge, that integrates LTL with differentiable simulators, facilitating efficient gradient-based learning directly from LTL specifications by coupling with differentiable paradigms. Our approach introduces soft labeling to achieve differentiable rewards and states, effectively mitigating the sparse-reward issue intrinsic to LTL without compromising objective correctness. We validate the efficacy of our method through experiments, demonstrating significant improvements in both reward attainment and training time compared to the discrete methods.

1 Introduction

The growing demand for artificial intelligence (AI) systems to operate in a wide range of environments underscores the need for systems that can learn through interaction with their environments, without relying on human intervention. Reinforcement learning (RL) has emerged as a powerful tool for training controllers to perform effectively in uncertain settings with intricate, high-dimensional, and nonlinear dynamics. Recent advances in RL have enabled attainment of high-performance controllers in a variety of applications [1], such as robotic arm control [2], hand manipulation [3], legged locomotion [4, 5], navigation in crowded spaces [6], and robot-assisted surgery [7]. Despite the promising results in controlled environments, deploying learned controllers in real-world systems –where malfunctioning can be costly or hazardous– requires not only high performance but also strict compliance with formally specified safety and reliability requirements. Therefore, ensuring that learned controllers meet these critical specifications is essential to fully realize the potential of AI systems in real-world applications. Safety in learning is often modeled with constrained Markov decision processes (MDPs) [8–12], where the accumulated cost must be within a budget. However, additive cost functions may not reflect real-world safety, as assigning meaningful costs to harms is challenging. Alternative approaches define safety by avoiding unsafe states or actions [13–27], which is simpler than designing cost functions. However, this may result in overly conservative policies and could not capture complex trajectory-level requirements.

Recently, researchers have explored specifying RL objectives using formal languages, which explicitly and unambiguously express trajectory-based task requirements, including safety and liveness properties. Among these, linear temporal logic (LTL) has gained particular popularity [28–50] due the automaton-based memory it offers, which ensures history-independence and makes it especially suitable for long-horizon tasks unlike other languages such as signal temporal logic (STL). Specifying desired properties in LTL inherently prevents mismatches between the intended behavior and the behavior learned through reward maximization–one of the most well-known safety challenges in AI [51]. Although these methods are proven to define the correct RL objectives, the sparse logical

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

rewards make learning extremely difficult, as obtaining a nonzero reward often requires significant exploration. Denser LTL-based rewards provided through heuristics might accelerate learning [38]; however, if not carefully designed, they can compromise the correctness of the objective and misguide exploration depending on the environment, ultimately reducing learning efficiency. In this work, we address the challenges of scalable learning with correct objectives for long-horizon learning tasks. We adopt LTL as the specification language, leveraging the intuitive high level language and the automaton-based memory it provides. Unlike prior methods, our approach harnesses gradients from differentiable simulators to facilitate efficient learning directly from LTL specifications while preserving the correctness of the objectives. Our contributions can be summarized as follows:

- We propose, to the best of our knowledge, the first approach that accelerates learning from LTL specifications using differentiable simulators. Our approach effectively mitigates the inherent issue of the sparse rewards without sacrificing the expressiveness and correctness that LTL provides.
- We introduce soft labeling techniques for continuous environments that yield probabilistic ε-actions and transitions within the automata derived from LTL, which ensures the differentiability of rewards and states with respect to actions.
- Through a serious of experiments, we demonstrate that our approach enables successful learning from LTL specifications in robotic systems, whereas traditional non-differentiable approaches fail to achieve feasible learning.

2 Related Work

Safe RL. One common perspective in Safe RL defines safety as the guarantee on the cumulative costs over time within a specified safety budget, which is often modeled using constrained MDPs and has been widely studied [8–12], relying on additive cost functions and budgets, which may not adequately capture safety in many scenarios. In practice, it is often difficult to assign unambiguous scalar costs reflecting trade-offs between different harmful situations [52]. Another approach defines safety in terms of avoiding unsafe states and focuses on preventing or modifying unsafe actions via shielding or barrier functions [13–27], which only require identification of unsafe states and actions and often easier than designing cost functions [53]; however, they can lead to overly conservative control policies [54]. Moreover, the requirements are often placed over trajectories, which could be more complex than simply avoiding certain states [55]. Our approach avoids these issues by employing LTL as the specifications language to obtain correct-by-construction RL objectives.

RL with Temporal Logics. There is growing increasing in using formal specification languages to encode trajectory-dependent task objectives, especially those involving safety. LTL is widely used due to its expressiveness and well-defined semantics over infinite traces. There has been increasing interest in using formal specification languages to encode task objectives that are trajectory-dependent, particularly those involving safety requirements. LTL has emerged as a widely adopted formalism due to its expressiveness and well-defined semantics over infinite traces. Recent efforts [28–50] derives rewards from LTL specifications for RL, typically by translating LTL into limit-deterministic Büchi automata (LDBAs) and assigning rewards based on acceptance conditions. The memory structure provided by LDBAs supports long-horizon tasks better than alternatives such as STL, which require history-dependent rewards [56]. However, LTL rewards are often sparse and hinder learning. While heuristic-based dense rewards [38] attempt to address this, they risk misguiding exploration and compromising correctness. Our approach avoids these pitfalls by leveraging gradients from differentiable simulators to accelerate learning without sacrificing correctness.

RL with Differentiable Simulators. Differentiable simulators enable gradient-based policy optimization in RL by computing gradients of states and rewards with respect to actions, using analytic methods [57–61] or auto-differentiation [62, 63]. While Backpropagation Through Time (BPTT) is commonly used [64–69], it suffers from vanishing or exploding gradients in long-horizon tasks as it ignores the Markov property of states [70]. To address this, several differentiable RL algorithms have been proposed [71, 72]. Short Horizon Actor-Critic (SHAC) [73] divides long trajectories into shorter segments where BPTT is tractable and bootstraps the remaining trajectory using the value function. Adaptive Horizon Actor-Critic (AHAC) [74] extends SHAC by dynamically adjusting the segment lengths based on contact information from the simulator. Gradient-Informed PPO [75] incorporates gradient information into the PPO framework in an adaptive manner. Our approach builds a differentiable, Markovian transition function for LTL-derived automata, making it compatible with all differentiable RL methods. Unlike prior STL-based efforts [76, 77], which rely on non-Markovian rewards and BPTT, our method supports efficient long-horizon learning with full differentiability.

3 Preliminaries and Problem Formulation

MDPs. We formalize the interaction between controllers with the environments as MDPs, which can be used for a wide range of robotic systems, including arm manipulation and legged locomotion.

Definition 1. *A (differentiable) MDP is a tuple $M = (S, A, f, p_0)$ such that S is a set of continuous states; A is a set of continuous actions; $f : S \times A \mapsto S$ is a differentiable transition function; p_0 is an initial state distribution where $p_0(s)$ denotes the probability density for the state s.*

For a given robotic task, the state space S can be defined by the positions $\mathtt{x}$ and velocities $\dot{\mathtt{x}}$ of relevant objects, body parts, and joints. The action space A may consist of torques applied to the joints. The transition function f captures the underlying system dynamics and outputs the next state via computing the accelerations $\ddot{\mathtt{x}}$ by solving $\mathtt{M}\ddot{\mathtt{x}} = \mathtt{J}^{\mathtt{T}}\mathtt{F}(\mathtt{x}, \dot{\mathtt{x}}) + \mathtt{C}(\mathtt{x}, \dot{\mathtt{x}}) + \mathtt{T}(\mathtt{x}, \dot{\mathtt{x}}, a)$, for a given state $s = \langle \mathtt{x}, \dot{\mathtt{x}} \rangle \in S$ and action $a \in A$. Here, $\mathtt{F}$, $\mathtt{C}$, and $\mathtt{T}$ are, respectively, force, Coriolis, and torque functions that can be approximated using differentiable physics simulators.

RL Objective. In RL, a given policy $\pi : S \mapsto A$ is evaluated based on the expected cumulative reward (known as return) associated with the paths $\sigma := s_0 s_1 \dots$ (sequence of visited states) generated by the Markov chain (MC) M_π induced by the policy π. Specifically, for given a reward function $R : S \mapsto \mathbb{R}$, a discount factor $\gamma \in (0, 1)$, and a horizon H, the return of a path σ from time $t \in \mathbb{N}$ is defined as $G_{t:H}(\sigma) = \sum_{i=t}^{H} \gamma^i R(\sigma[i])$. For simplicity, we denote the infinite-horizon return starting from $t = 0$ as $G_H(\sigma) := G_{0:H}(\sigma)$, and further drop the subscript to write $G(\sigma) := \lim_{H \to \infty} G_H(\sigma)$. The discount factor γ reduces the value of future rewards to prioritize immediate ones: a reward received after t steps, $R(\sigma[t])$, contributes $\gamma^t R(\sigma[t])$ to the return. The objective in RL is to learn a policy that maximizes the expected return over trajectories.

Labels. In robotic environments, we define the set of atomic propositions (APs), denoted by $\mathtt{A}$, as properties of interest that place bounds on functions of the state space. Formally, each AP takes the form $\mathtt{a}$:='$g(s)>0$', where $g : S \mapsto \mathbb{R}$ is assumed to be a differentiable function mapping a given state to a signal. For example, the function $g(\langle \mathtt{x}, \dot{\mathtt{x}} \rangle) := \dot{\mathtt{x}}_{\max}^2 - \dot{\mathtt{x}}_i^2$ can be used to define an AP that specifies that the velocity of the i-th robotic component must be below an upper bound $\dot{\mathtt{x}}_{\max}$. The labeling function $L : S \mapsto 2^{\mathtt{A}}$ returns the set of APs that hold true for a given state. Specifically, an AP $\mathtt{a} :=$ '$g(s) > 0$' is included in the label set $L(s)$ of state s – i.e., s is labeled by $\mathtt{a}$ if and only if (iff) $g(s) > 0$. We also write, with a slight abuse of notation, $L(\sigma) := L(\sigma[0])L(\sigma[1]) \dots$ to denote the trace (sequences of labels) of a path σ. Finally, we write $M^+{=}(M, L)$ to denote a labeled MDP.

LTL. LTL provides a high-level formal language for specifying the desired temporal behaviors of robotic systems. Alongside the standard operators in propositional logic – negation ($\neg$) and conjunction ($\wedge$) – LTL offers two temporal operators, namely next ($\bigcirc$) and until (U). The formal syntax of LTL is defined by the following grammar ([78]): $\varphi := \mathrm{true} \mid \mathtt{a} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2,\ \mathtt{a} \in \mathtt{A}$. The semantics of LTL formulas are defined over paths. Specifically, a path σ either satisfies φ, denoted by $\sigma \models \varphi$, or not ($\sigma \not\models \varphi$). The satisfaction relation is defined recursively as follows: $\sigma \models \varphi$; if $\varphi = \mathtt{a}$ and $\mathtt{a} \in L(\sigma[0])$ (i.e., $\mathtt{a}$ immediately holds); if $\varphi = \neg\varphi'$ and $\sigma \not\models \varphi'$; if $\varphi = \varphi_1 \wedge \varphi_2$ and $(\sigma \models \varphi_1) \wedge (\sigma \models \varphi_2)$; if $\varphi = \varphi_1 \mathsf{U} \varphi_2$ and there exists $t \geq 0$ such that $\sigma[t{:}] \models \varphi_2$ and for all $0 \leq i < t$, $\sigma[i{:}] \models \varphi_1$. The remaining Boolean and temporal operators can be derived via the standard equivalences such as eventually ($\Diamond\varphi := \mathrm{true}\ \mathsf{U}\ \varphi$) and always ($\Box\varphi := \neg(\Diamond\neg\varphi)$).

LDBAs. Whether a path σ satisfies a given LTL formula φ can be automated by building a corresponding LDBA, denoted by $\mathcal{A}_\varphi$ that is suitable for quantitative model-checking of MDPs ([79]). An LDBA is a tuple $\mathcal{A}_\varphi = (Q, q_0, \Sigma, \delta, B)$ where Q is a finite set of states; $q_0 \in Q$ is the initial state; $\Sigma{=}2^{\mathtt{A}}$ is the set of labels; $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$ is a transition function triggered by labels; $B \subseteq Q$ is the accepting states. An LDBA $\mathcal{A}_\varphi$ accepts a path σ (i.e., $\sigma \models \varphi$), iff its trace $L(\sigma)$ induces an LDBA execution visiting some of the accepting states infinitely often, known as the Büchi condition.

Control Synthesis Problem. Our objective is to learn control policies that ensure given path specifications are satisfied by a given labeled MDP. In stochastic environments, this objective translates to maximizing the probability of satisfying those specifications. We consider specifications given as LTL formulas since LTL provides a high-level formalism well-suited for expressing safety and other temporal constraints in robotic systems–and, importantly, finite-memory policies suffice to satisfy LTL specifications [80]. We now formalize the control synthesis problem as follows:

Problem 1. *Given a labeled MDP M^+ and a LTL formula φ, find an optimal finite-memory policy π^*_φ that maximizes the probability of satisfying φ, i.e., $\pi^*_\varphi := \underset{\pi \in \Pi}{\mathrm{argmax}} \Pr_{\sigma \sim M^+_\pi}\{\sigma \mid \sigma \models \varphi\}$, where Π is the set of policies and σ is a path drawn from the Markov chain (MC) M^+_π induced by π.*

4 Accelerated Learning from LTL using Differentiable Rewards

In this section, we present our approach for efficiently learning optimal policies that satisfy given LTL specifications by leveraging differentiable simulators. We first define product MDPs and discuss their conventional use in generating discrete LTL-based rewards for reinforcement learning. We then introduce our method for deriving differentiable rewards using soft labeling, enabling gradient-based optimization while preserving the logical structure of the specifications.

Product MDPs. A product MDP is constructed by augmenting the states and actions of the original MDP with indicator vectors representing the LDBA states. The state augmentations serve as memory modes necessary for tracking temporal progress, while the action augmentations, referred to as ε-actions, capture the nondeterministic ε-moves of the LDBA. The transition function of the product MDP reflects a synchronous execution of the LDBA and the MDP; i.e., upon taking an action, the MDP moves to a new state according to its transition probabilities, and the LDBA transitions by consuming the label of the current MDP state.

Definition 2. *A product MDP $\mathbf{M} = (\mathbf{S}, \mathbf{A}, \mathbf{f}, \mathbf{p_0}, \mathbf{B})$ is of a labeled MDP $M^+ = (S, A, f, p_0, \mathtt{A}, L)$ with an LDBA $\mathcal{A}_\varphi = (Q, \Sigma{=}2^{\mathtt{A}}, \delta, q_0, B)$ derived from a given LTL formula φ such that $\mathbf{S} = S \times \mathbf{Q}$ is the set of product states and $\mathbf{A} = A \times \mathbf{Q}$ is the set of product actions where $\mathbf{Q} = [0,1]^{|Q|}$ is the space set for the one-hot indicator vectors of automaton states; $\mathbf{f} : \mathbf{S} \times \mathbf{A} \mapsto \mathbf{S}$ is the transition function defined as*

$$\mathbf{f}(\langle s, \mathbf{q}^q \rangle, \langle a, \mathbf{q}^{q_\varepsilon} \rangle) \coloneqq \begin{cases} \langle s', \mathbf{q}^{q'} \rangle & q_\varepsilon \notin \delta(q', \varepsilon) \\ \langle s', \mathbf{q}^{q_\varepsilon} \rangle & q_\varepsilon \in \delta(q', \varepsilon) \end{cases} \tag{1}$$

for given $s, s', \in S, a \in A$ and the indicator vectors $\mathbf{q}^q, \mathbf{q}^{q'}, \mathbf{q}^{q_\varepsilon} \in \mathbf{Q}$ for $q, q', q_\varepsilon \in Q$, respectively, where $s' \coloneqq f(s,a)$ and $q' \coloneqq \delta(q, L(s))$; $\mathbf{p_0}$ is the initial product state distribution where $p_0^\times(\langle s, \mathbf{q}^q \rangle)[q = q_0]$; $\mathbf{B} = \{\langle s, \mathbf{q}^q \rangle \in \mathbf{S} \mid q \in B\}$ is the set accepting product states. A product MDP is said to accept a product path $\boldsymbol{\sigma}$ iff $\boldsymbol{\sigma}$ satisfies the Büchi condition, denoted as $\boldsymbol{\sigma} \models \square\lozenge\mathbf{B}$, which is to visit some states in $\mathbf{B}$ infinitely often.

By definition, any product path accepted by the product MDP corresponds to a path in the original MDP that satisfies the Büchi acceptance condition of the LDBA. Consequently, the satisfaction of the LTL specification φ is reduced to ensuring acceptance in the product MDP. This reduces Problem 1 to maximizing the probability of reaching accepting states infinitely often in the product MDP:

Lemma 1 (from Theorem 3 in [79])**.** *A memoryless product policy $\boldsymbol{\pi}^*_\varphi$ that maximizes the probability of satisfying the Büchi condition in a product MDP $\mathbf{M}$ constructed from a given labeled MDP M^+ and the LDBA $\mathcal{A}_\varphi$ derived from a given LTL specification φ, induces a policy π^*_φ with a finite-memory captured by $\mathcal{A}_\varphi$ maximizing the satisfaction probability of φ in M^+.*

Discrete LTL Rewards. The idea is to derive LTL rewards from the acceptance condition of the product MDP to be able train control policies via RL approaches. Specifically, we consider the approach proposed in [33] that uses carefully crafted rewards and state-dependent discounting based on the Büchi condition such that an optimal policy maximizing the expected return is also an objective policy $\boldsymbol{\pi}^*_\varphi$ maximizing the satisfaction probabilities as defined in Lemma 1, as formalized below:

Theorem 1. *For a given product MDP $\mathbf{M}$, the expected return for a policy $\boldsymbol{\pi}$ approaches the probability of satisfying the Büchi acceptance condition as the discount factor γ goes to 1; i.e., $\lim_{\gamma \to 1^-} \mathbb{E}_{\boldsymbol{\sigma} \sim \mathbf{M}_{\boldsymbol{\pi}}}[G(\boldsymbol{\sigma})] = Pr_{\boldsymbol{\sigma} \sim \mathbf{M}_{\boldsymbol{\pi}}}(\boldsymbol{\sigma} \models \square\lozenge\mathbf{B})$; if the return $G(\boldsymbol{\sigma})$ is defined as follows:*

$$G(\boldsymbol{\sigma}) \coloneqq \sum_{t=0}^{\infty} R(\boldsymbol{\sigma}[t]) \prod_{i=0}^{t-1} \Gamma(\boldsymbol{\sigma}[i]), \quad R(\mathbf{s}) \coloneqq \begin{cases} 1-\beta & \mathbf{s} \in \mathbf{B} \\ 0 & \mathbf{s} \notin \mathbf{B} \end{cases}, \quad \Gamma(\mathbf{s}) \coloneqq \begin{cases} \beta & \mathbf{s} \in \mathbf{B} \\ \gamma & \mathbf{s} \notin \mathbf{B} \end{cases} \tag{2}$$

where $\prod_{i=0}^{-1} \coloneqq 1$, β is a function of γ satisfying $\lim_{\gamma \to 1^-} \frac{1-\gamma}{1-\beta} = 0$, $R : \mathbf{S} \mapsto [0,1)$ and $\Gamma : \mathbf{S} \mapsto (0,1)$ are state-dependent reward and the discount functions respectively.

The proof can be found in [33]. The idea is to encourage the agent to repeatedly visit an accepting state as many times as possible by assigning a larger reward to the accepting states. Further, the rewards are discounted less in non-accepting states to reflect that the number of visitations to non-accepting states are not important. The LTL rewards provided this approach is that the rewards are

very sparse; depending on the environment and the structure of the automaton, the agent might need to blindly explore a large portion of the state space before getting a nonzero reward, which constitutes the main hurdle in learning from LTL specifications.

Differentiable LTL Rewards. We propose employing differentiable reinforcement learning (RL) algorithms and simulators to mitigate the sparsity issue and accelerate learning. However, the standard LTL rewards described earlier are not only sparse but discrete, rendering them non-differentiable with respect to states and actions. This lack of differentiability primarily stems from two factors: the binary state-based reward function and discrete automaton transitions. To address this challenge, we introduce probabilistic "soft" labels. We start by defining the probability that a given AP, denoted as $\mathtt{a} \coloneqq `g(s) > 0'$, belongs to the label $L(s)$ of a state s. Formally, we define this probability as:

$$\Pr(\mathtt{a} \in L(s)) = \Pr(g(s) > 0) \coloneqq h(g(s)) = \frac{1}{1 + \exp(-g(s))}. \tag{3}$$

Although we use the widely adopted sigmoid function here[1], any differentiable cumulative distribution function (CDF) $h : \mathbb{R} \mapsto [0, 1]$ could be applied. Building upon these probabilities, we define the probability associated with a label l as follows:

$$\Pr(L(s) = l) = \prod_{\mathtt{a} \in l} \Pr(\mathtt{a} \in L(s)) \prod_{\mathtt{a} \notin l} (1 - \Pr(\mathtt{a} \in L(s))). \tag{4}$$

These probabilistic labels induce probabilistic automaton transitions, causing the controller to observe automaton states probabilistically. Consequently, instead of modeling automaton states as deterministic indicator vectors in product MDPs, we represent them as probabilistic superpositions over all possible automaton states. By doing so, we design differentiable transitions and rewards within the product MDP. Let $f_L : S \times \mathbf{Q} \mapsto \mathbf{Q}$ denote the function that updates the automaton state probabilities based on the LDBA transitions triggered by probabilistic labels, and let $\mathbf{q}$ denote the vector where each element $\mathbf{q}_q$ is the probability of being in automaton state q, then we can formally define:

$$f_L(\langle s, \mathbf{q} \rangle) = \mathbf{q}' \text{ where } \mathbf{q}'_{q'} = \sum_q \mathbf{q}_q \sum_{l \in L_{q,q'}} \Pr(L(s) = l) \text{ and } L_{q,q'} \coloneqq \{l \mid q' = \delta(q, l)\}. \tag{5}$$

Intuitively, the probability of transitioning to a subsequent automaton state q' is computed by summing probabilities across all current automaton states q and labels $l \in L_{q,q'}$ capable of leading to state q'. This computation can be efficiently done through differentiable matrix multiplication.

The remaining hurdle is the binary ε-actions available to the controller, which trigger ε-transitions in the LDBA. Similarly to the soft labels approach, ε-actions can become differentiable by representing the probabilities of the ε-transitions to be triggered. Let $f_\varepsilon : \mathbf{Q} \times \mathbf{Q} \mapsto \mathbf{Q}$ denote the function updating automaton state probabilities based on the ε-action taken, and let $\mathbf{q}^\varepsilon$ denote the vector whose elements indicate the probabilities of taking the ε-actions leading to the corresponding automaton states, we then define:

$$f_\varepsilon(\mathbf{q}, \mathbf{q}^\varepsilon) = \mathbf{q}' \text{ where } \mathbf{q}'_{q'} = \sum_{q \in Q_{\varepsilon,q'}} \mathbf{q}_q \mathbf{q}^\varepsilon_{q'} + \sum_{q \in \overline{Q}_{q',\varepsilon}} \mathbf{q}_{q'} \mathbf{q}^\varepsilon_q,\ Q_{\varepsilon,q'} \coloneqq \{q \mid q' \in \delta(q, \varepsilon)\},\ \overline{Q}_{q',\varepsilon} \coloneqq \{q \mid q \notin \delta(q', \varepsilon)\}. \tag{6}$$

Conceptually, the probability of transitioning to automaton state q' involves two scenarios: (the first summation in (6)) the probability of moving to q' via valid ε-transitions, and (the second summation in (6)) the probability of remaining in q' after trying to leave from q' via nonexistent ε-transitions. These vector computations can be efficiently performed in a differentiable manner.

We can formulate the complete transition function $\mathfrak{f}$ by composing f_L, f_ε, and f as follows:

$$\mathfrak{f}(\langle s, \mathbf{q} \rangle, \langle a, \mathbf{q}^\varepsilon \rangle) \coloneqq \langle f(s, a), f_L(\langle s, f_\varepsilon(\mathbf{q}, \mathbf{q}^\varepsilon) \rangle) \rangle. \tag{7}$$

This transition function first executes the ε-actions, then performs the LDBA transitions triggered by state labels to update the automaton state probabilities, while applying the given action to update the MDP states. The function $\mathfrak{f}$ is fully differentiable with respect to s, $\mathbf{q}$, a, and $\mathbf{q}^\varepsilon$. We can now obtain

[1]For the correctness of LTL, $\Pr(g(s) > 0)$ must be exactly 0 or 1 for values below or above certain thresholds. In practice, this is not an issue, as overflow behavior of sigmoid ensures this condition is satisfied

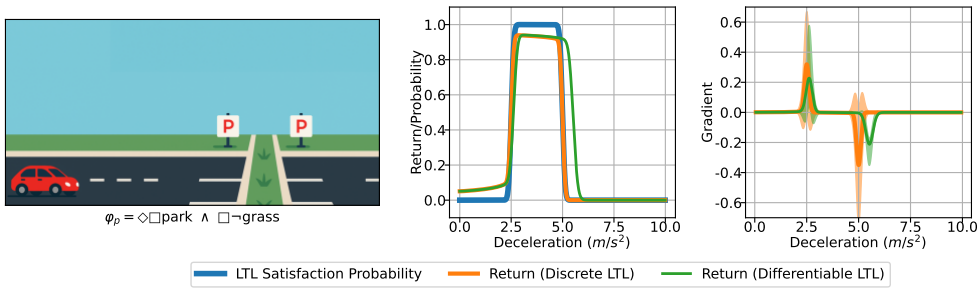

Figure 1: **LTL Returns and Derivatives.** *Left*: The parking scenario where the car must brake to stop in the parking area without entering the grass field (φ_p). *Middle*: LTL satisfaction probability and return estimates from discrete and differentiable LTL formulations as functions of deceleration. *Right*: LTL return gradients with respect to deceleration and their standard deviation. The key challenge in learning from LTL arises from slightly-sloped regions and sharp changes in the returns produced by discrete LTL rewards. Our *differentiable LTL* approach not only *smooths these abrupt changes but also enables the use of low-variance first-order gradient estimates essential for effective learning in slightly-sloped regions.*

a reward $\mathfrak{R} : \mathbf{Q} \mapsto (0, 1)$ and a discounting function $\mathfrak{D} : \mathbf{Q} \mapsto (0, 1)$ that are also differentiable with respect state and actions as follows:

$$\mathfrak{R}(\langle s, \mathbf{q} \rangle) \coloneqq (1 - \beta) \sum_{q \in B} \mathbf{q}_q, \quad \mathfrak{D}(\mathbf{q}) \coloneqq \beta \sum_{q \in B} \mathbf{q}_q + \gamma \sum_{q \notin B} \mathbf{q}_q \tag{8}$$

These differentiable reward, discounting and functions allow us to obtain first-order gradient estimates $\nabla^1_\psi J(\psi) \coloneqq \mathbb{E}_{\sigma \sim M_{\pi_\psi}} [\nabla_\psi G_H(\sigma)]$ which are known to exhibit lower variance compared to zeroth-order estimates [73]. Such first-order estimates can be effectively utilized by differentiable RL algorithms to accelerate learning. In the following example, we illustrate employing these lower-variance gradient estimates is particularly crucial when learning from LTL rewards.

Parking Example. Consider a parking scenario in which the vehicle starts with an initial velocity of $v_0 = 10$ m/s. The controller applies the brakes with a constant deceleration $a \in [0 \text{ m/s}^2, 10 \text{ m/s}^2]$ over the next 10 seconds, with the goal of bringing the car to rest inside the parking area. For safety, the vehicle must not enter the grass field before reaching the parking zone on the right-hand side. We formalize these requirements in LTL as $\varphi_p = \Diamond\Box\texttt{park} \wedge \Box\neg\texttt{grass}$ where the parking area and the grass field are defined as $\texttt{park} \coloneqq (x > 10 \text{ m} \wedge x < 20 \text{ m}) \vee (x > 30 \text{ m} \wedge x < 40 \text{ m})$ and $\texttt{grass} \coloneqq x > 20 \text{ m} \wedge x < 30 \text{ m}$, respectively.

Figure 1 illustrates this task, including satisfaction probabilities, returns, and gradients with respect to deceleration. The satisfaction probability is 1 for deceleration values between 2.5 m/s^2 and 5.0 m/s^2, and 0 outside this range. The differentiable LTL returns closely match the discrete ones, except near the boundaries of the satisfaction region, where the differentiable version produces smoother transitions. This smoothness is particularly evident in the gradient plots. Although differentiable LTL rewards yield smoother return curves, learning remains challenging due to the small gradient magnitudes across most of the parameter space except near the satisfaction boundaries. For instance, in the region between 0.0 m/s^2 and 2.5 m/s^2, the returns increase with deceleration, but noisy gradient estimates can still lead the learner away from the satisfaction region. Therefore, obtaining low-variance gradient estimates is especially beneficial when learning from LTL, where most of the landscape requires sharper gradients for effective optimization.

5 Experiments

In this section, through simulated experiments, we show learning from the differentiable LTL rewards provided by our approach is significantly faster than learning from standard discrete LTL rewards.

Implementation Details. We implemented our approach in Python utilizing the PyTorch-based differentiable physics simulator dFlex introduced in [73]. We used an NVIDIA GeForce RTX 2080 GPU, 4 Intel(R) Xeon(R) Gold 5218 CPU cores, and 32 gigabytes memory for each experiment.

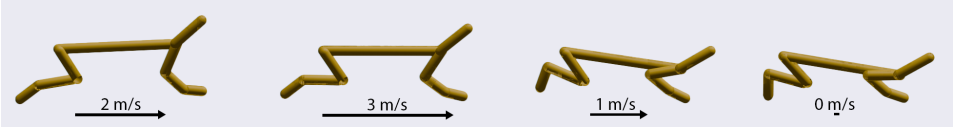

Figure 2: **Task Specification with LTL.** This figure illustrates a Cheetah policy learned by SHAC using differentiable rewards derived via our approach from the LTL formula φ_{legged} (10), which specifies accelerating forward, stopping, and maintaining a safe tip-to-ground distance. Specifying the desired behaviors of robots using the high-level language LTL provides is an intuitive alternative to manually designing reward functions, which often require extensive domain expertise and risk unintended behaviors. Enabling learning directly from LTL unlocks new possibilities for robust, safe, and flexible robotic applications.

Specifically, we generate the automaton description using Owl [81] and parse it using Spot [82]. We then construct reward and transition tensors from the automata. We then compute the probabilities for each observations as explained in the previous section using a sequence of differentiable vector operations using PyTorch. Lastly, using the constructed transition and reward tensors, we update the automaton states and provide rewards. The overall approach is summarized in Algorithm 1.

Baselines. We use two widely adopted and representative state-of-the-art (SOTA) model-free RL algorithms as our baseline non-differentiable RL methods ($\not\partial$RLs): the on-policy Proximal Policy Optimization (PPO) [83] and the off-policy Soft Actor-Critic (SAC) [84]. For differentiable RL baselines (∂RLs), we employ SHAC and AHAC, which, to the best of our knowledge, represent the SOTA in this category. For each environment and baseline, we adopted the tuned hyperparameters in [74].

Algorithm 1 Differentiable RL with LTL

Require: MDP M, LTL formula φ, Policy π_ψ
  Derive LDBA A_φ and APs $\mathtt{A}$ from φ
  Derive $\mathfrak{f}$ (7) and $\mathfrak{R}$, $\mathfrak{D}$ (8) from A_φ
  while True **do**
    Initialize $\mathbf{q}^{(0)} \sim A_\varphi$, $s^{(0)} \sim M$, $G \leftarrow 0$
    for $t = 1, 2, \ldots, H$ **do**
      Get action $\langle a, \mathbf{q}^\varepsilon \rangle \sim \pi_\psi(\langle s^{(t\text{-}1)}, \mathbf{q}^{(t\text{-}1)} \rangle)$
      Execute ε-action $\mathbf{q}' \leftarrow f_\varepsilon(\mathbf{q}, \mathbf{q}^\varepsilon)$
      Execute label transition $\mathbf{q}^{(t)} \leftarrow f_L(\langle s, \mathbf{q}' \rangle)$
      Execute MDP action $s^{(t)} \leftarrow f(s, a)$
      Compute reward $r \leftarrow \mathfrak{R}(\mathbf{q}^{(\mathbf{t})})$
      Update return $G \leftarrow G + \mathfrak{D}(\mathbf{q}) \cdot r$
    end for
    Train π_ψ using differentiable return G
  end while

Metric. We evaluate performance in terms of the collected LTL rewards averaged over 5 seeds since they can serve as proxies for satisfaction probabilities. We considered two criteria: (1) the maximum return achieved and (2) the speed of convergence. To maintain consistency, we used differentiable LTL rewards across all baselines as, for non-differentiable baselines, we observed no performance difference between the differentiable and discrete LTL rewards.

CartPole. The CartPole environment consists of a cart that moves along a one-dimensional track, with a pole hinged to its top that can be freely rotated by applying torque. The system yields a 5-dimensional observation space and a 1-dimensional action space. The control objective is to move the tip of the pole through a sequence of target positions while maintaining the cart within a desired region as much as possible and ensuring the velocity of the cart always remains within safe boundaries. We capture these requirements in LTL as follows:

$$\varphi_{\text{cartpole}} = \underbrace{\square\,\text{`}|\texttt{cart\_vx}|<v_0\text{'}}_{\text{safety}} \wedge \underbrace{\square\lozenge\,\text{`}|\texttt{cart\_x}|<x_0\text{'}}_{\text{repetition}} \wedge \underbrace{\lozenge\big(\text{`}|\texttt{pole\_z}\text{-}z_0|<\Delta\text{'} \wedge \lozenge\,\text{`}|\texttt{pole\_z}\text{-}z_1|<\Delta\text{'}\big)}_{\text{reachability \& sequencing}}. \quad (9)$$

Here, `cart_x`, `cart_vx`, and `pole_`z represent the cart position, the cart velocity, and the pole height respectively. This formula demonstrates how LTL can be leveraged to encode both complex safety constraints and performance objectives. Specifically, we set $x_0 = 10$ m, $v_0 = 10$ m/s as boundaries, $z_0 = -1$ m, $z_1 = 1$ m as the target positions, and $\Delta = 25$ cm as the allowable deviation.

Legged Robots. We consider three legged-robot environments: Hopper, Cheetah, and Ant. The Hopper environment features a one-legged robot with 4 components and 3 joints, resulting in a 10-dimensional state space and a 3-dimensional action space. The Cheetah environment consists of a two-legged robot with 8 components and 6 joints, yielding a 17-dimensional state space and a 6-dimensional action space. The Ant environment includes a four-legged robot with 9 components and 8 joints, producing a 37-dimensional state space and an 8-dimensional action space. In all three environments, the control task requires always keeping the torso/tip of the robot above a critical safety height, maintaining a certain distance between the torso/tip and the critical height as often as

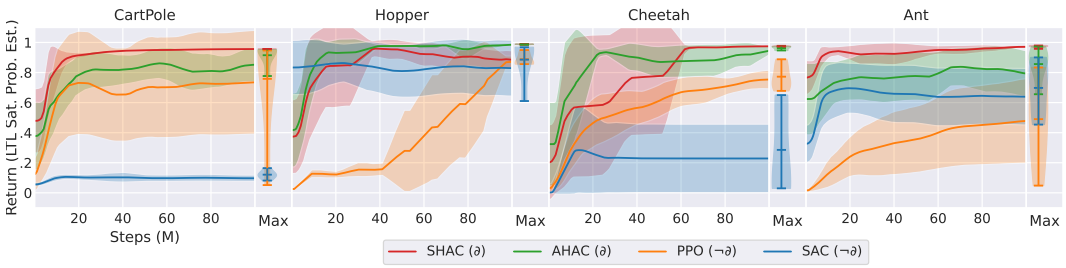

Figure 3: **Comparison Across Environments: Differentiable vs. Discrete LTL Rewards.** The wider plots show the learning curves of all baseline algorithms, while the narrower plots on the right display the maximum returns achieved after 100 M steps. All results are averaged over 5 random seeds, and the curves are smoothed using max and uniform filters for visual clarity. The reported returns, bounded between 0 and 1, serve as proxies for the probability of satisfying the LTL specifications. In all the environments algorithms utilizing **differentiable** LTL rewards (SHAC, AHAC) rapidly learn near-optimal policies, whereas those relying on discrete LTL rewards (PPO, SAC), display high variance, converge slowly, or getting stuck with sub-optimal/near-zero-return policies.

possible, and accelerating the robot forward, and then bringing the robot to a full stop. We formalize this task in LTL as follows:

$$\varphi_{\text{legged}} = \underbrace{\square\,\text{`torso\_z}>z_0\text{'}}_{\text{safety}} \wedge \underbrace{\square\lozenge\,\text{`torso\_z}>z_1\text{'}}_{\text{repetition}} \wedge \underbrace{\lozenge\big(\text{`torso\_vx}>v_1\text{'} \wedge \lozenge\,\text{`torso\_vx}<v_0\text{'}\big)}_{\text{reachability \& sequencing}}. \tag{10}$$

Here, `torso_z` and `torso_vx` denote the height and horizontal velocity of the robots. This formula captures several key aspects of LTL, including, safety, reachability, sequencing, and repetition. The values of z_0 and z_1 were chosen based on the torso height of each robot in their referential system. Specifically, we used $z_0 = -110$ cm, $z_1 = -105$ cm for Hopper; $z_0 = -75$ cm, $z_1 = -105$ cm for Cheetah; and $z_0 = 0$ cm, $z_1 = 5$ cm for Ant, where z_0 denotes the critical safety height and z_1 represents a safe margin above it. We set $v_1 = 1$ m/s, $v_1 = 3$ m/s, and $v_1 = 1.5$ m/s for Hopper, Cheetah, and Ant, respectively, reflecting movement speeds relatively challenging yet achievable for each of the robot. For deceleration, we set $v_0 = 0$ m/s for all the environments. An illustration of a policy learned from this specification for Cheetah is provided in Figure 2.

Results. Figure 3 presents our simulation results. Across all environments, ∂RL algorithms that leverage our differentiable LTL rewards consistently outperform $\not\partial$RL algorithms in terms of both maximum return achieved and learning speed from the LTL specifications.

CartPole. The safety specification induces an automaton with three states, each having 64 transitions–but only one of these transitions yields a reward. This extreme sparsity, even in a low-dimensional state space, severely hinders the learning process for $\not\partial$RLs, as shown in the leftmost plot of Figure 3. In contrast, ∂RL algorithms leverage the gradients provided by differentiable rewards, enabling them to efficiently learn policies that nearly satisfy the LTL specification. Specifically, ∂RLs converge to near-optimal policies (Pr>0.8) within just 20 M steps, whereas $\not\partial$RLs (SAC: all seeds; PPO: one seed) fail to learn any policy that achieves meaningful reward, even after 100 million (M) steps.

Legged Robots. As we move to environments with higher-dimensional state spaces–10, 17, and 37 dimensions for Hopper, Cheetah, and Ant, respectively–even relatively simple LTL specifications pose a significant challenge for $\not\partial$RLs. The automata derived from the LTL specifications in these environments consists of four states, each with 16 transitions, of which four transitions in the third state yield rewards. Reaching this state, however, requires extensive blind exploration of the state space, making it significantly hard for $\not\partial$RLs to learn optimal control policies. On the other hand, ∂RLs, guided by LTL reward gradients, quickly identify high-reward regions of the state space and learn effective policies.

For Hopper, ∂RLs converge to near-optimal policies (Pr>0.8) within 20 M steps, while PPO requires the full 100 M steps to converge, and one SAC seed gets trapped in a local optimum. For Cheetah, ∂RLs attain optimal performance (Pr>0.9), whereas PPO converges to a suboptimal policy even after 100 M steps, and SAC consistently fails by getting stuck in poor local optima. For Ant, ∂RLs again learn near-optimal policies rapidly, while $\not\partial$RLs converge only to suboptimal policies.

Ablation Study. To isolate the impact of differentiability of LTL rewards from inherent environment properties, we conduct an ablation study comparing ∂RLs and $\not\partial$RLs under simplified LTL specifications. Specifically, we use reduced versions of the LTL formulas from our earlier experiments:

$$\varphi'_{\text{cartpole}} := \Diamond \text{`}|\texttt{pole\_z}\text{-}z_0| < \Delta\text{'} \tag{11}$$

$$\varphi'_{\text{legged}} := \Diamond \text{`}\texttt{torso\_vx} > v_1\text{'} \tag{12}$$

using $z_0 = -1$ m, $\Delta = 25$ cm for Cartpole, and $v_1 = 50$ cm/s for all the legged-robot environments. These simplified formulas yield one-state automata with 4 and 2 transitions, respectively, of which one is accepting. As such, they lack the complexity that makes learning from LTL challenging. Figure 4 presents the maximum returns obtained for these simplified specifications. Each of the baselines, regardless of differentiability, learns an optimal policy (Pr$>$0.9) for all the environments. However, when comparing these results to those in Figure 3, we observe only a minor performance drop for ∂RLs, whereas the performance of $\not\partial$RLs degrades dramatically—for some cases, from near satisfaction to complete failure—as LTL complexity increases. These results support our hypothesis that the performance advantage of ∂RLs over $\not\partial$RLs arises primarily from leveraging the differentiability of LTL rewards provided by our approach, rather than from environment-specific properties.

6 Discussion and Limitations

Our approach accelerates learning from LTL specifications by leveraging differentiable RL algorithms that utilize gradients provided by differentiable simulators. Therefore, the overall performance of our method is inherently influenced by the quality and efficiency of the underlying simulators and RL algorithms. For example, if a simulator provides poor gradient information or computes gradients slowly, the learning process will be significantly slowed down. Another issue is the reliance on hyperparameters. Although we adopt tuned hyperparameters from existing work, applying our approach to new environments may require additional hyperparameter tuning. A further challenge lies in the formalization of LTL specifications. While LTL offers a more intuitive and structured way to specify tasks compared to manual reward engineering, it still requires familiarity with formal logic and sufficient domain knowledge to define meaningful bounds. Finally, our method introduces an additional hyperparameter: the CDF used for probability estimation, which must also be tuned for optimal performance.

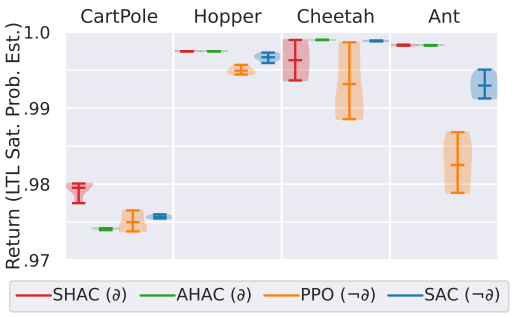

Figure 4: **Ablation Study for LTL.** The maximum returns obtained after 100 M steps for simplified LTL formulas (12), averaged over 5 seeds. Returns (0 to 1) indicate LTL satisfaction probabilities. Under these simpler specifications, both $\not\partial$RLs and ∂RLs successfully learn near-optimal policies. However, as shown in Figure 3, the *performance of discrete $\not\partial$RLs degrades dramatically with increasing LTL complexity—unlike differentiable ∂RLs, which maintain reasonable performance by leveraging the LTL rewards differentiability.*

7 Conclusion

In this work, we tackle the critical challenge of scalable RL for robotic systems under long-horizon, formally specified tasks. By adopting LTL as our specification framework, we ensure objective correctness and avoid the reward misspecification issues commonly encountered in conventional RL approaches. To overcome the learning inefficiencies caused by sparse logical rewards, we propose a novel method that leverages differentiable simulators, enabling gradient-based learning directly from LTL objectives without compromising their expressiveness or correctness. Our approach introduces soft labeling techniques that preserve the differentiability through the transitions of automata derived from LTL formulas, resulting in end-to-end differentiable learning framework. Through a series of simulated experiments, we demonstrate that our method substantially accelerates learning compared to SOTA non-differentiable baselines, paving the way for more reliable and scalable deployment of autonomous robotic systems in complex real-world environments.

References

[1] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[2] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[3] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[4] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[5] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[6] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[7] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

[8] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[9] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[10] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[11] Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient safe exploration via primal-dual policy optimization. In *International conference on artificial intelligence and statistics*, pages 3304–3312. PMLR, 2021.

[12] Puze Liu, Davide Tateo, Haitham Bou Ammar, and Jan Peters. Robot reinforcement learning on the constraint manifold. In *Conference on Robot Learning*, pages 1357–1366. PMLR, 2022.

[13] Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *NIPS*, 2017.

[14] Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *TAC*, 64(7):2737–2752, 2019.

[15] Richard Cheng, Gabor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *AAAI*, 2019.

[16] Björn Lütjens, Michael Everett, and Jonathan P How. Safe reinforcement learning with model uncertainty estimates. *ICRA*, 2019.

[17] Jaime F. Fisac, Neil F. Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. *ICRA*, 00:8550–8556, 2019.

[18] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. *RA-L*, 6(3):4915–4922, 2020.

[19] Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. *ICRA*, 00:7166–7172, 2020.

[20] Mario Zanon and Sebastien Gros. Safe reinforcement learning using robust MPC. *TAC*, 66(8):3638–3652, 2020.

[21] Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. *IROS*, 00:7139–7145, 2020.

[22] Jason Choi, Fernando Castaneda, Claire J. Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *RSS*, 2020.

[23] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Journal of Robotics Research*, 39(7):856–892, 2020.

[24] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *ICLR*, 2021.

[25] Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. *CoRL*, 2021.

[26] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *T-RO*, 39(3):1749–1767, 2023.

[27] Santiago Paternain, Miguel Calvo-Fullana, Luiz F. O. Chamon, and Alejandro Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *TAC*, 68(3):1321–1336, 2023.

[28] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 395–412, 2019.

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

[30] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic. Model-free reinforcement learning for stochastic games with linear temporal logic objectives. In *International Conference on Robotics and Automation (ICRA)*, pages 10649–10655. IEEE, 2021.

[31] A. K. Bozkurt, Y. Wang, and M. Pajic. Secure planning against stealthy attacks via model-free reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, pages 10656–10662. IEEE, 2021.

[32] A. K. Bozkurt, Y. Wang, and M. Pajic. Model-free learning of safe yet effective controllers. In *Conference on Decision and Control (CDC)*, pages 6560–6565. IEEE, 2021.

[33] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic. Learning optimal controllers for temporal logic specifications in stochastic games. *Transactions on Automatic Control (TAC)*, 2024.

[34] Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37), 2019.

[35] Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *RA-L*, 6(4):7973–7980, 2021.

[36] Mingyu Cai, Shaoping Xiao, Baoluo Li, Zhiliang Li, and Zhen Kan. Reinforcement learning based temporal logic control with maximum probabilistic satisfaction. *ICRA*, 00:806–812, 2021.

[37] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *JAIR*, 2022.

[38] Yiannis Kantaros. Accelerated reinforcement learning for temporal logic control objectives. *IROS*, 00:5077–5082, 2022.

[39] Cameron Voloshin, Hoang M Le, Swarat Chaudhuri, and Yisong Yue. Policy optimization with linear temporal logic constraints. *NeurIPS*, 2022.

[40] Cambridge Yang, Michael Littman, and Michael Carbin. On the (in)tractability of reinforcement learning for LTL objectives. *IJCAI*, 2022.

[41] Mingyu Cai, Shaoping Xiao, Junchao Li, and Zhen Kan. Safe reinforcement learning under temporal logic with reward design and quantum action selection. *Scientific Reports*, 13(1):1925, 2023.

[42] Hosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Certified reinforcement learning with logic guidance. *Artificial Intelligence*, 322:103949, 2023.

[43] Mingyu Cai, Erfan Aasi, Calin Belta, and Cristian-Ioan Vasile. Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications. *RA-L*, 8(4):2158–2165, 2023.

[44] Bohan Cui, Keyi Zhu, Shaoyuan Li, and Xiang Yin. Security-aware reinforcement learning under linear temporal logic specifications. *ICRA*, 00:12367–12373, 2023.

[45] Cameron Voloshin, Abhinav Verma, and Yisong Yue. Eventual discounting temporal logic counterfactual experience replay. *ICML*, 2023.

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

[47] Daiying Tian, Hao Fang, Qingkai Yang, Haoyong Yu, Wenyu Liang, and Yan Wu. Reinforcement learning under temporal logic constraints as a sequence modeling problem. *Robotics and Autonomous Systems*, 161:104351, 2023.

[48] Christos K. Verginis, Cevahir Koprulu, Sandeep Chinchali, and Ufuk Topcu. Joint learning of reward machines and policies in environments with partially known semantics. *Artificial Intelligence*, 333:104146, 2024.

[49] Xuan-Bach Le, Dominik Wagner, Leon Witzman, Alexander Rabinovich, and Luke Ong. Reinforcement learning with LTL and $\ omega$-regular objectives via optimality-preserving translation to average rewards. *NeurIPS*, 2024.

[50] Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. A PAC learning algorithm for LTL and omega-regular objectives in MDPs. *AAAI*, 38(19):21510–21517, 2024.

[51] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[52] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

[53] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*, pages 36593–36604. PMLR, 2023.

[54] Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International conference on machine learning*, pages 25636–25655. PMLR, 2022.

[55] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. *RSS*, 2021.

[56] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6565–6570, Dec 2016.

[57] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. *RSS*, 2018.

[58] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bacher, Bernhard Thomaszewski, and Stelian Coros. ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *TOG*, 2020.

[59] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. *ICML*, 2021.

[60] Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. *RSS*, 2021.

[61] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *RSS*, 2021.

[62] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. DiSECt: A differentiable simulation engine for autonomous robotic cutting. *RSS*, 2021.

[63] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. *NeurIPS*, 2021.

[64] Miguel Zamora, Momchil Peychev, Sehoon Ha, Martin Vechev, and Stelian Coros. PODS: Policy optimization via differentiable simulation. *ICML*, 2021.

[65] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. DiffPD: Differentiable projective dynamics. *TOG*, 41(2):1–21, 2021.

[66] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. *ICLR*, 2021.

[67] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[68] Junbang Liang, Ming C. Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *NeurIPS*, pages 1–22, 2019.

[69] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. *ICRA*, 00:6265–6271, 2019.

[70] Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

[71] Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pages 4065–4074. PMLR, 2018.

[72] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

[73] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. *ICLR*, 2022.

[74] Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learning in contact-rich differentiable simulation. *ICML*, 2024.

[75] Sanghyun Son, Laura Yu Zheng, Ryan Sullivan, Yi-Ling Qiao, and Ming C. Lin. Gradient informed proximal policy optimization. *NeurIPS*, 2023.

[76] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6):356–370, 2023.

[77] Yue Meng and Chuchu Fan. Signal temporal logic neural predictive control. *RAL*, 8(11):7719–7726, 2023.

[78] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, USA, 2008.

[79] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification*, pages 312–332, Cham, 2016. Springer International Publishing.

[80] Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394–413, 2012.

[81] Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. Owl: A library for ω-words, automata, and LTL. In *Proceedings of the 16th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 11138 of *LNCS*, pages 543–550, 2018.

[82] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0—a framework for ltl and-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 122–129, 2016.

[83] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[84] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

A Appendix

Please check the supplemental material.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose, to our knowledge, the first approach utilizing differentiable simulators to accelerate learning from LTL specifications, and we show the efficacy of our approach in experiments section.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our paper includes an explicit section dedicated to the limitations of our approach.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All assumptions are explicitly formalized for each theoretical statement. We provide intuitive explanations alongside the proofs, cite relevant resources for complete proofs, and include additional formalizations and explanations in the supplemental material.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We clearly outlined the steps of our approach and provided pseudo-code, along with a reference to the paper from which all hyperparameters were adopted.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is included in the supplemental material.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: We thoroughly specified the environment details for each experiment, with additional information provided in the supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We showed standard deviation for each learning curve, as well as minimum, maximum, and mean values for the final policies.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes] .

Justification: We included the information regarding the computer resources we used in the experiments section.

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We reviewed the NeurIPS Code of Ethics and confirm that the research conducted fully adheres to it.

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work does not have direct societal impacts.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper does not present any such risks.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All tools, models, and hyperparameters used in our work are properly cited.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is properly documented and included with a read-me file in the supplemental material.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our work does not involve crowdsourcing or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our work does not involve crowdsourcing or research with human subjects.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The LLMs were used solely for grammar checks in this work.