Adaptive Reward Design for Reinforcement Learning

Minjae Kwon[1] **Ingy ElSayed-Aly**[1] **Lu Feng**[1]

[1]The Department of Computer Science, University of Virginia, Charlottesville, VA 22904, USA

Abstract

There is a surge of interest in using formal languages such as Linear Temporal Logic (LTL) to precisely and succinctly specify complex tasks and derive reward functions for Reinforcement Learning (RL). However, existing methods often assign sparse rewards (e.g., giving a reward of 1 only if a task is completed and 0 otherwise). By providing feedback solely upon task completion, these methods fail to encourage successful subtask completion. This is particularly problematic in environments with inherent uncertainty, where task completion may be unreliable despite progress on intermediate goals. To address this limitation, we propose a suite of reward functions that incentivize an RL agent to complete a task specified by an LTL formula as much as possible, and develop an adaptive reward shaping approach that dynamically updates reward functions during the learning process. Experimental results on a range of benchmark RL environments demonstrate that the proposed approach generally outperforms baselines, achieving earlier convergence to a better policy with higher expected return and task completion rate. Code is available at `https://github.com/safe-autonomy-lab/AdaptiveRewardRL.git`.

1 INTRODUCTION

In reinforcement learning (RL), an agent's behavior is guided by reward functions, which are often difficult to specify manually when representing complex tasks. Alternatively, an RL agent can infer the intended reward from demonstrations Ng and Russell [2000], trajectory comparisons Wirth et al. [2017], or human instructions Fu et al. [2018]. Recent years have seen a surge of interest in using formal languages such as Linear Temporal Logic (LTL) and finite automata to specify complex tasks and derive reward functions for RL (see the extensive list of related work in Section 1.1). Nevertheless, existing methods often assign sparse rewards (e.g., giving a reward of 1 only if a task is completed and 0 otherwise). Sparse rewards may necessitate hundreds of thousands of exploratory episodes for convergence to a quality policy. Furthermore, many prior works are only compatible with specific RL algorithms tailored to their proposed reward structures, such as Q-learning for reward machines Camacho et al. [2019], modular DDPG Hasanbeig et al. [2020], and hierarchical RL for reward machines Icarte et al. [2022].

Reward shaping Ng et al. [1999] is a paradigm where an agent receives some intermediate rewards as it gets closer to the goal and has shown to be helpful for RL algorithms to converge more quickly. Inspired by this idea, we develop a logic-based adaptive reward shaping approach in this work. We use the syntactically co-safe fragment of LTL to specify complex RL tasks, such as "the task is to touch red and green balls in strict order without touching other colors, then touch blue balls". We then translate a co-safe LTL task specification into a deterministic finite automaton (DFA) and design reward functions that keeps track of the task completion status (i.e., a task is completed if an accepting state of the DFA has been reached).

The principle underlying our approach is to assign intermediate rewards to an agent as it makes progress toward completing a task. A key challenge is how to measure the closeness to task completion. We adopt the notion of *task progression* defined by Lacerda et al. [2019], which measures each DFA state's distance to accepting states. The smaller the distance, the higher degree of task progression. The distance is zero when the task is fully completed.

Another challenge is what reward values to assign for various degrees of task progression. To this end, we design two different reward functions. The *progression* reward function assigns rewards based on the reduced distance-to-acceptance

values. The *hybrid* reward function balances the progression reward and the penalty for self-loops (i.e., staying in the same DFA state). However, we find that optimal policies maximizing the expected return based on these reward functions may not necessarily lead to the best possible task progression.

To address this limitation, we develop an adaptive reward shaping approach that dynamically updates distance-to-acceptance values to reflect the actual difficulty of activating DFA transitions during the learning process. We then design two new reward functions, namely *adaptive progression* and *adaptive hybrid*, leveraging the updated distance-to-acceptance values. We show that our approach can learn an optimal policy with the highest expected return and the best task progression within a finite number of updates.

Finally, we evaluate the proposed approach on various discrete and continuous RL environments. Computational experiments show the compatibility of our approach with a wide range of RL algorithms. Results indicate our approach generally outperforms baselines, achieving earlier convergence to a better policy with higher expected return and task completion rate.

1.1 RELATED WORK

Li et al. [2017] presents one of the first works applying temporal logic to reward function design, assigning reward functions based on robustness degrees of satisfying truncated LTL formulas. De Giacomo et al. [2019] uses a fragment of LTL for finite traces (called LTL_f) to encode RL rewards. Several methods seek to learn optimal policies that maximize the probability of satisfying an LTL formula Hasanbeig et al. [2019], Bozkurt et al. [2020], Hasanbeig et al. [2020]. However, these methods assign sparse rewards for task completion and do not provide intermediate rewards for task progression.

There is a line of work on *reward machines* (RMs), a type of finite state machine that takes labels representing environment abstractions as input and outputs reward functions. Camacho et al. [2019] shows that LTL and other regular languages can be automatically translated into RMs via the construction of DFAs. Icarte et al. [2022] describes a collection of RL methods that exploit the RM structure, including *Q-learning for reward machines* (QRM), *counterfactual experiences for reward machines* (CRM), and *hierarchical RL for reward machines* (HRM). These methods are augmented with potential-based reward shaping Ng et al. [1999], where a potential function over RM states is computed to assign intermediate rewards. We adopt these methods (with reward shaping) as baselines for comparison in our experiments. As we will show in Section 5, our approach generally outperforms baselines, providing more effective design of intermediate rewards for task progression.

Jothimurugan et al. [2019] proposes a new specification language that can be translated into reward functions and later applies it for compositional RL in Jothimurugan et al. [2021]. These methods use a task monitor to track the degree of specification satisfaction and assign intermediate rewards. However, they require users to encode atomic predicates into quantitative values for reward assignment. In contrast, our approach automatically assigns intermediate rewards using DFA states' distance to acceptance values, eliminating the need for user-provided functions.

Jiang et al. [2021] presents a reward shaping framework for average-reward learning in continuing tasks. Their method automatically translates a LTL formula encoding domain knowledge into a function that provides additional reward throughout the learning process. This work has a different problem setup and thus is not directly comparable with our approach.

Cai et al. [2023a] proposes an approach that decomposes an LTL mission into sub-goal-reaching tasks solved in a distributed manner. The same authors also present a model-free RL method for minimally violating an infeasible LTL specification in Cai et al. [2023b]. Both works consider the assignment of intermediate rewards, but their definition of task progression requires additional information about the environment (e.g., geometric distance from each waypoint to the destination). In contrast, we define task progression based solely on the task specification, following Lacerda et al. [2019], which is a work on robotic planning with MDPs (but not RL).

2 BACKGROUND

2.1 REINFORCEMENT LEARNING

Consider an RL agent interacting with an environment modeled as an episodic *Markov decision process* (MDP), where each learning episode terminates within a finite horizon H. Formally, an MDP is denoted as a tuple $\mathcal{M} = (S, s_0, A, T, R, \gamma, L)$ where S is a set of states, $s_0 \in S$ is an initial state, A is a set of actions, $T : S \times A \times S \to [0, 1]$ is a probabilistic transition function, R is a reward function, $\gamma \in [0, 1]$ is a discount factor, and $L : S \to 2^{AP}$ is a labeling function with a set of atomic propositions AP. The reward function can be Markovian, denoted by $R : S \times A \times S \to \mathbb{R}$, or non-Markovian (i.e., history dependent), denoted by $R : (S \times A)^* \to \mathbb{R}$. Both the transition function T and the reward function R are unknown to the agent.

At each timestep t, the agent selects an action a_t given the current state s_t and reward r_t. The environment transitions to a subsequent state s_{t+1}, determined by the probability distribution $T(\cdot|s_t, a_t)$, and yields a reward r_{t+1}. A (memoryless) policy is defined as a mapping from states to probabil-

ity distributions over actions, denoted by $\pi : S \times A \to [0, 1]$. The agent seeks to learn an optimal policy that maximizes the expected return, represented by $\mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r_{t+1}]$.

2.2 CO-SAFE LTL SPECIFICATIONS

We utilize Linear Temporal Logic (LTL) Pnueli [1981], which is a form of modal logic that augments propositional logic with temporal operators, to specify complex tasks for the robotic agent. We focus on the syntactically co-safe LTL fragment, defined as follows.

$$\varphi := \alpha \mid \neg\alpha \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \Diamond\varphi$$

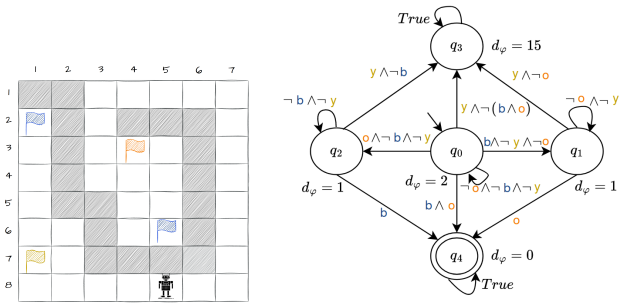
where $\alpha \in AP$ is an atomic proposition, $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are Boolean operators, while $\bigcirc$ (next), U (until), and $\Diamond$ (eventually) are temporal operators. Intuitively, $\bigcirc\varphi$ means that φ has to hold in the next step; $\varphi_1 \mathsf{U} \varphi_2$ means that φ_1 has to hold at least until φ_2 becomes true; and $\Diamond\varphi$ means that φ becomes true at some time eventually. A co-safe LTL formula φ can be converted into a DFA $\mathcal{A}_\varphi$ accepting exactly the set of good prefixes for φ Kupferman and Vardi [2001]. Formally, a DFA is denoted as a tuple $\mathcal{A}_\varphi = (Q, q_0, Q_F, 2^{AP}, \delta)$, where Q is a finite set of states, q_0 is the initial state, $Q_F \subseteq Q$ is a set of accepting states, 2^{AP} is the alphabet, and $\delta : Q \times 2^{AP} \to Q$ is the transition function.

Example 1. Consider a robot aiming to complete a task in a gridworld (Figure 1a). The task is to collect an *orange* flag and a *blue* flag (in any order) while avoiding the *yellow* flag. We describe this task using a co-safe LTL formula $\varphi = (\neg y)\mathsf{U}((o \wedge ((\neg y)\mathsf{U}b)) \vee (b \wedge ((\neg y)\mathsf{U}o)))$, where o, b and y represent collecting *orange*, *blue* and *yellow* flags, respectively. Figure 1b shows the corresponding DFA $\mathcal{A}_\varphi$, which has five states including the initial state q_0 depicted with an incoming arrow, a trap state q_3 from which no transitions to other states exist, and the accepting state $Q_F = \{q_4\}$ depicted with double circle. A transition is enabled when its labelled Boolean formula holds. Starting from the initial state q_0, a path ending in the accepting state q_4 represents a good prefix of satisfying φ, indicating that the task has been successfully completed. ■

2.3 TASK PROGRESSION

We adopt the notion of "task progression" introduced in Lacerda et al. [2019] to measure the degree to which a robotic task defined by a co-safe LTL formula φ is completed.

Given a DFA $\mathcal{A}_\varphi = (Q, q_0, Q_F, 2^{AP}, \delta)$, let $\mathrm{Suc}_q \subseteq Q$ be the set of successors of state q, and $|\delta_{q,q'}| \in \{0, \ldots, 2^{|AP|}\}$ denote the number of possible transitions from q to q'. We write $q \to^* q'$ if there is a path from q to q', and $q \not\to^* q'$ if q' is not reachable from q.



(a) Gridworld (b) DFA $\mathcal{A}_\varphi$

Figure 1: Example gridworld and a DFA $\mathcal{A}_\varphi$ for a co-safe LTL formula $\varphi = (\neg y)\mathsf{U}((o \wedge ((\neg y)\mathsf{U}b)) \vee (b \wedge ((\neg y)\mathsf{U}o)))$.

The *distance-to-acceptance function* $d_\varphi : Q \to \mathbb{R}_{\geq 0}$ is defined as:

$$d_\varphi(q) = \begin{cases} 0 & \text{if } q \in Q_F \\ \min\limits_{q' \in \mathrm{Suc}_q} d_\varphi(q') + h(q, q') & \text{if } q \notin Q_F, q \to^* Q_F \\ |AP| \cdot |Q| & \text{otherwise} \end{cases} \tag{1}$$

where $h(q, q') := \log_2 \left(\left\{ \frac{2^{|AP|}}{|\delta_{q,q'}|} \right\} \right)$ represents the difficulty of moving from q to q' in the DFA $\mathcal{A}_\varphi$.

The *progression function* $\rho_\varphi : Q \times Q \to \mathbb{R}_{\geq 0}$ between two states of $\mathcal{A}_\varphi$ is defined as:

$$\rho_\varphi(q, q') = \begin{cases} \max\{0, d_\varphi(q) - d_\varphi(q')\} & \text{if } q' \in \mathrm{Suc}_q,\ q' \not\to^* q \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The first condition mandates $q' \not\to^* q$ to ensure that there is no cycle in the DFA with a non-zero progression value, which is crucial for the convergence of infinite sums of progression Lacerda et al. [2019].

Example 2. In the DFA $\mathcal{A}_\varphi$ (Figure 1b), the distance-to-acceptance values of the trap state q_3 and the accepting state q_4 is $d_\varphi(q_3) = 3 \times 5 = 15$ and $d_\varphi(q_4) = 0$, respectively. Applying Equation 1 recursively yields $d_\varphi(q_0) = 2$, $d_\varphi(q_1) = 1$, and $d_\varphi(q_2) = 1$. The progression from the initial state q_0 to q_1 is $\rho_\varphi(q_0, q_1) = \max\{0, d_\varphi(q_0) - d_\varphi(q_1)\} = 1$, indicating that a positive task progression has been made. ■

3 PROBLEM FORMULATION

The objective of this work is to create reward functions that encourage an RL agent to achieve the best possible progression in accomplishing a task specified by a co-safe LTL formula φ. To this end, we define a product MDP $\mathcal{M}^\otimes$ that augments the environment MDP $\mathcal{M}$ with information about the task specification φ.

Product MDP. Given an episodic MDP $\mathcal{M} = (S, s_0, A, T, R, \gamma, L)$ and a DFA $\mathcal{A}_\varphi = (Q, q_0, Q_F, 2^{AP}, \delta)$, the product MDP is defined as $\mathcal{M}^\otimes = \mathcal{M} \otimes \mathcal{A}_\varphi = (S^\otimes, s_0^\otimes, A, T^\otimes, R^\otimes, \gamma, AP, L^\otimes)$, where $S^\otimes = S \times Q$, $s_0^\otimes = \langle s_0, \delta(q_0, L(s_0))\rangle$, $L^\otimes(\langle s, q\rangle) = L(s)$,

$$T^\otimes\left(\langle s, q\rangle, a, \langle s', q'\rangle\right) = \begin{cases} T(s, a, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise.} \end{cases}$$

This work focuses on designing Markovian reward functions $R^\otimes : S^\otimes \times A \times S^\otimes \to \mathbb{R}$ for the product MDP $\mathcal{M}^\otimes$, whose projection onto $\mathcal{M}$ yields non-Markovian reward functions.

In practice, the product MDP is built on-the-fly during learning. At each timestep t, given the current state $\langle s_t, q_t\rangle$, an RL agent selects an action a_t and transits to a successor state $\langle s_{t+1}, q_{t+1}\rangle$, where s_{t+1} is given by the environment, sampling from the distribution $T(\cdot|s_t, a_t)$, and q_{t+1} is derived from the DFA's transition function $\delta(q_t, L(s_{t+1}))$. The agent receives a reward r_{t+1} determined by the reward function $R^\otimes\left(\langle s_t, q_t\rangle, a, \langle s_{t+1}, q_{t+1}\rangle\right)$.

An RL agent aims to learn an optimal policy that maximizes the expected return in the product MDP $\mathcal{M}^\otimes$. A learned memoryless policy for $\mathcal{M}^\otimes$ equates to a finite-memory policy in the environment MDP $\mathcal{M}$, denoted by $\pi : S \times Q \times A \to [0, 1]$, with the DFA states Q delineating various modes.

Task progression for a policy. We define a partition of the state space of DFA $\mathcal{A}_\varphi = (Q, q_0, Q_F, 2^{AP}, \delta)$ based on an ordering of distance-to-acceptance values. Let $B_0 = Q_F$ and $B_i = \{q \in Q \setminus \bigcup_{j=0}^{i-1} B_j \mid d_\varphi(q) \text{ is minimal}\}$ for $i > 0$. The task progression for a policy π of the product MDP, denoted by $b(\pi)$, is the lowest index of reachable partitioned sets B_i from the initial state. A value of $b(\pi) = 0$ signifies the task has been successfully completed. The best possible task progression across all feasible policies Π in the product MDP is defined as $b^* = \min\{b(\pi) \mid \pi \in \Pi\}$.

Example 3. The state space of the DFA $\mathcal{A}_\varphi$ shown in Figure 1b can be partitioned into four sets: $B_0 = \{q_4\}$, $B_1 = \{q_1, q_2\}$, $B_2 = \{q_0\}$, and $B_3 = \{q_3\}$.

Let $g_{i,j}$ denote a grid cell in row i and column j in the gridworld (Figure 1a). The agent's initial location is $g_{8,5}$. Consider the following three candidate policies:

- π_1: The agent takes 10 steps to collect the blue flag in $g_{2,1}$, avoiding the yellow flag, but fails to reach the orange flag within the 25-step episode timeout.
- π_2: The agent moves 16 steps to collect the orange flag and then moves 4 more steps to collect the blue flag in $g_{6,5}$. The task is completed.
- π_3: The agent moves directly to the yellow flag in 5 steps. The task is failed and the episode ends.

We have $b(\pi_1) = 1$ as DFA state $q_1 \in B_1$ is reached with policy π_1, $b(\pi_2) = 0$ upon task completion, and $b(\pi_3) = 2$ due to a direct transition from initial state $q_0 \in B_2$ to trap state $q_3 \in B_3$. The best possible task progression across all policies is $b^* = b(\pi_2) = 0$. ■

Problem. This work aims to solve the following problem: Given an episodic MDP $\mathcal{M}$ with unknown transition and reward functions, along with a DFA $\mathcal{A}_\varphi$ representing a co-safe LTL task specification φ, the objective is to construct a Markovian reward function $R^\otimes$ for the product MDP $\mathcal{M}^\otimes = \mathcal{M} \otimes \mathcal{A}_\varphi$. This reward function should be designed such that an optimal policy π^*, learned by an RL agent via maximizing the expected return, also achieves the best possible task progression, that is, $b^* = b(\pi^*)$.

4 APPROACH

To solve this problem, we design two reward functions that incentivize an RL agent to improve the task progression (cf. Section 4.1), and develop an adaptive reward shaping approach that dynamically updates these reward functions during the learning process (cf. Section 4.2).

4.1 BASIC REWARD FUNCTIONS

Progression reward function. First, we propose a *progression reward function* based on the task progression function defined in Equation 2, representing the degree of reduction in distance-to-acceptance values.

$$\begin{aligned} &R^\otimes_{\mathsf{pg}}\left(\langle s, q\rangle, a, \langle s', q'\rangle\right) = \rho_\varphi(q, q') \\ &= \begin{cases} \max\{0, d_\varphi(q) - d_\varphi(q')\} & \text{if } q' \in \mathrm{Suc}_q,\ q' \not\to^* q \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

Example 4. Assuming a deterministic environment for the gridworld shown in Figure 1a, the MDP has a discount factor of $\gamma = 0.9$. We calculate the expected returns for policies from Example 3 using the progression reward function. $V^{\pi_1}_{\mathsf{pg}}(s_0^\otimes) = 0.9^9 \approx 0.39$, $V^{\pi_2}_{\mathsf{pg}}(s_0^\otimes) = 0.9^{15} + 0.9^{19} \approx 0.34$, and $V^{\pi_3}_{\mathsf{pg}}(s_0^\otimes) = 0$. Among these policies, π_1 yields the highest expected return, yet it fails to achieve the best possible task progression, as $b(\pi_1) = 1 > b^* = 0$. ■

Hybrid reward function. The progression reward function rewards only transitions that progress toward acceptance, without penalizing those that stay in the same DFA state. To

address this issue, we define a *hybrid reward function*:

$$R^{\otimes}_{\mathsf{hd}}\left(\langle s,q\rangle,a,\langle s',q'\rangle\right)=\begin{cases}\eta\cdot -d_{\varphi}(q) & \text{if } q=q'\\ (1-\eta)\cdot\rho_{\varphi}(q,q') & \text{otherwise}\end{cases} \tag{4}$$

where $\eta\in[0,1]$ balances the trade-offs between penalties and progression rewards.

Example 5. We calculate the expected returns of policies in Example 3 using the hybrid reward function (with $\eta=0.1$). $V^{\pi_1}_{\mathsf{hd}}(s^{\otimes}_0)\approx -1.15$, $V^{\pi_2}_{\mathsf{hd}}(s^{\otimes}_0)\approx -1.33$, and $V^{\pi_3}_{\mathsf{hd}}(s^{\otimes}_0)\approx -0.69$. Although π_3 yields the highest expected return, it falls short in the task progression with $b(\pi_3)=2$. Increasing η emphasizes penalties without altering the optimal policy in this example. Conversely, reducing η moves closer to the progression reward function, especially when $\eta=0$. ■

4.2 ADAPTIVE REWARD SHAPING

While reward functions defined in Section 4.1 motivate an RL agent to complete a task specified by a co-safe LTL formula, Examples 4 and 5 show that the learned optimal policies that maximize the expected return do not achieve the best possible task progression. A potential reason is that the distance-to-acceptance function d_{φ}, as defined in Equation 1, may not precisely reflect the difficulty of activating desired DFA transitions within a specific environment. To tackle this limitation, we develop an adaptive reward shaping approach that dynamically updates distance-to-acceptance values and reward functions during the learning process.

Updating distance-to-acceptance values. After every N learning episodes, with N being a hyperparameter, we evaluate the average success rate of task completion. An episode is deemed successful if it concludes in an accepting state of the DFA $\mathcal{A}_{\varphi}$. If the average success rate falls below a predefined threshold λ, we proceed to update the distance-to-acceptance values accordingly.

We derive initial values $d^0_{\varphi}(q)$ for each DFA state $q\in Q$ from Equation 1. The distance-to-acceptance values for the k-th update round are calculated recursively as follows:

$$d^k_{\varphi}(q)=\begin{cases}d^{k-1}_{\varphi}(q)+\theta & \text{if } q\in B_i,\forall i\geq b_k\\ d^{k-1}_{\varphi}(q) & \text{otherwise}\end{cases} \tag{5}$$

where b_k is the task progression of the optimal policy learned after $k\cdot N$ episodes, and θ is a hyperparameter, also used later in Equation 8, requiring that $\theta>1$.

Example 6. We have $d^0_{\varphi}(q_0)=2$, $d^0_{\varphi}(q_1)=d^0_{\varphi}(q_2)=1$, $d^0_{\varphi}(q_3)=15$, and $d^0_{\varphi}(q_4)=0$ following Example 2. Suppose π_1 is the optimal policy learned after the first N episodes and thus $b_1=1$. Let $\theta=100$. For states in $B_1\cup B_2\cup B_3=\{q_0,q_1,q_2,q_3\}$, We update their distance-to-acceptance values as follows: $d^1_{\varphi}(q_1)=d^1_{\varphi}(q_2)=101$, $d^1_{\varphi}(q_0)=102$, and $d^1_{\varphi}(q_3)=115$. For state $q_4\in B_0$, we retain its distance-to-acceptance value as $d^1_{\varphi}(q_4)=0$. ■

Note that Equation 5 does not alter the order of distance-to-acceptance values, so the DFA state partitions $\{B_i\}$ remain unchanged. We present two new reward functions that leverage the updated distance-to-acceptance values as follows.

Adaptive progression reward function. Given the updated distance-to-acceptance values $d^k_{\varphi}(q)$, we apply the progression function defined in Equation 2 and obtain

$$\rho^k_{\varphi}(q,q')=\begin{cases}\max\{0,d^k_{\varphi}(q)-d^k_{\varphi}(q')\} & \text{if } q'\in \mathrm{Suc}_q,\ q'\not\twoheadrightarrow^* q\\ 0 & \text{otherwise}\end{cases} \tag{6}$$

Then, we define an *adaptive progression reward function* for the k-th round of updates as:

$$R^{\otimes}_{\mathsf{ap},k}\left(\langle s,q\rangle,a,\langle s',q'\rangle\right)=\max\{\rho^0_{\varphi}(q,q'),\rho^k_{\varphi}(q,q')\} \tag{7}$$

When $k=0$, the adaptive progression reward function $R^{\otimes}_{\mathsf{ap},0}$ coincides with the progression reward function $R^{\otimes}_{\mathsf{pg}}$ defined in Equation 3.

Example 7. Using the updated distance-to-acceptance values from Example 6, we calculate the adaptive progression rewards $R^{\otimes}_{\mathsf{ap},1}$ for the first round of update. For instance, we have $\rho^1_{\varphi}(q_1,q_4)=\max\{0,d^1_{\varphi}(q_1)-d^1_{\varphi}(q_4)\}=101$. Recall $\rho^0_{\varphi}(q_1,q_4)=1$ from Example 2. Thus,

$$R^{\otimes}_{\mathsf{ap},1}\left(\langle g_{6,4},q_1\rangle,\text{right},\langle g_{6,5},q_4\rangle\right)=\max\{1,101\}=101.$$

The expected returns of policies in Example 3 with $R^{\otimes}_{\mathsf{ap},1}$ are $V^{\pi_1}_{\mathsf{ap},1}(s^{\otimes}_0)\approx 0.39$, $V^{\pi_2}_{\mathsf{ap},1}(s^{\otimes}_0)\approx 13.85$, and $V^{\pi_3}_{\mathsf{ap},1}(s^{\otimes}_0)=0$. Policy π_2 yields the highest expected return while completing the task (i.e., $b(\pi_2)=0$). ■

Adaptive hybrid reward function. We define an *adaptive hybrid reward function* for the k-th round of updates as:

$$R^{\otimes}_{\mathsf{ah},k}\left(\langle s,q\rangle,a,\langle s',q'\rangle\right)=\\ \begin{cases}\eta_k\cdot -d^k_{\varphi}(q) & \text{if } q=q'\\ (1-\eta_k)\cdot\max\{\rho^0_{\varphi}(q,q'),\rho^k_{\varphi}(q,q')\} & \text{otherwise}\end{cases} \tag{8}$$

with $\eta_0\in[0,1]$, and $\eta_k=\frac{\eta_{k-1}}{\theta}$ where θ is the same hyperparameter used in Equation 5. We require $\theta>1$ to ensure that the weight value η_k is reduced in each update round, avoiding undesired behavior from increased self-loop penalties. At $k=0$, the adaptive hybrid reward function $R^{\otimes}_{\mathsf{ah},0}$ aligns with the hybrid reward function $R^{\otimes}_{\mathsf{hd}}$ as defined in Equation 4.

Example 8. Let $\eta_0=0.1$, and $\theta_1=100$. The initial distance-to-acceptance values d^0_{φ} are the same as in Example 6. Suppose the agent's movement during the episodes follows

a policy π such that $b(\pi) = 1$. Following Equation 5, we update the distance-to-acceptance values of states in $B_1 \cup B_2 \cup B_3 = \{q_0, q_1, q_2, q_3\}$ to $d^1_\varphi(q_1) = d^1_\varphi(q_2) = 101$, $d^1_\varphi(q_0) = 102$, and $d^1_\varphi(q_3) = 115$. We compute $R^\otimes_{\mathsf{ah},1}$ with $\eta_1 = 0.001$, which yields $V^{\pi_1}_{\mathsf{ah},1}(s^\otimes_0) \approx -0.52$, $V^{\pi_2}_{\mathsf{ah},1}(s^\otimes_0) \approx 12.97$, and $V^{\pi_3}_{\mathsf{ah},1}(s^\otimes_0) \approx -0.35$. The optimal policy π_2 not only yields the highest expected return but also completes the task with $b(\pi_2) = 0$. ■

Correctness. The correctness of the proposed adaptive reward shaping approach, as it pertains to the problem formulated in Section 3, is stated below, with the proof provided in Appendix A.

Theorem 1. *Given an episodic MDP $\mathcal{M}$ and a DFA $\mathcal{A}_\varphi$ corresponding to a co-safe LTL formula φ, there exists an optimal policy π^* of the product MDP $\mathcal{M}^\otimes = \mathcal{M} \otimes \mathcal{A}_\varphi$ that maximizes the expected return based on a reward function $R^\otimes \in \{R^\otimes_{\mathsf{ap},k}, R^\otimes_{\mathsf{ah},k}\}$ for some $k \in \mathbb{N}$, where the task progression for policy π^* matches the best possible task progression b^* across all feasible policies in the product MDP $\mathcal{M}^\otimes$, that is, $b^* = b(\pi^*)$.*

5 EXPERIMENTS

We evaluate the proposed adaptive reward shaping approach in a variety of benchmark RL domains. We describe the experimental setup including environments, RL algorithms, baselines, and evaluation metrics in Section 5.1, and analyze the experimental results in Section 5.2.

5.1 EXPERIMENTAL SETUP

Environments. The following RL domains are used: the taxi domain from OpenAI Gym [Brockman et al., 2016], and three other domains adapted from Icarte et al. [2022].

- *Office world*: The agent navigates a 12×9 grid world to: get coffee and mail (in any order), deliver them to the office, and avoid obstacles. The test environment assigns a reward of 1 for each sub-goal: (i) get coffee, (ii) get coffee and mail, and (iii) deliver coffee and mail to the office, all while avoiding obstacles.
- *Taxi world*: The agent drives around a 5×5 grid world to pick up and drop off a passenger, starting from a random location. There are five possible pickup locations and four possible destinations. The task is completed when the passenger is dropped off at the target destination. The test environment assigns a reward of 1 for each sub-goal: (i) pick up the passenger, (ii) reach the target destination, and (iii) drop off the passenger.
- *Water world*: The agent moves in a continuous 2D box with six colored floating balls, changing velocity toward one of the four cardinal directions each step. The task is to touch red and green balls in strict order without touching other colors, then touch blue balls. The test environment assigns a reward of 1 for touching each target ball.
- *HalfCheetah*: The agent is a cheetah-like robot with a continuous action space, controlling six joints to move. The task is completed by reaching the farthest location. The test environment assigns a reward of 1 for reaching each of the five locations along the way.

For each domain, we consider three types of environments: (1) *deterministic* environments, where each state-action pair leads to a single success state only; (2) *noisy* environments, where each action has a certain control noise; and (3) *infeasible* environments, where some sub-goals are impossible to complete (e.g., a blocked office that the agent cannot access, or missing blue balls in the water world).

Baselines. We compare the proposed approach with the following methods as baselines: *Q-learning for reward machines* (QRM) with reward shaping [Camacho et al., 2019], *counterfactual experiences for reward machines* (CRM) with reward shaping and *hierarchical RL for reward machines* (HRM) with reward shaping [Icarte et al., 2022]. We also evaluate RM-based algorithms incorporating partial rewards, which are detailed in Appendix C. We use the code accompanying publications.

Moreover, we consider a naive baseline that rewards transitions that decrease the distance to acceptance. For each transition $(\langle s, q\rangle, a, \langle s', q'\rangle)$ in the product MDP, assign a reward of 1 if $d_\varphi(q) > d_\varphi(q')$ and there is a path from q to accepting states Q_F, otherwise assign a reward of 0.

RL Algorithms. We use DQN Mnih et al. [2015] for learning in discrete domains (office world and taxi world), DDQN [Van Hasselt et al., 2016] for water world with continuous state space, and DDPG [Lillicrap et al., 2016] for HalfCheetah with continuous action space. Note that QRM implementation does not work with DDPG, so we only use HRM and CRM as the baselines for HalfCheetah. We also apply PPO [Schulman et al., 2017] and A2C [Mnih et al., 2016] to HalfCheetah (QRM, CRM and HRM baselines are not compatible with these RL algorithms) and report results in Appendix B due to the page limit. Our implementation was built upon OpenAI Stable-Baselines3 [Raffin et al., 2021].

Metrics. We pause the learning process every 100 training steps in the office world and every 1,000 training steps in other domains, then evaluate the current policy in the test environment over 5 episodes. We evaluate the performance using two metrics: *success rate of task completion*, calculated by counting the frequency of successful episodes where the task is completed, and *normalized expected return*, which

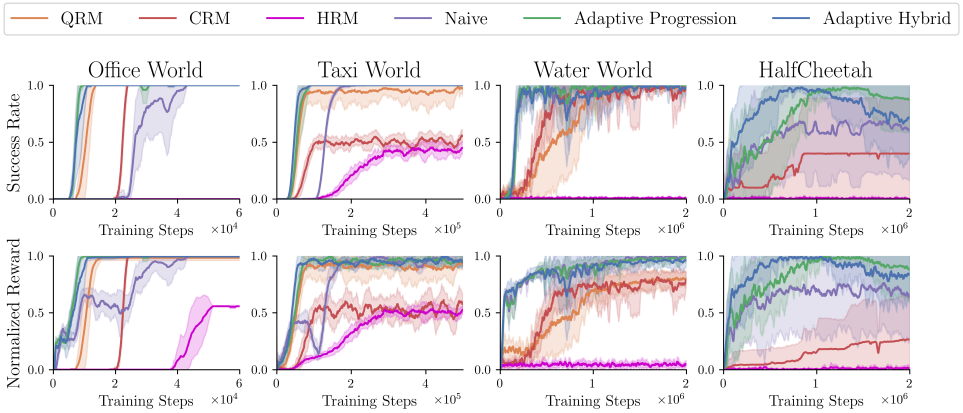


Figure 2: Results for deterministic environments.

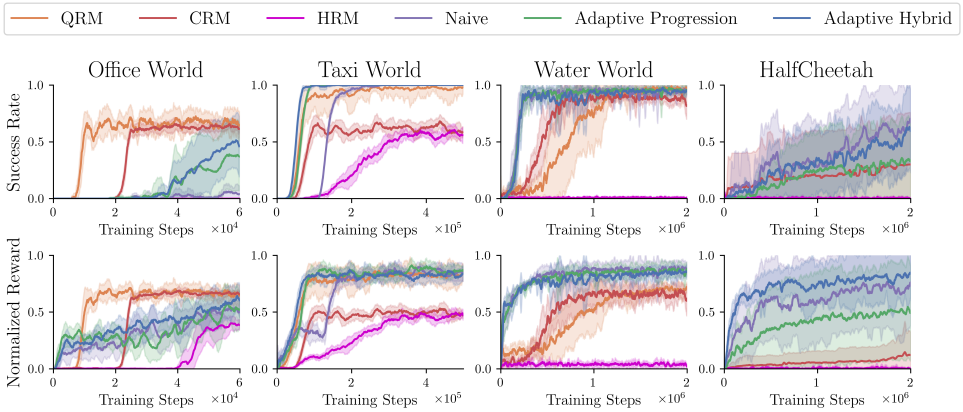


Figure 3: Results for noisy environments.

is normalized using the maximum possible return for that task. The only exception is taxi world, where the maximum return varies for different initial states and we normalize by averaging the maximum return of all initial states.

5.2 RESULTS ANALYSIS

We ran 10 independent trials for each method. Figures 2, 3 and 4 plot the mean performance with a 95% confidence interval (the shaded area) in deterministic, noisy, and infeasible environments, respectively. The success rate of task completion is omitted in Figure 4 because it is zero for all trials (i.e., the task is infeasible to complete).

Performance comparison. These results show that the proposed approach using adaptive progression or adaptive hybrid reward functions generally outperforms baselines, achieving earlier convergence to policies with a higher success rate of task completion and a higher normalized expected return.

The significant advantage of our approach is best illustrated in Figure 4, where baselines fail to learn effectively in environments with infeasible tasks. Although baselines apply potential-based reward shaping [Ng et al., 1999] to assign intermediate rewards, they cannot distinguish between good and bad terminal states (e.g., completing a sub-goal and colliding with an obstacle have the same potential value). In contrast, our approach provides more effective intermediate rewards, encouraging the agent to learn and maximize task progression.

The only outlier is the noisy office world where QRM and CRM outperform the proposed approach. One possible reason is that our approach gets stuck with a sub-optimal policy in this environment, which opts for fetching coffee at a closer location but results in a longer route to complete other sub-goals (i.e., getting mail and delivering to office).

Comparing the proposed reward functions, we observe that the adaptive hybrid reward function has the best overall performance. Comparing different RL environments, the proposed approach can achieve a success rate of 1 and the maximum possible return in most deterministic environments, but its performance is degraded in noisy environments due to control noise and in infeasible environments due to environmental constraints.

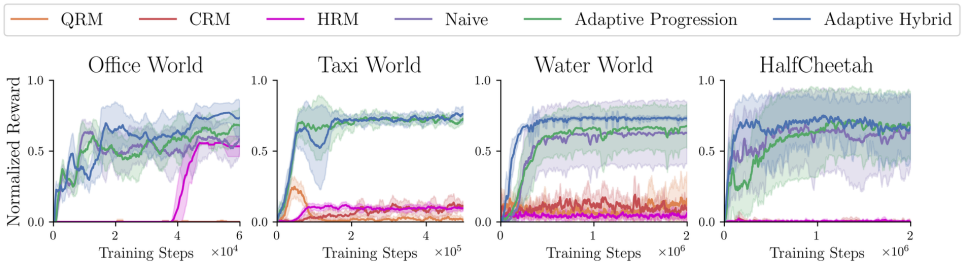


Figure 4: Results for infeasible environments.

Ablation study. Additionally, we conduct an ablation study to investigate the sensitivity of the hyperparameters θ and N used for updating distance-to-acceptance values (cf. Section 4.2). Figure 5 shows the normalized reward for two infeasible environments: Taxi World and Water World. The results demonstrate that the proposed approach converges with a sufficiently large value of $\theta \in \{2,000, 5,000, 10,000\}$. Moreover, it takes more training steps to achieve policy convergence with larger values of N, indicating longer intervals between consecutive updates of reward functions. Figure 6 shows the success rates for the feasible version of Taxi World and Water World. These results suggest that feasible environments benefit from longer update intervals N, as the agent has more time to explore and gather experience before the reward function is modified.

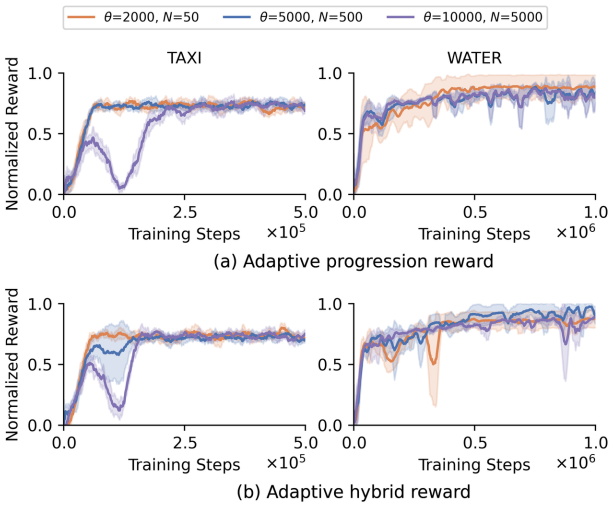


Figure 5: Results of the ablation study on the sensitivity of hyperparameters θ and N for updating distance-to-acceptance values in infeasible environments.

Hyperparameter Selection: Practical Guidance. We offer the following heuristics for selecting key hyperparameters in our framework. For the reward update interval N, a useful starting point is the total training budget divided by the number of distinct task stages (e.g., states in a task-governing DFA), as this aims to provide the agent with sufficient interaction episodes within each task stage before potential reward adjustments. The reward scaling factor θ can be initially set to the sum of progression rewards, $\sum_{q,q'} p_\phi(q, q')$, which approximates the cumulative effort. Insights from the ablation study further suggest that task feasibility can guide these choices: potentially infeasible tasks may benefit from smaller θ values and more frequent updates (smaller N) to enable regular progress assessment and dynamic reward adjustment in challenging settings. In contrast, feasible tasks often accommodate larger θ and N to allow for the collection of more meaningful data before reward function modification. When employing hybrid reward functions, we recommend small magnitudes for step-wise penalties (e.g., 10^{-3} or 10^{-4}) to avoid overwhelming the positive shaping signals. These guidelines serve as practical starting points, though optimal settings can be task-dependent.

6 CONCLUSION

We have developed a logic-based adaptive reward shaping approach for RL. Our approach uses reward functions designed to incentivize an agent to complete a task specified by a co-safe LTL formula as much as possible, and dynamically updates these reward functions during the learning process. This dynamic reward shaping is beneficial for scenarios where environmental uncertainties can lead to task failure despite successful subtask progress.

Computational experiments demonstrate that our approach is applicable to various discrete and continuous RL domains and is compatible with a wide range of RL algorithms such as DQN, DDQN, DDPG, PPO, and A2C. Experimental results also show that the proposed approach generally outperforms state-of-the-art baselines, achieving faster convergence to a better policy with higher expected return and task completion rate.

There are several directions for future work. First, we will evaluate the proposed approach on a broader range of RL domains beyond the benchmarks used in our experiments. Second, we will explore extending the approach to multi-agent RL. Finally, we aim to apply the proposed approach to real-world RL tasks, such as autonomous driving.

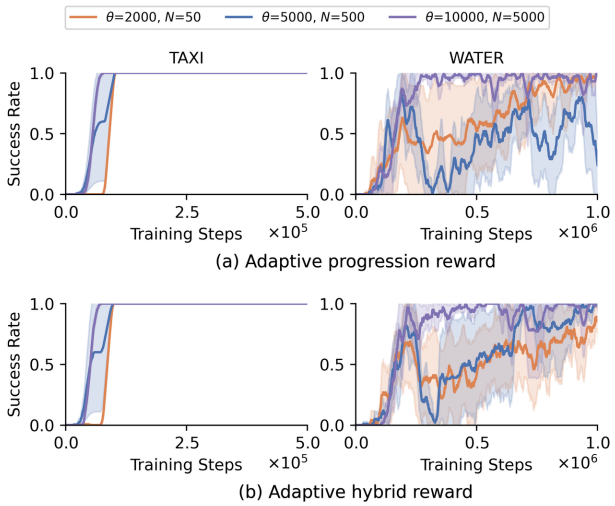


Figure 6: Results of the ablation study on the sensitivity of hyperparameters θ and N for updating distance-to-acceptance values in feasible environments.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under Grants CCF-1942836 and CCF-2131511. The opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agencies.

References

Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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. *IEEE Robotics and Automation Letters*, 2023a.

Mingyu Cai, Makai Mann, Zachary Serlin, Kevin Leahy, and Cristian-Ioan Vasile. Learning minimally-violating continuous control for infeasible linear temporal logic specifications. In *American Control Conference*, 2023b.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 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 International Conference on Automated Planning and Scheduling*, 2019.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*, 2018.

Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas, and Insup Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *International Confernece on Formal Modeling and Analysis of Timed Systems*, 2020.

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*, 2022.

Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 2019.

Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 2021.

Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal Methods in System Design*, 2001.

Bruno Lacerda, Fatma Faruq, David Parker, and Nick Hawes. Probabilistic planning with formal performance guarantees for mobile service robots. *The International Journal of Robotics Research*, 2019.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

Andrew Y Ng, Daishi Harada, and Stuart J Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 1981.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

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

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 2017.

Supplementary Material

Minjae Kwon[1] **Ingy ElSayed-Aly**[1] **Lu Feng**[1]

[1]The Department of Computer Science, University of Virginia, Charlottesville, VA 22904, USA

A CORRECTNESS

Here, we prove the correctness of our approach, as stated in Theorem 1. We start by proving the following auxiliary lemmas.

Lemma 1. *Adaptive hybrid reward function $R^{\otimes}_{\mathsf{ah},k}$ tends to adaptive progression reward function $R^{\otimes}_{\mathsf{ap},k}$ with an increasing number of updates k, that is,* $\lim_{k\to\infty} R^{\otimes}_{\mathsf{ah},k} = R^{\otimes}_{\mathsf{ap},k}$.

Proof. By the definition of adaptive hybrid reward function $R^{\otimes}_{\mathsf{ah},k}$ (cf. Equation 8), $\eta_0 \in [0,1]$ and $\eta_k = \frac{\eta_{k-1}}{\theta}$ with $\theta > 1$. We have $\lim_{k\to\infty} \eta_k = 0$. The first case of Equation 8, $\eta_k \cdot -d^k_\varphi(q)$, tends to 0; and the second case tends to $\max\{\rho^0_\varphi(q,q'), \rho^k_\varphi(q,q')\}$. Thus, it holds that $\lim_{k\to\infty} R^{\otimes}_{\mathsf{ah},k} = R^{\otimes}_{\mathsf{ap},k}$. □

Lemma 2. *Given an episodic MDP $\mathcal{M}$ and a DFA $\mathcal{A}_\varphi$ for a co-safe LTL formula φ, let π^*_k and π^*_{k+1} denote the optimal policies of the product MDP $\mathcal{M}^{\otimes} = \mathcal{M} \otimes \mathcal{A}_\varphi$, maximizing the expected return based on adaptive progression reward functions $R^{\otimes}_{\mathsf{ap},k}$ and $R^{\otimes}_{\mathsf{ap},k+1}$, respectively. If a policy exists that achieves a higher expected return than π^*_k based on $R^{\otimes}_{\mathsf{ap},k+1}$, then π^*_{k+1} achieves better task progression than π^*_k, that is, $b(\pi^*_{k+1}) < b(\pi^*_k)$.*

Proof. For the sake of contradiction, suppose that $b(\pi^*_{k+1}) \geq b(\pi^*_k)$. Let τ be a path through the product MDP $\mathcal{M}^{\otimes}$ under policy π^*_{k+1}. For any state $\langle s, q\rangle$ in the path τ, we have $q \in B_i$ where $i \geq b(\pi^*_{k+1}) \geq b(\pi^*_k) = b_k$. For every transition $(\langle s,q\rangle, a, \langle s',q'\rangle) \in \tau$, it holds that:

$$
\begin{aligned}
&R^{\otimes}_{\mathsf{ap},k+1}\left(\langle s,q\rangle, a, \langle s',q'\rangle\right)\\
=&\max\{\rho^0_\varphi(q,q'), \rho^{k+1}_\varphi(q,q')\}\\
=&\max\{\rho^0_\varphi(q,q'), \max\{0, d^{k+1}_\varphi(q) - d^{k+1}_\varphi(q')\}\}\\
=&\max\{\rho^0_\varphi(q,q'), \max\{0, d^{k}_\varphi(q) + \theta - d^{k}_\varphi(q') - \theta\}\}\\
=&\max\{\rho^0_\varphi(q,q'), \max\{0, d^{k}_\varphi(q) - d^{k}_\varphi(q')\}\}\\
=&\max\{\rho^0_\varphi(q,q'), \rho^{k}_\varphi(q,q')\}\\
=&R^{\otimes}_{\mathsf{ap},k}\left(\langle s,q\rangle, a, \langle s',q'\rangle\right)
\end{aligned}
$$

Thus, we have $V^{\pi^*_{k+1}}_{\mathsf{ap},k+1}(s^{\otimes}_0) = V^{\pi^*_{k+1}}_{\mathsf{ap},k}(s^{\otimes}_0)$, meaning that the expected return stays the same. Similarly, we can show that $V^{\pi^*_k}_{\mathsf{ap},k+1}(s^{\otimes}_0) = V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0)$.

Since π^*_k is the optimal policy maximizing the expected return based on $R^{\otimes}_{\mathsf{ap},k}$, we have

$$
V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0) \geq V^{\pi^*_{k+1}}_{\mathsf{ap},k}(s^{\otimes}_0) = V^{\pi^*_{k+1}}_{\mathsf{ap},k+1}(s^{\otimes}_0). \tag{9}
$$

Given that there exists a policy that achieves a higher expected return than π^*_k based on $R^{\otimes}_{\mathsf{ap},k+1}$, it holds that

$$V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0) = V^{\pi^*_k}_{\mathsf{ap},k+1}(s^{\otimes}_0) < V^{\pi^*_{k+1}}_{\mathsf{ap},k+1}(s^{\otimes}_0). \tag{10}$$

Equation 9 contradicts with Equation 10. Thus, we have $b(\pi^*_{k+1}) < b(\pi^*_k)$. □

Now we are ready to prove Theorem 1 as stated in Section 4 and repeated here.

Theorem 1. *Given an episodic MDP $\mathcal{M}$ and a DFA $\mathcal{A}_\varphi$ corresponding to a co-safe LTL formula φ, there exists an optimal policy π^* of the product MDP $\mathcal{M}^{\otimes} = \mathcal{M} \otimes \mathcal{A}_\varphi$ that maximizes the expected return based on a reward function $R^{\otimes} \in \{R^{\otimes}_{\mathsf{ap},k}, R^{\otimes}_{\mathsf{ah},k}\}$ for some $k \in \mathbb{N}$, where the task progression for policy π^* matches the best possible task progression b^* across all feasible policies in the product MDP $\mathcal{M}^{\otimes}$, that is, $b^* = b(\pi^*)$.*

Proof. Without loss of generality, we focus on the adaptive progression reward function $R^{\otimes}_{\mathsf{ap},k}$, as Lemma 1 shows that $\lim_{k\to\infty} R^{\otimes}_{\mathsf{ah},k} = R^{\otimes}_{\mathsf{ap},k}$.

Let π^*_k denote an optimal policy of the product MDP $\mathcal{M}^{\otimes}$ that maximizes the expected return based on $R^{\otimes}_{\mathsf{ap},k}$. Suppose that $b(\pi^*_k) > b^*$. There exists a policy π in the product MDP that achieves the best possible task progression b^*, where $V^{\pi}_{\mathsf{ap},k}(s^{\otimes}_0) \leq V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0)$. If $V^{\pi}_{\mathsf{ap},k}(s^{\otimes}_0) = V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0)$, then π is the desired optimal policy π^* that maximizes the expected return based on $R^{\otimes}_{\mathsf{ap},k}$while achieving the best possible task progression b^*. This theorem is thus proved.

Otherwise, when $V^{\pi}_{\mathsf{ap},k}(s^{\otimes}_0) < V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0)$, we proceed to prove the theorem as follows. Let the difference in expected returns be denoted by $\sigma = V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0) - V^{\pi}_{\mathsf{ap},k}(s^{\otimes}_0) > 0$. Consider a worst-case scenario where policy π reaches a state with the best possible task progression only at the end of an episode. Formally, there is only one path τ of length $|\tau| = H$ through the product MDP $\mathcal{M}^{\otimes}$ under policy π that ends with a transition $(\langle s, q\rangle, a, \langle s', q'\rangle)$ where $q \in B_i$, $q' \in B_j$, and $i > j = b^*$. Based on the definition of adaptive progression reward function, we have $R^{\otimes}_{\mathsf{ap},k+1}(\langle s, q\rangle, a, \langle s', q'\rangle) = R^{\otimes}_{\mathsf{ap},k}(\langle s, q\rangle, a, \langle s', q'\rangle) + \theta$. Thus, $V^{\pi}_{\mathsf{ap},k+1}(s^{\otimes}_0) = V^{\pi}_{\mathsf{ap},k}(s^{\otimes}_0) + p \cdot \gamma^{H-1} \cdot \theta$, where p is the probability of path τ and γ is the MDP's discount factor. Following the argument in Lemma 2, it holds that $V^{\pi^*_k}_{\mathsf{ap},k+1}(s^{\otimes}_0) = V^{\pi^*_k}_{\mathsf{ap},k}(s^{\otimes}_0)$. When the hyperparameter value θ is sufficiently large, more precisely, $\theta > \frac{\sigma}{p\cdot\gamma^{H-1}}$, we have $V^{\pi}_{\mathsf{ap},k+1}(s^{\otimes}_0) > V^{\pi^*_k}_{\mathsf{ap},k+1}(s^{\otimes}_0)$. Let π^*_{k+1} denote an optimal policy of the product MDP $\mathcal{M}^{\otimes}$ that maximizes the expected return based on $R^{\otimes}_{\mathsf{ap},k+1}$. If $V^{\pi}_{\mathsf{ap},k+1}(s^{\otimes}_0) = V^{\pi^*_{k+1}}_{\mathsf{ap},k+1}(s^{\otimes}_0)$, then π is the desired optimal policy π^* and the theorem is thus proved. Otherwise, following Lemma 2, it holds that $b(\pi^*_{k+1}) < b(\pi^*_k)$, meaning that the task progression for π^*_{k+1} has improved compared to that of policy π^*_k. Since a task progression value is bounded by the state partition size of DFA $\mathcal{A}_\varphi$, it takes only a finite number of updates before an optimal policy yielding b^* is learned.

In conclusion, there exists an optimal policy π^* for the product MDP $\mathcal{M}^{\otimes}$ that achieves the best possible task progression b^* while maximizing the expected return based on $R^{\otimes}_{\mathsf{ap},k}$ for some $k \in \mathbb{N}$, which is an adaptive progression reward function updated in a finite number of rounds with a sufficiently large hyperparameter value θ. □

B COMPATIBILITY WITH ON-POLICY LEARNING

Results for HalfCheetah. Figure 7 shows the results of applying three different RL algorithms, DDPG [Lillicrap et al., 2016], PPO [Schulman et al., 2017], and A2C [Mnih et al., 2016], to HalfCheetah environments. The comparison between the proposed approach and all baselines using DDPG has already been discussed in Section 5. Since the QRM, CRM, and HRM baselines are not compatible with PPO and A2C, we only compare with the naive baseline here.

Comparing the results of the three RL algorithms, we observe that DDPG exhibits relatively higher variance than the others. This is likely due to its off-policy nature, relying heavily on a replay buffer and exploration driven by control noise. In our experiments, we used a replay buffer with a capacity of 10^6 while sampling only 100 experiences for each update, introducing significant randomness as most samples in the large replay buffer do not yield positive rewards. Exploration also adds to the randomness. In contrast, PPO and A2C are on-policy algorithms, where updates depend solely on the current policy. These algorithms tend to maintain their behavior once the current policy achieves partial task completion. Additionally, PPO incorporates a stabilizing technique that helps reduce variance.

Comparing different reward functions, we find that the Naive baseline achieves comparable performance with the proposed reward functions in all HalfCheetah environments. However, as noted in Section 5, it usually performs the worst in other domains. One possible explanation is that the HalfCheetah task has a unique structure, where each sub-goal requires moving forward by the same distance. The Naive reward function assigns a reward of 1 for each sub-goal, maintaining consistency in the learning process.

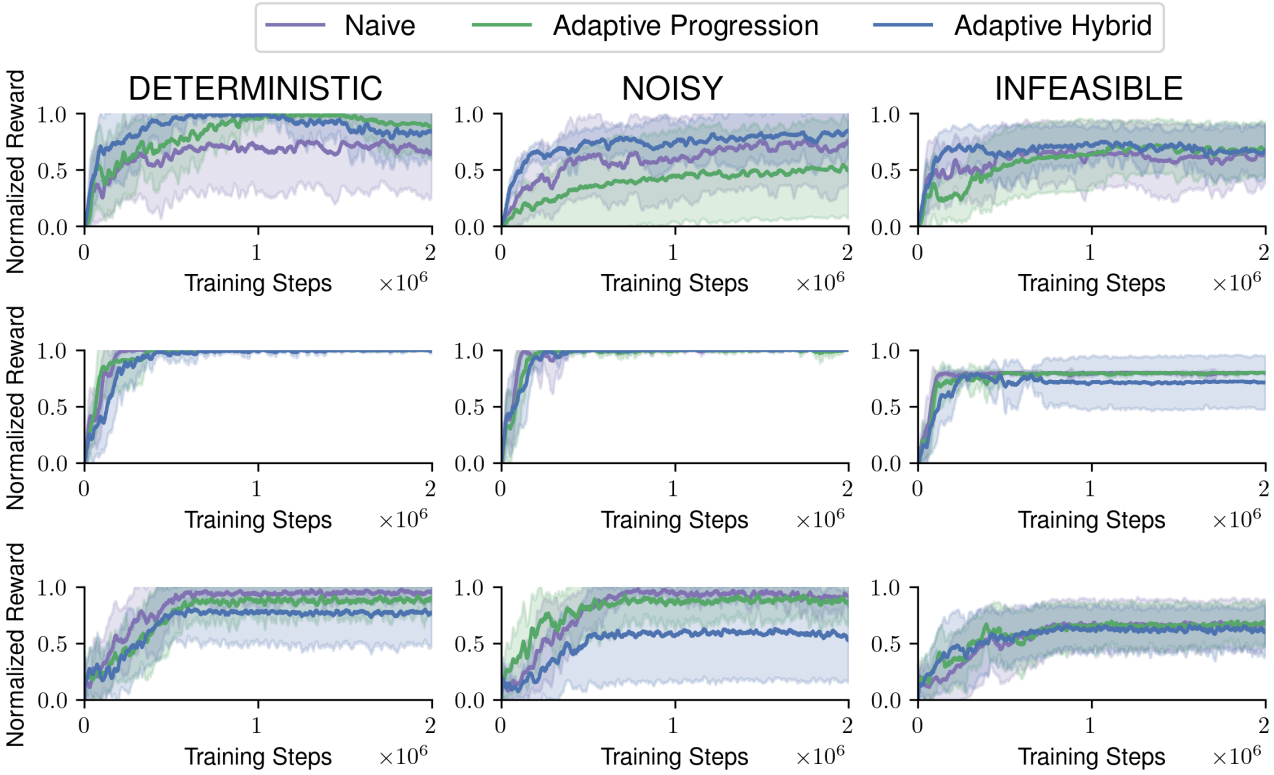


Figure 7: Results of applying various RL algorithms to HalfCheetah environments.

C PARTIAL REWARDS IN REWARD MACHINES

This ablation study investigates the effect of incorporating partial rewards into Reward Machine (RM) structures along with the use of potential-based reward shaping. While RMs are theoretically capable of representing and utilizing partial rewards (e.g., in the Office World environment, the RM transitions through states u_0 [initial], u_1, u_2 [intermediate], and u_3 [goal], as depicted in Figure 2 (b) Icarte et al. [2022], our empirical evaluation reveals that their inclusion does not consistently enhance performance and can lead to performance degradation.

To evaluate the impact of partial rewards on RM-based algorithms, we conducted experiments in deterministic environments: Office World and Taxi World. We define "Partial Reward Q-learning Reward Machine" (PR QRM) as the QRM algorithm variant that incorporates partial rewards. For consistency across algorithms, we similarly introduce PR CRM and PR HRM, denoting CRM and HRM variants also utilizing partial rewards. All baselines in this study are equipped with potential-based reward shaping. Across both Office World and Taxi World environments, all RM-based algorithms suffered a performance degradation when supplemented with partial rewards of 1 for each intermediate step.

These findings suggest that the algorithms, particularly in their current configurations, may not be inherently designed to effectively leverage partial rewards in conjunction with potential reward shaping. One plausible explanation for the observed performance degradation is that, as discussed in Icarte et al. [2022], potential reward shaping can assign positive rewards to actions that lead to undesirable "violation" states within the RM, potentially exacerbating the negative effects of partial rewards. Therefore, careful consideration and potentially algorithm modifications are necessary to effectively harness the benefits of partial rewards within Reward Machine frameworks, especially when integrated with reward shaping techniques. In contrast to these observations, our proposed algorithms are designed to effectively incorporate partial rewards across diverse environments without performance degradation, while ensuring both task completion and reward maximization.

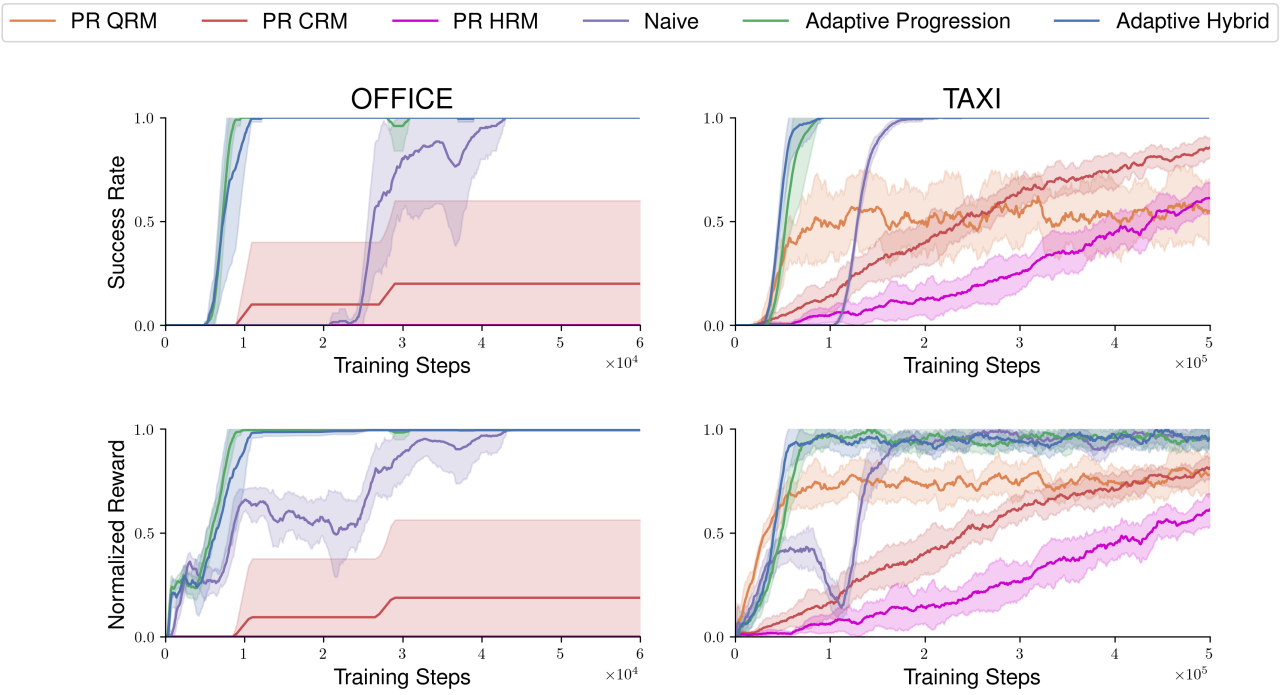


Figure 8: Results in Office World and Taxi World for RM algorithms using partial rewards.