
One Subgoal at a Time: Zero-Shot Generalization to Arbitrary Linear Temporal Logic Requirements in Multi-Task Reinforcement Learning

Zijian Guo¹, İlker Işık², H. M. Sabbir Ahmad¹, Wenchao Li^{1,2}

¹Division of Systems Engineering, Boston University

²Department of Electrical and Computer Engineering, Boston University
{zjguo, iilker, sabbir92, wenchao}@bu.edu

Abstract

Generalizing to complex and temporally extended task objectives and safety constraints remains a critical challenge in reinforcement learning (RL). Linear temporal logic (LTL) offers a unified formalism to specify such requirements, yet existing methods are limited in their abilities to handle nested long-horizon tasks and safety constraints, and cannot identify situations when a subgoal is not satisfiable and an alternative should be sought. In this paper, we introduce GenZ-LTL, a method that enables zero-shot generalization to arbitrary LTL specifications. GenZ-LTL leverages the structure of Büchi automata to decompose an LTL task specification into sequences of reach-avoid subgoals. Contrary to the current state-of-the-art method that conditions on subgoal sequences, we show that it is more effective to achieve zero-shot generalization by solving these reach-avoid problems *one subgoal at a time* through proper safe RL formulations. In addition, we introduce a novel subgoal-induced observation reduction technique that can mitigate the exponential complexity of subgoal-state combinations under realistic assumptions. Empirical results show that GenZ-LTL substantially outperforms existing methods in zero-shot generalization to unseen LTL specifications. The code is available at <https://github.com/BU-DEPEND-Lab/GenZ-LTL>.

1 Introduction

Generalization is a critical aspect of reinforcement learning (RL), which aims to develop policies that are capable of adapting to novel and unseen tasks [38, 48, 21] and varying safety constraints [7, 17]. Various methods have been proposed and show promising results in applications such as autonomous driving [49, 36], robotics [27, 33, 28], and healthcare [8, 41]. However, most existing approaches fall short in handling complex and temporally extended task objectives and safety constraints. For example, an autonomous vehicle may be instructed to reach a sequence of destinations for passenger pick-up and drop-off while adhering to traffic rules such as speed limits and yielding at intersections.

Temporal logic provides a principled framework for specifying system behaviors over time. Linear temporal logic (LTL) [39], in particular, has been considered extensively in single-task and multi-task RL settings [20, 29, 51]. LTL has well-defined semantics and a compositional syntax, which facilitates the unambiguous specification of task objectives and safety constraints. However, existing methods often fail to account for the potential conflicting objectives within a specification [50, 40, 23]. For example, an autonomous vehicle navigating between multiple destinations aims to minimize travel time, but adhering to speed limits and other safety constraints may slow the vehicle down. In such cases, the vehicle must prioritize satisfying safety requirements over optimizing for travel time.

In this paper, we propose a novel framework, called GenZ-LTL, for learning RL policies that generalize in a zero-shot manner to satisfy arbitrary LTL specifications, including both finite-horizon and infinite-horizon tasks. We adopt the subgoal decomposition approach, which exploits the structure of the equivalent Büchi automaton to decompose an LTL specification into individual subgoals, each comprising a *reach* component that defines task progression and an *avoid* component that encodes safety constraints. We show that training a goal-conditioned policy to achieve the subgoals *one at a time*, which is often perceived as myopic by existing literature, is more effective in general than conditioning the policy on subgoal sequences [23] or the entire automaton [54]. This somewhat surprising result stems from two key innovations in this paper: a proper safe RL formulation with *state-wise* constraints for solving the individual reach-avoid subgoals, and a novel *subgoal-induced observation reduction* technique to address the exponential complexity of observation-subgoal combinations. The former builds on recent advances of Hamilton-Jacobi reachability in RL [12] whereas the latter exploits symmetry in the state observations to extract information only relevant to the immediate subgoal. To zero-shot generalize to arbitrary LTL specifications at test time, we use the equivalent Büchi automata as well as the learned value functions to guide the subgoal selection. Our main contributions are summarized as follows.

- We propose a novel approach for learning RL policies that can satisfy arbitrary LTL specifications in a zero-shot manner by sequentially completing individual subgoals. To our knowledge, this is the first safe RL formulation with state-wise constraints for LTL task satisfaction.
- We introduce a subgoal-induced observation reduction technique to mitigate the combinatorial explosion of observation-subgoal pairs and enable efficient generalization, along with a timeout mechanism that handles subgoal switching and unsatisfiable subgoals.
- We conduct comprehensive experiments across a set of navigation-style environments and diverse task specifications, demonstrating that our method consistently outperforms state-of-the-art baselines in zero-shot generalization to arbitrary LTL specifications.

2 Related Work

Specification-guided and goal-conditioned RL. Various methods have been proposed to guide RL agents by extending traditional goal-conditioned RL [35] to temporal-logic goals. Early works focused on learning policies for a single and fixed specification, including methods based on LTL specifications and automaton-like models [20, 18, 51, 46, 45, 3], and methods leveraging quantitative semantics of specifications [32, 26, 44, 16]. More recently, the importance of adapting to changing conditions in real-world applications has brought increased attention to zero-shot generalization to new specifications. Existing approaches can be broadly categorized into subgoal decomposition [30, 31, 34, 40, 23] and direct specification/automata encoding [50, 54, 55]. Our work follows the direction of subgoal decomposition, but shows that it is more effective to solve the subgoals one at a time through proper safe RL formulations and subgoal-induced observation reduction.

RL with safety constraints. Various methods have been proposed to solve RL problems with safety constraints, such as policy optimization-based methods [59, 56, 24, 53], Gaussian processes-based approaches [52, 60], and control theory-based methods, including Hamilton-Jacobian (HJ) reachability [10, 57, 47, 13, 61] and control barrier functions [14, 6]. We point the readers to [15] for a comprehensive survey of safe RL techniques. In the context of formal specification-guided RL, the interplay between task progression and safety constraints remains underexplored. Most existing methods simply assign a positive reward when a specification or a subgoal is satisfied, and a negative reward at episode termination if unsatisfied. This approach is ineffective in achieving safety [1, 9, 4], especially when the task progression objective and the safety requirements exert conflicting influences on policy learning. In this paper, we show how to prioritize safety satisfaction properly by incorporating Hamilton-Jacobi reachability constraints.

3 Preliminaries

Reinforcement learning. RL agents learn policies by interacting with environments, which is usually modeled as a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \gamma, d_0)$, where \mathcal{S} is the state space and \mathcal{A} is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ defines the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor, and $d_0 \in \Delta(\mathcal{S})$ is the initial state distribution. Let $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ denote a policy and $\tau = \{s_t, a_t, r_t\}_{t=0}^{\infty}$ denote a trajectory

where $r_t = r(s_t, a_t)$. The value function under policy π is $V_r^\pi(s) = \mathbb{E}_{\tau \sim \pi, s_0=s} [\sum_{t=0}^{\infty} \gamma^t r_t]$ and the corresponding state-action value function is $Q_r^\pi(s, a) = \mathbb{E}_{\tau \sim \pi, s_0=s, a_0=a} [\sum_{t=0}^{\infty} \gamma^t r_t]$. An additional constraint violation function $h : \mathcal{S} \mapsto \mathbb{R}$ can be used to model state-wise safety constraints, such that a state s is considered safe if $h(s) \leq 0$ and unsafe if $h(s) > 0$. The reachability value function $V_h^\pi(s)$ is defined as $V_h^\pi(s) := \max_{t \in \mathbb{N}} h(s_t)$, $s_0 = s, a \sim \pi$ that captures the worst-case constraint violation under policy π from state s . The largest feasible set contains the largest set of states from which the agent can reach a goal safely: $\mathcal{S}_f := \{s \mid \exists \pi, V_h^\pi(s) \leq 0\}$. To handle such constraints, Hamilton-Jacobi (HJ) reachability has been incorporated into RL [12] by simultaneously learning the reachability value function to estimate the feasible set and learning the policy:

$$\max_{\pi} \mathbb{E}_{s \sim d_0} [V_r^\pi(s) \cdot \mathbb{1}[s \in \mathcal{S}_f] - V_h^\pi(s) \cdot \mathbb{1}[s \notin \mathcal{S}_f]] \quad \text{s.t. } V_h^\pi(s) \leq 0, \forall s \in \mathcal{S}_f \cap \mathcal{S}_0$$

where $\mathcal{S}_0 := \{s \mid s \sim d_0\}$ is the set of initial states. Intuitively, the goal is to maximize expected return and ensure safety for initial states within the largest feasible set, while minimizing the reachability value function for states outside it, where constraint violations are inevitable.

Linear Temporal Logic. LTL [39] is a formal language for specifying temporal properties of a system. LTL formulas are built from a set of Boolean operators – negation (\neg), conjunction (\wedge), disjunction (\vee), and temporal operators – "until" (U), "eventually" (F), and "always" (G). Given a finite set of atomic propositions AP and $a \in AP$, the syntax of LTL is defined recursively as:

$$\varphi := a \mid \neg \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \text{F} \varphi \mid \text{G} \varphi \mid \varphi_1 \text{U} \varphi_2$$

Intuitively, the formula $\varphi_1 \text{U} \varphi_2$ holds if φ_1 is satisfied at all time steps prior to the first occurrence of φ_2 . The operator $\text{F} \varphi$ holds if φ is satisfied at some future time, while $\text{G} \varphi$ holds if φ is satisfied from the current step onward. To relate LTL to MDPs, we assume a known labeling function $L : \mathcal{S} \mapsto 2^{AP}$ that maps each state to the set of true atomic propositions. The satisfaction probability of φ under a policy π is $\Pr(\pi \models \varphi) = \mathbb{E}_{\tau \sim \pi} [\mathbb{1}[\tau \models \varphi]]$, where $\tau \models \varphi$ denotes that the trace of the trajectory, $\text{Tr}(\tau) = L(s_0)L(s_1)\dots$, satisfies the formula φ , and $\mathbb{1}$ is the indicator function. We are interested in training a policy that maximizes this probability for an arbitrary φ .

Büchi automata and subgoals. To enable algorithmic reasoning over LTL specifications, we use Büchi automata [5] as a formal representation of temporal-logic formulas. Given an LTL formula φ , the corresponding (non-deterministic) Büchi automaton is defined by a tuple $\mathcal{B}_\varphi := (\mathcal{Q}, \Sigma, \delta, \mathcal{F}, q_0)$ where \mathcal{Q} is the finite set of automaton states, $\Sigma = 2^{AP}$ the finite alphabet, $\delta \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$ the transition relation, $\mathcal{F} \subseteq \mathcal{Q}$ the set of accepting automaton states, and q_0 the initial state. An infinite word $\omega = \alpha_0 \alpha_1 \dots \in \Sigma^\omega$ induces a run $r = q_0 q_1 \dots$ over ω if for every $i \geq 0$, $(q_i, \alpha_i, q_{i+1}) \in \delta$. A run r is accepting if it visits an accepting state infinitely often. A product MDP \mathcal{M}^φ [11] synchronizes a given MDP and Büchi automaton with a new state space $\mathcal{S}^\varphi = \mathcal{S} \times \mathcal{Q}$ and a transition function $\mathcal{P}^\varphi((s', q') \mid (s, q), a)$ that equals $\mathcal{P}(s' \mid s, a)$ if $a \in \mathcal{A}$ and $(q, L(s), q') \in \delta$, and 0 otherwise. In practice, rather than explicitly building the product MDP, one can simply update the current automaton state q with the propositions $L(s)$ observed at each time step. Given a state q , a *reach* subgoal is defined as an $\alpha^+ \in \Sigma$ such that $(q, \alpha^+, q') \in \delta$ and q and q' are consecutive automaton states along some accepting run, and $q' \neq q$ if $q \notin \mathcal{F}$. Conversely, an *avoid* subgoal is defined as an $\alpha^- \in \Sigma$ such that $(q, \alpha^-, q') \in \delta$ and q and q' are not consecutive automaton states along any accepting run. Intuitively, a reach subgoal indicates progression towards satisfying the LTL specification, and an avoid subgoal indicates a way that leads to violation of the specification.

4 Method

In this section, we present our proposed method, as illustrated in Figure 1. During training, we begin by sampling subgoals $\sigma = (\alpha^+, A^-)$, where α^+ is a single reach subgoal and A^- is a (possibly empty) set of avoid subgoals (and $\alpha^+ \notin A^-$). Given the current state s and subgoal σ , we apply subgoal-induced observation reduction (details in Section 4.2), when applicable, to obtain a processed state s^σ . We then train a general reach-avoid subgoal-conditioned policy along with its corresponding value functions. During testing, given a target LTL task specification φ , we first construct an equivalent Büchi automaton, and then extract a set of reach-avoid subgoals $\{\sigma_i\}_{i=1}^n$ based on the current MDP and automaton state. For each candidate subgoal σ_i , we compute s^{σ_i} and evaluate the value functions to select the optimal subgoal σ^* . The agent then generates actions a according to $\pi(a \mid s^{\sigma^*})$. In Section 4.1, we describe how to construct reach-avoid subgoals from a Büchi automaton. In Section 4.2, we introduce a subgoal-induced observation reduction technique to

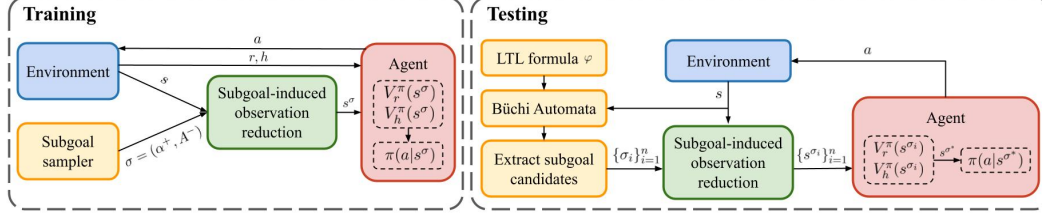


Figure 1: Overview of GenZ-LTL. During training, we sample reach-avoid subgoals $\sigma = (\alpha^+, A^-)$ and apply a subgoal-induced observation reduction to learn a subgoal-conditioned policy and value functions. At test time, given a target LTL formula φ , we construct the corresponding Büchi automaton and identify candidate subgoals based on the current automaton state. The optimal subgoal σ^* is selected using the learned value functions, and the policy generates actions conditioned on σ^* .

mitigate the exponential complexity arising from the combination of subgoals and MDP states. In Section 4.3, we detail the learning of value functions and policies under reachability constraints to comply with the safety constraints (A^-) while optimizing for the task progression objective (α^+).

4.1 Reach-Avoid Subgoal Construction from Büchi Automaton

Büchi automata can be represented as directed state-transition graphs [40], and an accepted run can be viewed as having a lasso structure with a cycle containing an accepting state and a prefix path that leads to the cycle. Similar to existing approaches [40, 23], we use depth-first search (DFS) over the state-transition graph to enumerate all possible lasso structures starting from a given state q . From these lasso paths, we extract the set of reach subgoals, denoted by A^+ , following the definition in Section 3. Similarly, we extract the set of avoid subgoals A^- by considering transitions to a state that is different from q and not along any of the lasso paths. Each pair $(\alpha^+ \in A^+, A^-)$ then becomes a candidate reach-avoid subgoal. To satisfy a subgoal (α^+, A^-) , the agent must visit a state s such that $L(s) = \alpha^+$, while avoiding any states labeled with propositions in A^- . For example, for $(A^+ = \{\{a, b\}, \{c\}\}, A^- = \{\{d\}, \{e\}\})$, it forms subgoals $\sigma_1 = (\{a \wedge b\}, A^-)$ and $\sigma_2 = (\{c\}, A^-)$, and the agent can choose to reach a state labeled with $a \wedge b$ or a state labeled with c , while avoiding states labeled with d and e in both cases. While our method follows a similar subgoal decomposition approach as prior works, the way that we solve the reach-avoid subgoals is significantly different. In fact, a straightforward way is to encode a reach-avoid subgoal as a bitvector of dimension $|AP| + 2^{|AP|}$, where the first $|AP|$ bits encodes the reach subgoal, and the rest of the bits encode the avoid subgoal.

4.2 Subgoal-Induced Observation Reduction

Given a reach-avoid subgoal, our next step is to train a subgoal-conditioned policy based on the MDP state or observation of the MDP state, and the subgoal. However, notice that the number of possible subgoals grows exponentially with the number of atomic propositions. Additionally, in order for the agent to complete a task in such a complex environment, it needs to be able to make observation of every state-subgoal combination, i.e. being able to determine the label $L(s)$ of a state s that it observes (possibly indirectly through an observation function o). Otherwise, task completion cannot be determined in general. For instance, if the task requires the agent to reach a ‘green’ region, then an agent equipped with only ‘blue’ sensors cannot learn to accomplish this task if the green and blue regions are placed randomly in the environment. Thus, the number of observations that the policy needs to condition on also grows exponentially with $|AP|$, and directly learning a goal-conditioned policy this way quickly becomes intractable as $|AP|$ grows. While various dimensionality reduction techniques have been proposed in the RL literature [2, 19, 58], they are based on abstraction or bisimulation of the MDP states, and do not consider state-subgoal combinations.

In this paper, we introduce a novel subgoal-induced observation reduction technique that can circumvent this exponential complexity. The key observation is that, ultimately, each subgoal reduces to a reach-avoid problem, where there is some state that we want to reach while avoiding some other states. In other words, what really matters is whether a state is ‘reach’ or ‘avoid’, and not its label $L(s)$ once a subgoal is fixed. Thus, we can perform a symbolic simplification of the state/observation space

along with the subgoal while preserving information essential for solving the reach-avoid problem. Formally, for an MDP state s , we assume that it can be partitioned into a *proposition-independent* component $s_{\neg AP}$ and a *proposition-dependent* component s_{AP} (e.g., the blue and green sensors). For the proposition-dependent component, we further assume that it can be broken down into individual observations each corresponding to an element in 2^{AP} , and the observations are produced by *identical observation functions under output transformation*. We say that two observation functions (with a slight abuse of notation of s), $o_1(s) \in \mathbb{R}^k$ and $o_2(s) \in \mathbb{R}^k$ are identical under an output transformation g if $\forall s, o_1(s) = g(o_2(s))$. For instance, suppose $o_1(s) \in \mathbb{R}^{360}$ is a LiDAR sensor for sensing the shortest distance to a blue state along each of its 360 beams, and $o_2(s) \in \mathbb{R}^{360}$ is the same type of LiDAR sensor for sensing the shortest distance to a green state but rotated 180° , then g is the 180° rotation (a permutation of the beam indices). Then, we can reduce the proposition-dependent component (of size $k \times 2^{|AP|}$) and the encoding of a reach-avoid subgoal (of size $(|AP| + 2^{|AP|})$) into just two observations each of size k , where the first observation is an observation corresponding to the reach subgoal and the second observation is a *fused* observation of the observations corresponding to the avoid subgoals. For simplicity of deriving a fusion operator that preserves task-relevant semantics such as distances to avoid zones, we assume that the $o(s)$ are coordinate-wise monotonic. The actual fusion operator f is environment/sensor-specific – in the LiDAR example, as shown in Figure 2, it will be an elementwise min over \mathbb{R}^k . Implementation details are provided in Appendix B. Empirically, we demonstrate the benefits of this reduction, when it is applicable, in Sections 5.3 and Appendix C.

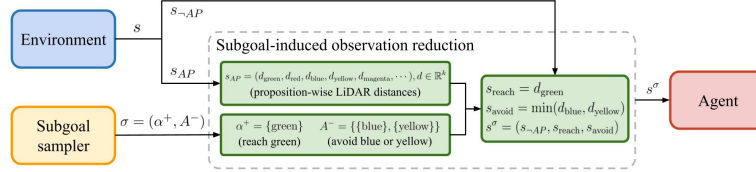


Figure 2: Illustration of the subgoal-induced observation reduction for LiDAR observations. The environment provides the raw state consisting of a proposition-independent component $s_{\neg AP}$ and a proposition-dependent component s_{AP} . Given a subgoal σ , s_{AP} is reduced to two vectors: a reach observation and a fused avoid observation. Together with the $s_{\neg AP}$, these form the agent’s input.

4.3 Goal-Conditioned Policy Learning with Reachability Constraints

A key difference between our approach and existing subgoal-based approaches lies in the treatment of the avoid subgoal. We treat an avoid subgoal as a *hard* constraint, since whenever a violation occurs, satisfying the LTL formula becomes impossible. For example, consider the specification $\neg a \cup b$, where a and b are atomic propositions; this requires the agent to avoid reaching any state labeled with a until a state labeled with b is reached. If a is encountered before b , the whole trajectory can no longer satisfy the specification. To enforce these hard constraints, we leverage the notion of HJ reachability as introduced in Section 3 and formulate our RL objective as follows:

$$\begin{aligned} & \max_{\pi} \mathbb{E}_{s \sim d_0, \sigma \sim \text{Unif}(\xi)} [V_r^\pi(s^\sigma) \cdot \mathbb{1}[s^\sigma \in \mathcal{S}_f^\sigma] - V_h^\pi(s^\sigma) \cdot \mathbb{1}[s^\sigma \notin \mathcal{S}_f^\sigma]] \\ & \text{s.t. } V_h^\pi(s^\sigma) \leq 0, \forall s^\sigma \in \mathcal{S}_f^\sigma \cap \mathcal{S}_0^\sigma, \forall \sigma \sim \text{Unif}(\xi) \end{aligned} \quad (1)$$

where ξ is the set of all possible subgoals (details on its construction are provided in Appendix B), $s^\sigma := f(s, \sigma)$ is the processed state if the observation reduction is applied, and $s^\sigma := (s, \sigma)$ otherwise, $\mathcal{S}_f^\sigma := \{s^\sigma \mid \sigma, \exists \pi, V_h^\pi(s^\sigma) \leq 0\}$ denotes the largest subgoal-dependent feasible set, $\mathcal{S}_0^\sigma := \{s^\sigma \mid \sigma, s \sim d_0\}$ denotes the set of processed initial states given a subgoal σ , and $V_h^\pi(s^\sigma)$ is the reachability value function of a policy π starting from a processed state s^σ with the current subgoal σ . We use π as a shorthand for the policy $\pi(a \mid s^\sigma)$. However, since \mathcal{S}_f^σ is unknown when learning policies, directly solving Eq. (1) is not possible. Inspired by previous methods [1, 57], we convert the objective into the following constrained policy optimization problem:

$$\begin{aligned} & \pi_{k+1} = \arg \max_{\pi} \mathbb{E}_{\sigma \sim \text{Unif}(\xi), s \sim d^{\pi_k}, a \sim \pi_k} \left[\frac{\pi}{\pi_k} A_r^{\pi_k}(s^\sigma, a) \right] \\ & \text{s.t. } \mathbb{E}_{\sigma \sim \text{Unif}(\xi), s \sim d^{\pi_k}} [\mathcal{D}_{KL}(\pi, \pi_k)] \leq \epsilon, \\ & \mathbb{E}_{\sigma \sim \text{Unif}(\xi), s \sim d^{\pi_k}, a \sim \pi_k} \left[(1 - \gamma) J_h(\pi_k) + \frac{\pi}{\pi_k} A_h^{\pi_k}(s^\sigma, a) \right] \leq 0 \end{aligned} \quad (2)$$

where $A_r^\pi(s^\sigma, a) := Q_r^\pi(s^\sigma, a) - V_r^\pi(s^\sigma)$ and $A_h^\pi(s^\sigma, a) := Q_h^\pi(s^\sigma, a) - V_h^\pi(s^\sigma)$ are the advantages functions with $Q_h^\pi(s^\sigma, a) := \max_{t \in \mathbb{N}} h(s_t^\sigma)$, $s_0 = s^\sigma$, $a_0 = a$, $a_t \sim \pi$ and $J_h(\pi) := \max_{t \in \mathbb{N}} h(\cdot)$, $a \sim \pi$ is the maximum safety violation for a trajectory under policy π . By maximizing the discounted cumulative reward, the policy is optimized to effectively satisfy the reach subgoal, while the safety constraints ensures that the avoid subgoals are satisfied. Then we incorporate the clipped surrogate objective [43] to handle the trust-region constraint and a *state-dependent* Lagrangian multiplier [57] to enforce the state-wise safety constraint:

$$\min_{\lambda \geq 0} \max_{\pi} \mathbb{E}_{\sigma \sim \text{Unif}(\xi), s \sim d^{\pi_k}, a \sim \pi} \left[\min \left(\frac{\pi}{\pi_k} A_r^{\pi_k}(s^\sigma, a), \text{clip} \left(\frac{\pi}{\pi_k}, 1 - \epsilon, 1 + \epsilon \right) A_r^{\pi_k}(s^\sigma, a) \right) - \lambda(s^\sigma) \left((1 - \gamma) J_h(s^\sigma) + \frac{\pi}{\pi_k} A_h^{\pi_k}(s^\sigma, a) \right) \right] \quad (3)$$

The overall method is summarized in Algorithm 1. The training process of our method is simple compared to previous methods that sample full LTL specifications [50, 40] or employ curriculum learning that gradually exposes the policy to more challenging reach-avoid sequences [23]. We sample one subgoal at a time, apply observation reduction to obtain a reduced state when applicable, and execute the policy to collect data. When the subgoal is satisfied, a new one is sampled, and this cycle continues until the episode terminates due to reaching the maximum length or a subgoal violation. During evaluation, given an LTL formula φ , we convert it to a Büchi automaton and track the current automaton state q . We identify the set of candidate subgoals $\{\sigma_i\}_{i=1}^n$ that can advance toward satisfying φ . The subgoal that achieves the best trade-off between task progression and satisfying safety constraint is selected as $\sigma^* = \arg \max_{\sigma} V_r(s^\sigma) - \lambda(s^\sigma) V_h(s^\sigma)$. The agent takes actions based on the selected subgoal and updates it upon each transition to a new automaton state.

Subgoal switching. In general, we cannot determine *a priori* whether a selected subgoal is satisfiable in the environment. To handle this issue, we introduce a mechanism to identify unsatisfiable subgoals and allow the agent to switch to alternative ones. The mechanism is triggered if the current subgoal is not satisfied within a timeout threshold, which is set to $(1 + \epsilon_{\text{scale}}) \cdot \mu_{\text{subgoal}}$, where μ_{subgoal} denotes the maximum number of steps required to complete a subgoal after policy convergence during training, and ϵ_{scale} is a scaling factor. When a reach subgoal times out, the best option from the remaining reach subgoals in A^+ is selected. This procedure is sound because the avoid subgoals are respected during the attempt for the prior reach subgoal(s). Appendix B.2 provides further details.

Remark. If there is no more subgoal remaining, our method returns failure. However, this does not necessarily mean that the specification is unsatisfiable. It is possible that the specification is satisfiable but the greedy nature of solving the subgoals sequentially prevents us from finding a solution. In general, unless the environment MDP is known, we cannot determine *a priori* if the LTL specification is satisfiable or not. If the agent is allowed to reset in the same environment, then one can enumerate the possible subgoal sequences from the initial state of the automaton.

5 Experiments

In this section, we evaluate our method GenZ-LTL in multiple environments and across a wide range of LTL specifications of varying complexity to answer the following questions: **Q1:** How well can GenZ-LTL zero-shot generalize to arbitrary LTL specifications? **Q2:** How does GenZ-LTL perform under increasing specification complexity and environment complexity? **Q3:** What are the individual impacts of the proposed observation reduction and safe RL approach? Additional evaluations and results in are provided in Appendix C.

Environments. Our environments include the LetterWorld environment [50], a 7×7 discrete grid world with discrete actions where letters are placed at randomly sampled positions, and the ZoneEnv environment [50], a high-dimensional environment with lidar observations, where a robotic agent with a continuous action space must navigate between randomly positioned colored regions. We exclude the FlatWorld environment [23, 51] from our experiments, as its configuration is fixed and the state space includes only the agent’s (x, y) -position, without any information about the environment state. We also consider a variant of ZoneEnv that contains overlapping regions which is by design more challenging than FlatWorld. Additional details are provided in Appendix A.1.

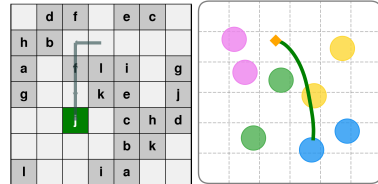


Figure 3: Environment illustrations of LetterWorld and ZoneEnv with example trajectories for $\neg l \cup j$ and $\neg \text{green} \cup \text{blue}$, respectively.

Baselines. We compare GenZ-LTL against the following baselines, all of which train a goal-conditioned policy of some sort: 1) LTL2Action [50], which applies a technique called LTL progression to dynamically track the parts of a specification that remain to be satisfied, and use a graph neural network to encode the specification; 2) GCRL-LTL [40], which uses a heuristic to identify the optimal subgoal sequence; 3) DeepLTL [23], which learns policies conditioned on embeddings of subgoal sequences; 4) RAD-embeddings [54], which pre-trains embeddings of compositions of deterministic finite automata (DFAs) and learns policies conditioned on these embeddings. While these baselines consider safety, as the LTL specification itself can involve the notion of safety, they do not model safety constraints explicitly in their RL formulation. Further details of the implementation are provided in Appendix B.

LTL specifications. We consider a wide variety of LTL specifications with varying complexity, including (1) sequential reach-only specifications of the form $F(\alpha_1 \wedge (F\alpha_2 \wedge (\dots \wedge F\alpha_n)))$; (2) sequential reach-avoid specifications of the form $\neg\alpha_1 \cup (\alpha_2 \wedge (\neg\alpha_3 \cup (\alpha_4 \wedge \dots \wedge (\neg\alpha_{n-1} \cup \alpha_n)))$, and (3) complex specifications that combine reach-only and reach-avoid properties. We also conduct a separate evaluation on infinite-horizon tasks that span a diverse set of LTL objectives, including safety, liveness, and their combinations. The specifications used for evaluation are listed in Table 1 and a full set of specifications is provided in Appendix A.2, with trajectory visualizations in Appendix D. In this section, all specifications are satisfiable.

Table 1: LTL specifications used for evaluation. For specifications in ZoneEnv, b, g, m, and y denote blue, green, magenta, and yellow, respectively. Results are shown in Table 2 and 3.

	LetterWorld	ZoneEnv
Finite-horizon	$\varphi_1 \ F(a \wedge (\neg b \cup c)) \wedge Fd$	$\varphi_9 \ (Fb) \wedge (\neg b \cup (g \wedge Fy))$
	$\varphi_2 \ (Fd) \wedge (\neg f \cup (d \wedge Fb))$	$\varphi_{10} \ \neg(m \vee y) \cup (b \wedge Fg)$
	$\varphi_3 \ \neg a \cup (b \wedge (\neg c \cup (d \wedge (\neg e \cup f))))$	$\varphi_{11} \ \neg g \cup ((b \vee m) \wedge (\neg g \cup y))$
	$\varphi_4 \ (a \vee b \vee c \vee d \Rightarrow F(e \wedge F(f \wedge Fg))) \cup (h \wedge Fi)$	$\varphi_{12} \ (g \vee b \Rightarrow (\neg y \cup m)) \cup y$
	$\varphi_5 \ F(d \wedge (\neg(a \vee b) \cup (b \wedge (\neg e \cup c)))) \wedge F(\neg(f \vee g \vee h) \cup a)$	$\varphi_{13} \ F(g \wedge (\neg(b \vee y) \cup (y \wedge (\neg m \cup b)))) \wedge F(\neg g \cup y)$
	$\varphi_6 \ F((k \wedge ((\neg b \vee c) \cup f)) \wedge (\neg(a \vee e \vee h) \cup g)) \wedge Fd$	$\varphi_{14} \ F((b \vee g) \wedge (\neg y \cup (b \wedge (\neg(g \vee m) \cup m)))) \wedge F(y \wedge (\neg b \cup g))$
	$\varphi_7 \ \neg(j \vee b \vee d) \cup (a \wedge (\neg c \cup (f \wedge F(g \wedge (\neg d \cup e)))))$	$\varphi_{15} \ \neg(m \vee y) \cup (b \wedge (\neg g \cup (y \wedge F(g \wedge (\neg b \cup m)))))$
	$\varphi_8 \ \neg(f \vee g) \cup (a \wedge (\neg b \cup c) \wedge F(d \wedge (\neg e \cup f)))$	$\varphi_{16} \ F(b \wedge (\neg y \cup (g \wedge F(y \wedge (\neg(m \vee g) \cup b)))))$
Infinite-horizon	$\psi_1 \ GF(e \wedge (\neg a \cup f)) \wedge G\neg(c \vee d)$	$\psi_4 \ GFb \wedge GFg \wedge G\neg(y \vee m)$
	$\psi_2 \ GFa \wedge GFb \wedge GFc \wedge G\neg(e \vee f \vee i)$	$\psi_5 \ GFb \wedge GFy \wedge GFg \wedge G\neg m$
	$\psi_3 \ GFc \wedge GFa \wedge GF(e \wedge (\neg f \cup g)) \wedge GFk \wedge G\neg(i \vee j)$	$\psi_6 \ FGy \wedge G\neg(g \vee b \vee m)$

Evaluation metrics. For fair comparisons, we follow the standard practice of executing a deterministic policy by taking the mean of the action distribution [22]. For each LTL specification, we report the following rates: success rate η_s , defined as the ratio of trajectories that satisfy the specification; violation rate η_v , defined as the ratio of trajectories that violate it; and other rate $\eta_o = 1 - (\eta_s + \eta_v)$. We also report the average number of steps μ to satisfy the specification among successful trajectories as a secondary metric, as step count is less important when the success rate is low.

5.1 Can GenZ-LTL zero-shot generalize to arbitrary LTL specifications (Q1)?

Finite-horizon tasks. We first evaluate finite-horizon tasks, i.e. tasks that can be satisfied with finite-length trajectories. All the specifications considered are complex LTL formulas with nested temporal and Boolean operators (details in Table 1). The evaluation results are shown in Table 2. Our method consistently outperforms the baselines in terms of both success and violation rates. LTL2Action encodes the full structure of the current LTL specification, making it less effective at adapting to out-of-distribution (OOD) specifications at test time. GCRL-LTL learns a goal-conditioned policy but does not model safety constraints explicitly, limiting its ability to enforce constraint satisfaction. DeepLTL relies on curriculum training which samples random reach-avoid subgoal sequences only up to a certain length. When evaluating specifications φ_9 – φ_{12} , although the specifications themselves are unseen, the underlying subgoal sequences are likely to have been encountered during training, leading to better efficiency, i.e., smaller μ . However, for specifications with longer subgoal sequences, e.g., φ_{13} – φ_{16} , which are OOD with respect to the training sequences, DeepLTL exhibits reduced performance. RAD-embeddings pre-trains the automaton embeddings by sampling from a fixed distribution. As a result, it also struggles with the issue of OOD generalization when faced with unseen

Table 2: Evaluation results for success rate η_s , violation rate η_v , and average steps μ to satisfy the complex specifications listed in Table 1. Specifications φ_1 – φ_8 are evaluated in LetterWorld, and φ_9 – φ_{16} in ZoneEnv. φ_1 – φ_4 and φ_9 – φ_{12} are from [23] and we construct more complex φ_5 – φ_8 and φ_{13} – φ_{16} . \uparrow : higher is better; \downarrow : lower is better. **Bold**: the best performance for each metric. Each value is averaged over 5 seeds, with 100 trajectories per seed.

	LTL2Action			GCRL-LTL			DeepLTL			RAD-embeddings			GenZ-LTL(ours)			
	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	
LetterWorld	φ_1	0.51 \pm 0.10	0.00\pm0.00	29.03 \pm 3.08	0.87 \pm 0.04	0.02 \pm 0.02	15.37 \pm 0.39	0.87 \pm 0.02	0.00\pm0.00	8.12 \pm 0.21	0.96 \pm 0.04	0.00\pm0.00	18.71 \pm 2.06	0.98\pm0.02	0.00\pm0.00	7.38\pm0.13
	φ_2	0.66 \pm 0.16	0.14 \pm 0.11	23.25 \pm 4.93	0.90 \pm 0.03	0.00\pm0.00	9.26 \pm 0.75	0.91 \pm 0.02	0.01 \pm 0.01	5.88 \pm 0.12	0.90 \pm 0.07	0.07 \pm 0.04	13.64 \pm 2.71	1.00\pm0.00	0.00\pm0.00	5.48\pm0.09
	φ_3	0.75 \pm 0.07	0.14 \pm 0.04	24.60 \pm 4.38	0.72 \pm 0.06	0.09 \pm 0.07	14.02 \pm 1.09	0.81 \pm 0.02	0.01 \pm 0.00	9.00 \pm 0.14	0.86 \pm 0.05	0.08 \pm 0.03	20.02 \pm 2.05	0.96\pm0.01	0.00\pm0.00	8.61\pm0.46
	φ_4	0.56 \pm 0.19	0.00\pm0.00	29.69 \pm 8.50	0.99\pm0.00	0.00\pm0.00	25.54 \pm 8.84	0.89 \pm 0.02	0.00\pm0.00	7.05\pm0.28	0.89 \pm 0.04	0.00\pm0.00	18.78 \pm 2.82	0.98 \pm 0.01	0.00\pm0.00	7.41 \pm 0.37
	Ave.	0.62 \pm 0.16	0.07 \pm 0.09	26.64 \pm 5.87	0.87 \pm 0.11	0.03 \pm 0.05	16.05 \pm 6.13	0.87 \pm 0.04	0.00\pm0.01	7.51 \pm 1.21	0.90 \pm 0.06	0.04 \pm 0.04	17.79 \pm 3.37	0.98\pm0.02	0.00\pm0.00	7.22\pm1.18
	φ_5	0.30 \pm 0.13	0.00\pm0.00	39.12 \pm 2.08	0.67 \pm 0.07	0.07 \pm 0.05	20.53 \pm 0.49	0.78 \pm 0.05	0.00\pm0.00	11.24 \pm 0.39	0.88 \pm 0.05	0.00\pm0.00	26.02 \pm 1.62	0.99\pm0.00	0.00\pm0.00	10.60\pm0.12
	φ_6	0.18 \pm 0.10	0.00\pm0.00	28.40 \pm 6.15	0.70 \pm 0.08	0.15 \pm 0.05	14.69 \pm 0.75	0.81 \pm 0.02	0.00\pm0.00	8.23 \pm 0.18	0.90 \pm 0.04	0.00\pm0.00	19.48 \pm 1.96	0.95\pm0.01	0.00\pm0.00	7.48\pm0.06
	φ_7	0.29 \pm 0.16	0.33 \pm 0.20	40.71 \pm 1.43	0.59 \pm 0.05	0.14 \pm 0.07	19.52 \pm 1.25	0.71 \pm 0.02	0.03 \pm 0.01	11.82 \pm 0.39	0.73 \pm 0.05	0.17 \pm 0.05	26.07 \pm 3.73	0.94\pm0.01	0.00\pm0.00	10.97\pm0.08
	φ_8	0.20 \pm 0.08	0.47 \pm 0.22	37.50 \pm 4.16	0.63 \pm 0.09	0.09 \pm 0.07	20.10 \pm 0.82	0.75 \pm 0.02	0.02 \pm 0.01	11.20 \pm 0.25	0.76 \pm 0.11	0.13 \pm 0.05	25.60 \pm 2.89	0.93\pm0.01	0.00\pm0.00	10.24\pm0.08
	Ave.	0.24 \pm 0.12	0.20 \pm 0.25	36.43 \pm 6.08	0.65 \pm 0.08	0.11 \pm 0.06	18.71 \pm 2.54	0.76 \pm 0.05	0.01 \pm 0.02	10.62 \pm 1.47	0.82 \pm 0.10	0.07 \pm 0.08	24.29 \pm 3.77	0.95\pm0.03	0.00\pm0.00	9.82\pm1.42
ZoneEnv	φ_9	0.13 \pm 0.08	0.18 \pm 0.25	496.09 \pm 128.52	0.88 \pm 0.05	0.03 \pm 0.00	474.41 \pm 23.42	0.89 \pm 0.03	0.04 \pm 0.00	327.80\pm23.78	0.90 \pm 0.07	0.08 \pm 0.07	423.11 \pm 51.08	0.99\pm0.01	0.01\pm0.01	380.15 \pm 5.17
	φ_{10}	0.39 \pm 0.32	0.38 \pm 0.24	421.46 \pm 58.76	0.88 \pm 0.03	0.06 \pm 0.01	303.43 \pm 4.46	0.88 \pm 0.03	0.09 \pm 0.01	221.94\pm23.26	0.92 \pm 0.02	0.06 \pm 0.02	301.21 \pm 101.99	0.97\pm0.01	0.02\pm0.02	253.44 \pm 10.22
	φ_{11}	0.82 \pm 0.15	0.08 \pm 0.03	284.41 \pm 95.62	0.86 \pm 0.04	0.06 \pm 0.03	306.94 \pm 11.26	0.91 \pm 0.06	0.04 \pm 0.02	215.42\pm15.01	0.93 \pm 0.06	0.04 \pm 0.02	247.83 \pm 42.98	0.99\pm0.01	0.01\pm0.01	249.57 \pm 11.85
	φ_{12}	0.94 \pm 0.03	0.02 \pm 0.03	122.89 \pm 20.97	0.90 \pm 0.01	0.05 \pm 0.02	136.32 \pm 10.27	0.97 \pm 0.01	0.01 \pm 0.00	116.40 \pm 13.07	1.00\pm0.01	0.00\pm0.00	105.97\pm17.31	1.00\pm0.00	0.00\pm0.00	135.60 \pm 8.06
	Ave.	0.57 \pm 0.37	0.17 \pm 0.21	331.21 \pm 165.88	0.88 \pm 0.04	0.05 \pm 0.02	305.28 \pm 123.33	0.91 \pm 0.05	0.04 \pm 0.03	220.39\pm78.77	0.94 \pm 0.05	0.04 \pm 0.05	269.53 \pm 129.95	0.99\pm0.01	0.01\pm0.01	254.69 \pm 8.18
	φ_{13}	0.17 \pm 0.23	0.01 \pm 0.01	886.66 \pm 104.03	0.73 \pm 0.06	0.09 \pm 0.02	618.42 \pm 22.80	0.91 \pm 0.03	0.00\pm0.00	508.26 \pm 54.81	0.69 \pm 0.17	0.00\pm0.00	664.59 \pm 29.91	0.98\pm0.02	0.00\pm0.00	436.25\pm18.35
	φ_{14}	0.36 \pm 0.29	0.01 \pm 0.01	885.13 \pm 148.93	0.68 \pm 0.06	0.11 \pm 0.06	611.24 \pm 38.37	0.91 \pm 0.03	0.00\pm0.00	473.45 \pm 52.67	0.77 \pm 0.21	0.00\pm0.00	628.57 \pm 22.89	0.98\pm0.01	0.00\pm0.00	434.45\pm14.61
	φ_{15}	0.08 \pm 0.05	0.49 \pm 0.30	855.56 \pm 311.81	0.70 \pm 0.06	0.10 \pm 0.03	599.71 \pm 18.62	0.78 \pm 0.05	0.12 \pm 0.05	523.50 \pm 64.73	0.62 \pm 0.09	0.02 \pm 0.01	680.11 \pm 9.51	0.97\pm0.01	0.01\pm0.00	388.01\pm15.47
	φ_{16}	0.03 \pm 0.04	0.02 \pm 0.04	931.19 \pm 420.81	0.68 \pm 0.09	0.08 \pm 0.04	596.29 \pm 25.87	0.89 \pm 0.10	0.00\pm0.00	515.48 \pm 45.53	0.70 \pm 0.18	0.00\pm0.00	616.28 \pm 59.88	0.99\pm0.01	0.00\pm0.00	396.83\pm20.39
	Ave.	0.12 \pm 0.18	0.17 \pm 0.28	886.39 \pm 288.60	0.70 \pm 0.07	0.09 \pm 0.04	606.42 \pm 26.76	0.87 \pm 0.08	0.03 \pm 0.06	505.17 \pm 54.03	0.70 \pm 0.15	0.00\pm0.01	647.39 \pm 40.74	0.98\pm0.02	0.01 \pm 0.01	408.71\pm27.47

automata, which is reflected in its performance degradation under more complex LTL specifications: the success rate η_s drops from 0.9 on φ_1 –4 to 0.82 on φ_5 –8 and 0.94 on φ_9 –12 to 0.7 on φ_{13} –16. In contrast, our method conditions only on the current subgoal, sidestepping the issue of OOD subgoal sequences. By incorporating an HJ reachability constraint, our method improves adherence to safety constraints. Additionally, since we optimize for the discounted cumulative reward obtained upon satisfying a subgoal, efficient policies that take fewer steps are encouraged. We further evaluate our method in ZoneEnv with agents featuring more complex dynamics, which induce additional hard constraints such as preventing head-down falls. Our method consistently outperforms the baselines in both success and violation rates. Details of the setup and results are provided in Appendix C.

Table 3: Evaluation results for violation rate η_v and average number of visits to accepting states μ_{acc} on infinite-horizon tasks listed in Table 1. Specifications ψ_1 – ψ_3 are evaluated in LetterWorld, and ψ_4 – ψ_6 in ZoneEnv. \uparrow : higher is better; \downarrow : lower is better. Each value is averaged over 5 seeds, with 100 trajectories per seed. **Bold** indicates the best performance for each metric.

Methods	Metrics	LetterWorld			ZoneEnv		
		ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6
GenZ-LTL(ours)	$\mu_{acc} \uparrow$	208.95\pm14.39	102.12\pm7.02	55.17\pm1.08	55.16\pm4.23	32.75\pm1.20	8135.67\pm1489.99
	$\eta_v \downarrow$	0.00\pm0.00	0.00\pm0.00	0.00\pm0.00	0.07\pm0.02	0.03\pm0.01	0.03\pm0.02
DeepLTL	$\mu_{acc} \uparrow$	142.56 \pm 22.44	48.28 \pm 12.37	19.21 \pm 4.57	30.03 \pm 13.23	15.73 \pm 4.44	7337.38 \pm 2019.56
	$\eta_v \downarrow$	0.04 \pm 0.02	0.09 \pm 0.01	0.09 \pm 0.04	0.39 \pm 0.10	0.38 \pm 0.24	0.13 \pm 0.05
GCRL-LTL	$\mu_{acc} \uparrow$	41.98 \pm 15.80	22.77 \pm 9.50	9.53 \pm 2.28	30.00 \pm 3.72	14.61 \pm 1.62	5584.34 \pm 3180.15
	$\eta_v \downarrow$	0.18 \pm 0.08	0.30 \pm 0.08	0.30 \pm 0.18	0.37 \pm 0.08	0.40 \pm 0.08	0.14 \pm 0.01

Infinite-horizon tasks. Next, we evaluate performance on infinite-horizon tasks, i.e. tasks that can only be satisfied with infinite-length trajectories. RAD-embeddings is excluded from this evaluation, as DFAs cannot represent such specifications. Since it is not possible to simulate infinite trajectories in these environments, we set the maximum episode length to be 10 times of the length used during training. The evaluated specifications are listed in Table 1, and results are reported in Table 3, including the average number of visits to accepting states for non-violating trajectories μ_{acc} and the violation rate that is calculated based on the proportion of trajectories that violate the $G \rightarrow \alpha$ constraint in the evaluated specifications. For example, in the specification $G F \alpha_1 \wedge G F \alpha_2 \wedge \dots \wedge G F \alpha_{n-1} \wedge G \neg \alpha_n$, each successful cycle of α_1 – α_{n-1} (in any order) increases μ_{acc} , while any visit to α_n counts as a violation. In the case of $F G \alpha$, μ_{acc} is computed as the number of consecutive steps, counted backward from the final timestep, in which α holds. As shown

in Table 3, our method outperforms the baselines on both metrics. By conditioning only on the current subgoal, it naturally extends to infinite-horizon tasks. The agent can follow the infinite subgoal sequence extracted from the automaton one subgoal at a time. The integration of reachability constraints further enables safe execution, whereas the baselines often struggle to satisfy safety constraints in long-horizon settings.

5.2 How does GenZ-LTL perform under increasing specification complexity and environment complexity (Q2)?

To further evaluate the zero-shot adaptation ability of our method, we design experiments that test performance under increasing complexity in specification and environment: (1) Subgoal chaining: we evaluate performance on reach-only and reach-avoid specifications with an increasing of sequence length of subgoals up to $n = 12$; (2) Environment complexity: we vary the region size and the number of regions associated with each atomic proposition; and (3) Unsatisfiable subgoals.

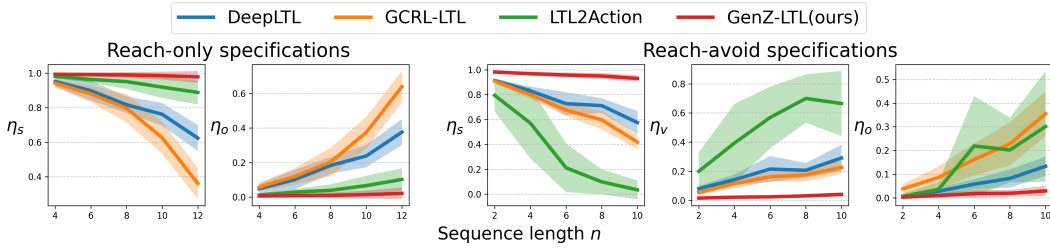


Figure 4: Performance in ZoneEnv under increasing subgoal sequence lengths, showing success rate η_s , violation rate η_v , and others rate η_o . We use reach-only specifications with sequence lengths $n \in \{4, 6, 8, 10, 12\}$ and reach-avoid specifications with $n \in \{2, 4, 6, 8, 10\}$. The specifications are shown in Table 4. Each value is averaged over 5 seeds, with 100 trajectories per seed.

Subgoal chaining. In this experiment, we increase the number of subgoals that the agent must reach sequentially. Since satisfying longer specifications naturally requires more steps, we double the maximum episode length used in Section 5.1. The results are shown in Figure 4. For the reach-only specifications, the performance of DeepLTL and GCRL-LTL degrades more rapidly as n grows. In the case of DeepLTL, it suffers from OOD specifications that involve longer subgoal sequences than those used during training. GCRL-LTL performs the worst as it struggles to complete the subgoals in an efficient manner, preventing it from reaching all the subgoals within the maximum episode length. LTL2Action performs better than DeepLTL and GCRL-LTL on reach-only specifications, as the corresponding graph structures remain relatively simple compared to the complex and nested specifications in Section 5.1. Our method maintains a high success rate and a low violation rate despite the increase in sequence length, as it solves one subgoal at a time, thus avoiding the distribution shift that affects other methods and resulting in better generalization. For reach-avoid specifications, where safety satisfaction is critical, we can observe that our method achieves near-zero constraint violations compared to the baselines, which validates our safe RL approach with reachability constraints.

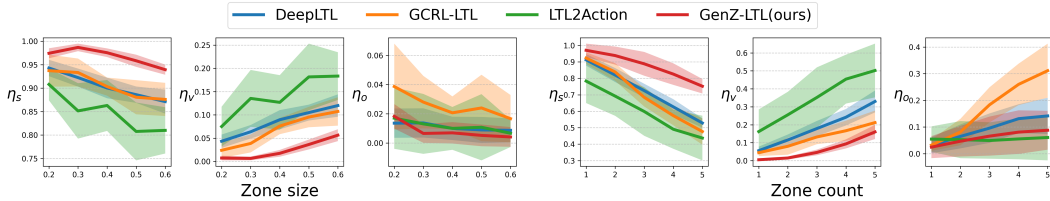


Figure 5: Performance in ZoneEnv under varying environment complexity, including different zone sizes and different numbers of zones per atomic proposition. Visualizations of the environment layout are provided in Figure 16. We report success rate η_s , violation rate η_v , and other rate η_o . The default zone size is 0.4, and the default number of zones per atomic proposition is 2. The evaluated specifications are the reach-avoid specifications with $n = 2$ shown in Table 4.

Environment complexity. We next investigate the impact of varying environment complexity performance. The results are shown in Figure 5. As the zone size increases or the number of zones associated with each atomic proposition increases, satisfying the safety constraints becomes more challenging. While all methods experience performance degradation, our method achieves the highest success rates and the lowest violation rate, which highlights its better ability to comply with safety constraints and generalize under environment variations.

Unsatisfiable subgoals. To evaluate the subgoal-switching mechanism, we use a satisfiable specification that contains an unsatisfiable subgoal: $\phi_1 : \neg \text{yellow} \cup ((\text{blue} \wedge \text{green}) \vee \text{magenta})$ in ZoneEnv. We randomly generate region layouts such that the green and blue regions (without overlaps) are positioned closer to the agent than the magenta region. In this setup, $\text{blue} \wedge \text{green}$ is usually selected as the first subgoal based on the learned value functions. As shown in Figure 6, the baselines struggle to satisfy the specification: the subgoal is unsatisfiable due to non-overlapping regions, and the agent gets stuck due to the lack of a mechanism to switch to an alternative. In contrast, our method can escape this deadlock by selecting an alternative subgoal. The trajectories in Figure 11 in Appendix C further illustrate this behavior.

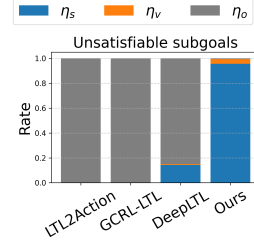


Figure 6: Evaluation results on tasks with unsatisfiable subgoals.

5.3 Ablation study: what are the individual impacts of the proposed observation reduction and safe RL approach (Q3)?

We consider a ZoneEnv environment with one region per color and reach-avoid specifications with a sequence length of $n = 2$. The environment layout is randomized in a controlled manner such that the region to be avoided is placed between the agent and the region to be reached according to an LTL specification. Under this setting, the agent must first circumvent the avoid region before reaching the goal region. For example, consider $\neg \text{magenta} \cup \text{blue}$, the magenta area is placed between the agent and the blue area. We construct a variant of our method by removing the observation reduction module and instead encoding the subgoal using the bitvector encoding discussed in Section 4.1. We also include DeepLTL as a reference, as it achieves the best performance among the baselines. The results are shown in Figure 7. We can observe that integrating the state-wise constraints leads to improved safety performance compared to the baseline method, and further incorporating the subgoal-induced observation reduction technique improves performance by significantly reducing the input dimension of the policy.

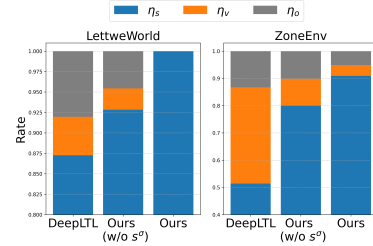


Figure 7: Average performance of success rate η_s , violation rate η_v , others rate η_o for Ours(w/o s^σ), Ours and DeepLTL for reference.

6 Conclusion

This paper presents a novel method for learning RL policies that generalize, in a zero-shot manner, to arbitrary LTL specifications by performing subgoal decomposition through the corresponding Büchi automata and completing one subgoal at a time. Empirical results demonstrate that our method, despite being myopic, achieves substantially better generalization to complex and unseen specifications compared to existing baselines. We posit that the real complexity of LTL task generalization does not lie in the structure of the LTL formula or the corresponding automata, but rather in the combinatorial nature of the atomic propositions associated with the MDP states. The observation reduction technique that we introduce is a step towards addressing this complexity. Future work will explore alternative techniques that can exploit the Boolean structure within a subgoal, such as Boolean task algebra [37], with the goal of further improving the sample efficiency and generalization ability of our method.

Acknowledgement

The authors thank the anonymous reviewers for their invaluable feedback and constructive suggestions. This work was supported in part by the U.S. National Science Foundation under grant CCF-2340776.

References

- [1] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.
- [2] C. Allen, N. Parikh, O. Gottesman, and G. Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8229–8241, 2021.
- [3] M. Bagatella, A. Krause, and G. Martius. Directed Exploration in Reinforcement Learning from Linear Temporal Logic. In *Seventeenth European Workshop on Reinforcement Learning*, Oct. 2024.
- [4] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.
- [5] J. R. Büchi. Symposium on decision problems: On a decision method in restricted second order arithmetic. In *Studies in Logic and the Foundations of Mathematics*, volume 44, pages 1–11. Elsevier, 1966.
- [6] Y. Cheng, P. Zhao, and N. Hovakimyan. Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions. In *Learning for Dynamics and Control Conference*, pages 104–115. PMLR, 2023.
- [7] M. Cho and C. Sun. Constrained meta-reinforcement learning for adaptable safety guarantee with differentiable convex programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20975–20983, 2024.
- [8] S. Cui and P. Mitra. Automated multi-task learning for joint disease prediction on electronic health records. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [9] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.
- [10] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.
- [11] J. Fu and U. Topcu. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In *Robotics: Science and Systems X*, July 2014.
- [12] M. Ganai, S. Gao, and S. Herbert. Hamilton-jacobi reachability in reinforcement learning: A survey. *IEEE Open Journal of Control Systems*, 2024.
- [13] M. Ganai, Z. Gong, C. Yu, S. Herbert, and S. Gao. Iterative reachability estimation for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36:69764–69797, 2023.
- [14] B. Gangopadhyay, P. Dasgupta, and S. Dey. Safe and stable rl (s 2 rl) driving policies using control barrier and control lyapunov functions. *IEEE Transactions on Intelligent Vehicles*, 8(2):1889–1899, 2022.
- [15] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.
- [16] Z. Guo, W. Zhou, and W. Li. Temporal logic specification-conditioned decision transformer for offline safe reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.
- [17] Z. Guo, W. Zhou, S. Wang, and W. Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

- [18] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Mungojerrie: Linear-Time Objectives in Model-Free Reinforcement Learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 527–545, 2023.
- [19] P. Hansen-Estruch, A. Zhang, A. Nair, P. Yin, and S. Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pages 8407–8426. PMLR, 2022.
- [20] M. Hasanbeig, A. Abate, and D. Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.
- [21] A. Hendawy, J. Peters, and C. D’Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024.
- [22] S. Huang, Q. Gallouédec, F. Felten, A. Raffin, R. F. J. Dossa, Y. Zhao, R. Sullivan, V. Makoviy-chuk, D. Makoviichuk, M. H. Danesh, C. Roumégous, J. Weng, C. Chen, M. M. Rahman, J. G. M. Araújo, G. Quan, D. Tan, T. Klein, R. Charakorn, M. Towers, Y. Berthelot, K. Mehta, D. Chakraborty, A. KG, V. Charraut, C. Ye, Z. Liu, L. N. Alegre, A. Nikulin, X. Hu, T. Liu, J. Choi, and B. Yi. Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning. *arXiv preprint arXiv:2402.03046*, 2024.
- [23] M. Jackermeier and A. Abate. DeepLtl: Learning to efficiently satisfy complex ltl specifications for multi-task rl. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [24] A. K. Jayant and S. Bhatnagar. Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm. *Advances in Neural Information Processing Systems*, 35:24432–24445, 2022.
- [25] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.
- [26] K. Jothimurugan, R. Alur, and O. Bastani. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 32, 2019.
- [27] D. Kim, K. Lee, and S. Oh. Trust region-based safe distributional reinforcement learning for multiple constraints. *Advances in neural information processing systems*, 36:19908–19939, 2023.
- [28] K. Kim, G. Swamy, Z. Liu, D. Zhao, S. Choudhury, and S. Z. Wu. Learning shared safety constraints from multi-task demonstrations. *Advances in Neural Information Processing Systems*, 36:5808–5826, 2023.
- [29] B. G. León, M. Shanahan, and F. Belardinelli. Systematic generalisation through task temporal logic and deep reinforcement learning. *arXiv preprint arXiv:2006.08767*, 2020.
- [30] B. G. León, M. Shanahan, and F. Belardinelli. Systematic Generalisation through Task Temporal Logic and Deep Reinforcement Learning. In *arXiv*. arXiv, 2021.
- [31] B. G. León, M. Shanahan, and F. Belardinelli. In a Nutshell, the Human Asked for This: Latent Goals for Following Temporal Specifications. In *International Conference on Learning Representations*, 2022.
- [32] X. Li, C.-I. Vasile, and C. Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.
- [33] Q. Lin, B. Tang, Z. Wu, C. Yu, S. Mao, Q. Xie, X. Wang, and D. Wang. Safe offline reinforcement learning with real-time budget constraints. In *International Conference on Machine Learning*, pages 21127–21152. PMLR, 2023.

- [34] J. X. Liu, A. Shah, E. Rosen, M. Jia, G. Konidaris, and S. Tellex. Skill Transfer for Temporal Task Specification. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2535–2541, 2024.
- [35] M. Liu, M. Zhu, and W. Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.
- [36] Z. Ma, X. Liu, and Y. Huang. Unsupervised reinforcement learning for multi-task autonomous driving: Expanding skills and cultivating curiosity. *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [37] G. Nangue Tasse, S. James, and B. Rosman. A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.
- [38] J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017.
- [39] A. Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. iee, 1977.
- [40] W. Qiu, W. Mao, and H. Zhu. Instructing goal-conditioned reinforcement learning agents with temporal logic objectives. *Advances in Neural Information Processing Systems*, 36:39147–39175, 2023.
- [41] A. Rafiei, R. Moore, S. Jahromi, F. Hajati, and R. Kamaleswaran. Meta-learning in healthcare: A survey. *SN Computer Science*, 5(6):791, 2024.
- [42] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- [43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [44] A. Shah, C. Voloshin, C. Yang, A. Verma, S. Chaudhuri, and S. A. Seshia. Deep policy optimization with temporal logic constraints. *arXiv e-prints*, pages arXiv–2404, 2024.
- [45] A. Shah, C. Voloshin, C. Yang, A. Verma, S. Chaudhuri, and S. A. Seshia. LTL-Constrained Policy Optimization with Cycle Experience Replay. In *arXiv*, 2024.
- [46] D. Shao and M. Kwiatkowska. Sample efficient model-free reinforcement learning from LTL specifications with optimality guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4180–4189, 2023.
- [47] O. So and C. Fan. Solving Stabilize-Avoid Optimal Control via Epigraph Form and Deep Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, July 2023.
- [48] A. A. Taiga, R. Agarwal, J. Farebrother, A. Courville, and M. G. Bellemare. Investigating multi-task pretraining and generalization in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.
- [49] Z. Tang, X. Chen, Y. Li, and J. Chen. Efficient and generalized end-to-end autonomous driving system with latent deep reinforcement learning and demonstrations. *arXiv preprint arXiv:2401.11792*, 2024.
- [50] P. Vaezipoor, A. C. Li, R. A. T. Icarte, and S. A. McIlraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.
- [51] C. Voloshin, A. Verma, and Y. Yue. Eventual discounting temporal logic counterfactual experience replay. In *International Conference on Machine Learning*, pages 35137–35150. PMLR, 2023.

- [52] A. Wachi, Y. Sui, Y. Yue, and M. Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [53] N. Xiong, Y. Du, and L. Huang. Provably safe reinforcement learning with step-wise violation constraints. *Advances in Neural Information Processing Systems*, 36:54341–54353, 2023.
- [54] B. Yalcinkaya, N. Lauffer, M. Vazquez-Chanlatte, and S. Seshia. Compositional automata embeddings for goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 37:72933–72963, 2024.
- [55] B. Yalcinkaya, N. Lauffer, M. Vazquez-Chanlatte, and S. A. Seshia. Provably correct automata embeddings for optimal automata-conditioned reinforcement learning. *arXiv preprint arXiv:2503.05042*, 2025.
- [56] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020.
- [57] D. Yu, H. Ma, S. Li, and J. Chen. Reachability constrained reinforcement learning. In *International conference on machine learning*, pages 25636–25655. PMLR, 2022.
- [58] H. Zang, X. Li, L. Zhang, Y. Liu, B. Sun, R. Islam, R. Tachet des Combes, and R. Laroché. Understanding and addressing the pitfalls of bisimulation-based representations in offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:28311–28340, 2023.
- [59] Y. Zhang, Q. Vuong, and K. Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.
- [60] W. Zhao, T. He, and C. Liu. Probabilistic safeguard for reinforcement learning using safety index guided gaussian process models. In *Learning for Dynamics and Control Conference*, pages 783–796. PMLR, 2023.
- [61] Y. Zheng, J. Li, D. Yu, Y. Yang, S. E. Li, X. Zhan, and J. Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024.

Broader Impacts

Our work aims to improve the generalization of reinforcement learning to satisfy arbitrary LTL specifications. This framework enables flexible and expressive task definitions, but it also places responsibility on the user to ensure that the specifications are well designed. The behavior of the agent is tightly coupled with the given specification, so poorly constructed or harmful specifications may lead to unsafe or unintended behaviors. Thus, careful design and verification of specifications are essential when applying such methods in real-world settings.

A Experimental Settings

A.1 Environments

We use the LetterWorld [50] and ZoneEnv [50, 23] for evaluation. The LetterWorld is a 7×7 grid world that contains 12 letters corresponding to atomic propositions $AP = \{a, b, \dots, l\}$. Each letter appears twice and is randomly placed on the grid. The observation consists of the entire grid in an egocentric view. The agent can move in four directions: up, down, left, and right. The grid wraps around, i.e., if the agent moves out of bounds, it reappears on the opposite side. The maximum episode length is $T = 75$. The ZoneEnv contains different colored regions corresponding to atomic propositions $AP = \{\text{blue, green, magenta, yellow}\}$ and walls acting as boundaries. Our implementation is adapted from [23]. We use the point robot that has a continuous action space for acceleration and steering. Observations consist of Lidar measurements of the colored regions and egocentric state information from other sensors. The maximum episode length is $T = 1000$. We further construct several variants of the ZoneEnv to evaluate the generalization performance of our method and baselines by modifying: (1) the region size, with $r = \{0.2, 0.3, 0.4(\text{default}), 0.5, 0.6\}$; (2) the number of regions per atomic proposition, with $\text{count} = \{1, 2(\text{default}), 3, 4, 5\}$; (3) and the number of atomic propositions, expanded to include new symbols $\{\text{red, cyan, orange, purple, teal, lime}\}$. Illustrations of the environments are provided in Figure 8 and trajectories under different LTL specifications are provided in Figure 12, 15, and 16.

We exclude the FlatWorld[23, 45] environment from our evaluation. It contains colored regions similar to ZoneEnv. However, the region layout is fixed, meaning that certain Boolean formulas over the atomic propositions cannot be evaluated since they are never true in this environment. Additionally, the observation space is limited to the agent’s (x, y) -position. Due to this lack of variability and expressiveness, we exclude FlatWorld from our evaluation.

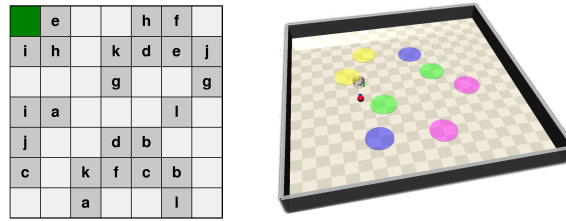


Figure 8: Visualisations of environments. Left: LetterWorld. Right: ZoneEnv

A.2 LTL Specifications

We evaluate zero-shot generalization across a range of specifications, including complex and nested finite-horizon tasks (Table 1), complex infinite-horizon tasks (Table 1), varying specification complexity with different sequence lengths of reach-only and reach-avoid specifications (Table 4), varying environment complexity in reach-avoid tasks (Table 4), and different numbers of atomic propositions (Table 5). The specifications $\varphi_1 - \varphi_4$ and $\varphi_9 - \varphi_{12}$ are from [23]. To better evaluate each method’s generalization ability and its handling of the trade-off between task progression and safety constraints, we construct more complex specifications, $\varphi_5 - \varphi_8$ and $\varphi_{13} - \varphi_{16}$, which feature longer temporal sequences and stricter safety requirements, i.e., more atomic propositions appear under negation.

Table 4: Reach-only and reach-avoid specifications. Evaluation results are shown in Figure 4.

Reach-only	n=4	$F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{magenta}))))$ $F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{yellow}))))$ $F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{green}))))$
	n=6	$F(\text{green} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{green}))))))$ $F(\text{yellow} \wedge F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{green}))))))$ $F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{magenta}))))))$
	n=8	$F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{magenta}))))))$ $F(\text{green} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{yellow}))))))$ $F(\text{magenta} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow}))))))$
	n=10	$F(\text{magenta} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow}))))))$ $F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{yellow}))))))$ $F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{magenta}))))))$
	n=12	$F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{magenta}))))))$ $F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{green} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow}))))))$ $F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{blue} \wedge F(\text{green} \wedge F(\text{yellow} \wedge F(\text{blue} \wedge F(\text{magenta} \wedge F(\text{green} \wedge F(\text{blue} \wedge F(\text{yellow} \wedge F(\text{magenta} \wedge F(\text{green}))))))$
	n=2	$(\neg \text{blue} \wedge \neg \text{green}) \text{ U } \text{yellow}$ $(\neg \text{yellow} \wedge \neg \text{magenta}) \text{ U } \text{green}$ $(\neg \text{green} \wedge \neg \text{magenta}) \text{ U } \text{blue}$
Reach-avoid	n=4	$\neg(\text{blue} \vee \text{yellow}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } \text{yellow}))$ $\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{blue} \vee \text{yellow}) \text{ U } \text{green}))$ $\neg(\text{yellow} \vee \text{magenta}) \text{ U } (\text{blue} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } \text{yellow}))$
	n=6	$\neg(\text{blue} \vee \text{yellow}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } \text{magenta}))))$ $\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{blue} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } \text{yellow}))))$ $\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } \text{green}))))$
	n=8	$\neg(\text{green} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{yellow} \vee \text{blue}) \text{ U } \text{green}))))))$ $\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } (\text{blue} \wedge (\neg(\text{magenta} \vee \text{yellow}) \text{ U } \text{green}))))))$ $\neg(\text{magenta} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } (\text{blue} \wedge (\neg(\text{magenta} \vee \text{green}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } \text{magenta}))))))$
	n=10	$\neg(\text{magenta} \vee \text{blue}) \text{ U } (\text{green} \wedge (\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{yellow} \vee \text{green}) \text{ U } (\text{blue} \wedge (\neg(\text{magenta} \vee \text{green}) \text{ U } (\text{yellow} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } \text{green}))))))$ $\neg(\text{green} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } (\text{green} \wedge (\neg(\text{magenta} \vee \text{yellow}) \text{ U } (\text{blue} \wedge (\neg(\text{magenta} \vee \text{green}) \text{ U } \text{yellow}))))))$ $\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } (\text{yellow} \wedge (\neg(\text{magenta} \vee \text{blue}) \text{ U } (\text{green} \wedge (\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{yellow}) \text{ U } \text{blue}))))))$
	n=2	$(\neg \text{blue} \wedge \neg \text{green}) \text{ U } \text{yellow}$ $(\neg \text{yellow} \wedge \neg \text{magenta}) \text{ U } \text{green}$ $(\neg \text{green} \wedge \neg \text{magenta}) \text{ U } \text{blue}$
	n=4	$\neg(\text{blue} \vee \text{yellow}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } \text{yellow}))$ $\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{blue} \vee \text{yellow}) \text{ U } \text{green}))$ $\neg(\text{yellow} \vee \text{magenta}) \text{ U } (\text{blue} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } \text{yellow}))$

Table 5: Reach-avoid specifications with more APs. Evaluation results are shown in Table 6.

ZoneEnv	IAP1 = 4	$\neg(\text{blue} \vee \text{yellow}) \text{ U } (\text{green} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } \text{magenta}))))$ $\neg(\text{yellow} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{blue} \wedge (\neg(\text{green} \vee \text{magenta}) \text{ U } \text{yellow}))))$ $\neg(\text{green} \vee \text{yellow}) \text{ U } (\text{magenta} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } \text{green}))))$
	IAP1 = 6	$\neg(\text{red} \vee \text{cyan}) \text{ U } (\text{blue} \wedge (\neg(\text{yellow} \vee \text{green}) \text{ U } (\text{magenta} \wedge (\neg(\text{cyan} \vee \text{yellow}) \text{ U } \text{green}))))$ $\neg(\text{green} \vee \text{red}) \text{ U } (\text{cyan} \wedge (\neg(\text{blue} \vee \text{magenta}) \text{ U } (\text{yellow} \wedge (\neg(\text{cyan} \vee \text{red}) \text{ U } \text{blue}))))$ $\neg(\text{yellow} \vee \text{cyan}) \text{ U } (\text{red} \wedge (\neg(\text{green} \vee \text{blue}) \text{ U } (\text{magenta} \wedge (\neg(\text{red} \vee \text{yellow}) \text{ U } \text{green}))))$
	IAP1 = 8	$\neg(\text{orange} \vee \text{red}) \text{ U } (\text{cyan} \wedge (\neg(\text{blue} \vee \text{green}) \text{ U } (\text{yellow} \wedge (\neg(\text{purple} \vee \text{blue}) \text{ U } \text{magenta}))))$ $\neg(\text{purple} \vee \text{cyan}) \text{ U } (\text{orange} \wedge (\neg(\text{red} \vee \text{yellow}) \text{ U } (\text{green} \wedge (\neg(\text{magenta} \vee \text{red}) \text{ U } \text{blue}))))$ $\neg(\text{yellow} \vee \text{purple}) \text{ U } (\text{green} \wedge (\neg(\text{cyan} \vee \text{orange}) \text{ U } (\text{blue} \wedge (\neg(\text{magenta} \vee \text{green}) \text{ U } \text{orange}))))$
	IAP1 = 10	$\neg(\text{teal} \vee \text{magenta}) \text{ U } (\text{orange} \wedge (\neg(\text{lime} \vee \text{blue}) \text{ U } (\text{yellow} \wedge (\neg(\text{green} \vee \text{purple}) \text{ U } \text{red}))))$ $\neg(\text{cyan} \vee \text{purple}) \text{ U } (\text{magenta} \wedge (\neg(\text{red} \vee \text{teal}) \text{ U } (\text{lime} \wedge (\neg(\text{orange} \vee \text{yellow}) \text{ U } \text{green}))))$ $\neg(\text{blue} \vee \text{green}) \text{ U } (\text{yellow} \wedge (\neg(\text{teal} \vee \text{lime}) \text{ U } (\text{purple} \wedge (\neg(\text{magenta} \vee \text{red}) \text{ U } \text{orange}))))$

B Implementation Details

We use the official codebases of the baselines: LTL2Action¹, GCRL-LTL², RAD-embeddings³, and DeepLTL⁴. Our implementation is adapted from DeepLTL by incorporating a safety value function and a state-dependent Lagrangian multiplier and removing DeepLTL-specific modules. For DeepLTL, LTL2Action, and our method, we use a fully connected actor network with three hidden layers of sizes [64, 64, 64]. The value function network has two hidden layers of sizes [64, 64]. For our method, both the safety value function and the Lagrangian multiplier network share the same architecture as the value function. GCRL-LTL employs a larger actor-critic architecture with [512, 1024, 256] units due to its use of a graph neural network. For fair comparison, we use the Adam optimizer with a learning rate of $3e-4$ and train all methods for 15M environment interactions. The discount factor is set to $\gamma = 0.94$ for LetterWorld and $\gamma = 0.998$ for ZoneEnv. Additional hyperparameter details are provided in the code in the supplementary materials. We train all the methods using a 16-core AMD CPU and an NVIDIA GeForce RTX 4090 GPU, and it takes roughly 3 hours to train our model.

Our method is summarized in Algorithm 1. We begin by constructing a subgoal set that includes all feasible subgoals: we first enumerate each assignment $\alpha \in 2^{AP}$ as a candidate α^+ . For each such α^+ , we filter out the remaining assignments that conflict with it. We then enumerate all possible

¹<https://github.com/LTL2Action/LTL2Action>

²<https://github.com/RU-Automated-Reasoning-Group/GCRL-LTL>

³<https://github.com/RAD-Embeddings/neurips24>

⁴<https://github.com/mathiasj33/deep-ltl>

combinations of the filtered assignments to form A^- , resulting in the full set of feasible reach-avoid subgoals $\xi = \{(\alpha^+, A^-)_i\}_{i=1}^M$, from which subgoals are sampled uniformly. The agent receives a reward $r = 1$ if the subgoal is satisfied and $r = 0$ otherwise. At each timestep, it receives a safety signal $h = -1$ if the subgoal is not violated, and $h = 1$ otherwise. The reward and safety functions are defined as $r = \mathbb{1}[L(s^\sigma) \in \alpha^+]$ and $h = 2 \cdot \mathbb{1}[L(s^\sigma) \in A^-] - 1$, respectively. If the current subgoal is satisfied, we sample a new subgoal whose avoid set does not contain the current state, i.e., $L(s_t) \notin A^-$, and whose reach set also excludes the current state, i.e., $L(s_t) \notin \alpha^+$, to prevent the new subgoal from being immediately violated or trivially satisfied. If a subgoal is violated, the episode terminates. We use Generalized Advantage Estimation (GAE) [42] to estimate value functions:

$$A_r^\pi = \sum_{l=0}^{T-t-1} (\gamma \lambda_{\text{GAE}})^l \delta_{r,t+l} \quad \delta_{r,t} = r_t + \gamma V(s_{t+1}^\sigma) - V(s_t^\sigma) \quad (4)$$

$$A_h^\pi = \sum_{l=0}^{T-t-1} (\gamma \lambda_{\text{GAE}})^l \delta_{h,t+l} \quad \delta_{h,t} = (1 - \gamma) h_t + \max(h_t, V(s_{t+1}^\sigma)) \quad (5)$$

where $\lambda_{\text{GAE}} = 0.95$ is the discount factor of GAE. At test time, given a target LTL specification, we convert it into a Büchi automaton and apply the DFS to enumerate all possible paths to accepting states. For each path, we evaluate the current subgoal using the learned value functions and select the optimal subgoal to execute. This process is applied iteratively until an accepting state is reached.

Algorithm 1 GenZ-LTL

Require: Initial policy π , value and safety value function V_r, V_h , state-dependent Lagrangian multiplier λ , subgoal set ξ , maximum training iterations K , interactions per iteration N

- 1: **for** $k = 0, 1, 2, \dots, K$ **do**
- 2: **for** $n = 0, 1, 2, \dots, N$ **do**
- 3: **if** the current subgoal is not specified **or** the current subgoal is satisfied/violated **then**
- 4: Sample a subgoal $\sigma \sim \text{Unif}(\xi)$
- 5: **end if**
- 6: Collect trajectory $\tau = \{s_t^\sigma, a_t, r_t, h_t\}$
- 7: **end for**
- 8: Compute reward-to-go $\hat{R}_t \doteq \sum_{i=t}^T \gamma^i r_i$ and cost-to-go $\hat{H}_t \doteq \max_t h_t$ for each τ
- 9: Compute advantage functions A_r^π, A_h^π , based on V_r and V_h using (4) and (5).
- 10: Compute the Lagrangian multiplier λ .
- 11: Fit value function, safety value function by regression on mean-square error.
- 12: Update the policy parameters by maximizing (3).
- 13: Update the multiplier parameters by minimizing (3).
- 14: **end for**

B.1 Subgoal-induced observation reduction.

For Lidar measurements in the ZoneEnv, $f(\cdot)$ is implemented as the element-wise min to obtain the closest distance along each Lidar beam, such that the resulting s^σ . For grid maps in LetterWorld, we assign distinct values to each cell associated with σ at location (i, j) : $f(s_{i,j}) = v_{\text{reach}}$ if $L(s_{i,j}) = \alpha^+$; v_{avoid} if $L(s_{i,j}) \in A^-$; v_{neutral} otherwise. This reduction allows the agent to focus on "reach" and "avoid" rather than specific set of atomic propositions and eliminates the need of encoding subgoals, enabling efficient policy learning and generalization to new atomic propositions that satisfies the conditioned discussed in Section 4.2 as shown later in Appendix C. If multiple observation functions are available, each capable of observing an assignment $\alpha \in 2^{AP}$, we can first fuse the observations within each observation type based on the current subgoal σ and then concatenate the results to form the final state s^σ . In cases where each atomic proposition is tied to a distinct observation type, our method can be applied; however, learning policies in such settings is inherently intractable due to the exponential number of subgoal-state combinations.

The actual implementation of the observation reduction may appear straightforward, as the fusion operator is environment/sensor-specific. However, the idea behind this technique is broadly applicable. When solving the reach-avoid problem, what truly matters is whether a state satisfies the reach or avoid condition under the current subgoal, rather than its specific label $L(s)$ as we mentioned in Section 4.2. Moreover, our method can be extended to more realistic settings with some additional effort, but without requiring fundamental changes to the core approach. For example, in autonomous driving scenarios with multiple sensors such as cameras and LiDAR, observation reduction can be enabled through perception modules. Semantic segmentation can be applied to both image and point

cloud data to extract relevant features based on the current reach-avoid subgoals. For instance, if the avoid subgoal involves avoiding other cars and pedestrians, and the reach subgoal involves staying within the current lane, then perception can be used to segment the scene accordingly. This allows the agent to focus on task-relevant information and use processed observations, rather than raw sensory data, to learn value functions and policies effectively.

Algorithm 2 Timeout-based subgoal-switching mechanism

Require: Trained agent, Büchi automaton $\mathcal{B}_\varphi := (\mathcal{Q}, \Sigma, \delta, \mathcal{F}, q_0)$, timeout value t_{\max}

```

1:  $\mathcal{Q}_c \leftarrow \{q_0\}$  ▷ Current set of states.
2:  $\mathcal{U} \leftarrow \emptyset$  ▷ Set of unsatisfiable transitions.
3: while FindSubgoals( $\mathcal{B}_\varphi, \mathcal{Q}_c, \mathcal{U}$ )  $\neq \emptyset$  do
4:   Select subgoal  $\sigma = (\alpha^+, A^-)$  from FindSubgoals( $\mathcal{B}_\varphi, \mathcal{Q}_c, \mathcal{U}$ ) with maximum value.
5:   for  $t = 1, 2, \dots, t_{\max}$  do
6:     Get an action from the agent, apply it, and observe  $\alpha \in \Sigma$ .
7:      $\mathcal{Q}_o \leftarrow \mathcal{Q}_c$  ▷ Remember the old states.
8:      $\mathcal{Q}_c \leftarrow \{q' \mid \exists q \in \mathcal{Q}_o, (q, \alpha, q') \in \delta\}$  ▷ Update the current set of states.
9:     if  $\mathcal{Q}_c \neq \mathcal{Q}_o$  then
10:      break ▷ A new state set is reached; the subgoal should be updated.
11:     else if  $t = t_{\max}$  then
12:       $\mathcal{U} \leftarrow \mathcal{U} \cup \{(q, \alpha^+) \mid q \in (\mathcal{Q}_c - \mathcal{F})\}$  ▷ If  $q$  is not accepting, mark this subgoal as unsatisfiable.
13:     end if
14:   end for
15: end while
16: All subgoals are marked as unsatisfiable. Terminate.
```

B.2 Timeout & Subgoal-switching Mechanism

At the test time, given a target LTL specification, we convert it into a Büchi automaton and apply a depth-first search (DFS) to enumerate all possible paths to accepting states. However, not all of these paths, or in some cases, any, are feasible in the physical environment, as some may include subgoals that are unsatisfiable. Although the labeling function is known, meaning we can determine which propositions are true for any given state, it does not necessarily imply that we can exhaustively iterate over the entire state space to identify unsatisfiable assignments.

For example, consider the specification $\neg \text{yellow} \cup ((\text{blue} \wedge \text{green}) \vee \text{magenta})$. To satisfy this formula, the agent must eventually reach either a magenta region or a region where blue and green overlap. However, the latter becomes unsatisfiable if the blue and green regions are disjoint. In such cases, the agent must be able to adapt by pursuing an alternative subgoal, a capability that is essential for goal-conditioned policies, including our approach as well as existing baselines such as DeepLTL and GCRL-LTL. However, these baselines do not address this issue.

To address this, we implement a timeout mechanism that allows the agent to revisit the automaton and search for alternative subgoals when the current one cannot be satisfied. We determine subgoal satisfiability by enforcing a timeout threshold $(1 + \epsilon_{\text{scale}}) \cdot \mu_{\text{subgoal}}$ as discussed in Section 4.1: if the agent fails to achieve this subgoal within that timestep, it is considered unsatisfiable, and the agent switches its subgoal. If no feasible subgoals remain, the episode terminates.

This timeout & subgoal-switching mechanism is explained in Algorithm 2, which outlines the general mode of operation during the test time. We use a set $\mathcal{U} : \mathcal{Q} \times \Sigma$ to store the transitions that are marked as unsatisfiable after a timeout. These unsatisfiable transitions are ignored by the FindSubgoals function. Apart from this, the only difference between FindSubgoals and the DFS algorithm used in DeepLTL is that we separate the subgoals after identifying the paths with accepting cycles.

C Additional Experiments

Training curves. We first evaluate simple reach-only and reach-avoid specifications with sequence length of $n = 3$ every specific training interval, and the success and violation rates are shown in Figure 9. We can observe that our method outperforms the baselines in terms of achieving a lower

violation rate (close to zero constraint violation) while maintaining a high success rate. The violation rates of the baselines also decrease over training because task progression and safety constraints are jointly encoded in the LTL specification, so optimizing for satisfaction of the specification implicitly improves safety performance. However, due to the inherent trade-off between these requirements, the baselines struggle to further reduce violation rates. In contrast, by formulating the problem using safe RL framework with reachability constraints, our method achieves stricter constraint satisfaction, thereby improving the overall probability of satisfying the full LTL specifications.

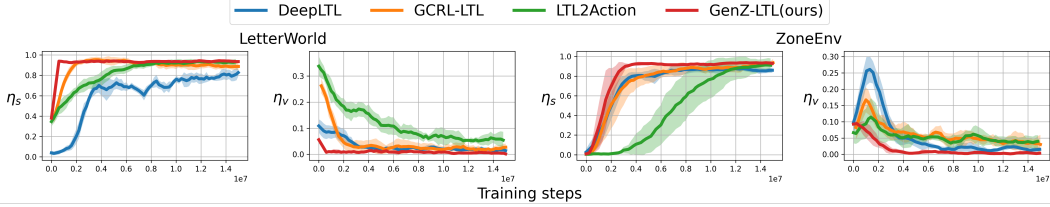


Figure 9: Evaluation curves on simple reach-only and reach-avoid specifications with $n = 3$. Each value is averaged over 5 seeds with 50 trajectories with randomly sampled tasks.

Partial observability. To satisfy arbitrary LTL specifications, we leverage the structure of the Büchi automaton to decompose each formula into a sequence of subgoals and handle one subgoal at a time, progressing through the sequence incrementally. While our method could be extended to condition on the full subgoal sequence, we argue that this introduces significant drawbacks that outweigh the potential benefits. Among the baselines, DeepLTL conditions its policy on the full subgoal sequence to achieve non-myopic behavior. Intuitively, this "look-ahead" capability may help the agent satisfy the specification in fewer steps. However, subgoal sequences come with several disadvantages. First, policies trained on full sequences are more likely to encounter OOD issues at test time, as the sequence length can vary across specifications. Although one might limit the policy to observe at most k subgoals, where k is the maximum length seen during training, choosing k is nontrivial. A large k can lead to exponential sample complexity due to the combinatorial explosion of subgoal sequences, e.g., each subgoal is drawn from 2^{AP} , while a small k limits the look-ahead benefit. Moreover, in practice, the agent may not perceive all relevant subgoals at once, rendering the subgoal sequence ineffective. To illustrate this, we conduct experiments where the agent’s observability is restricted to half of the environment. We train both DeepLTL and our method in LetterWorld and ZoneEnv, and then evaluate them using specifications φ_1 – φ_3 and reach-avoid tasks with $n = 4$. We report the average success rate η_s and violation rate η_v . As shown in the left and middle plots of Figure 10, our single-subgoal-at-a-time method outperforms DeepLTL in terms of both the success and violation rates, despite the latter using full subgoal sequences.

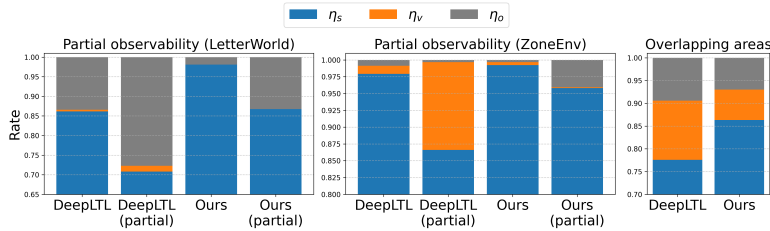


Figure 10: Evaluation results of DeepLTL and our method in two settings: (1) partial observability (left two plots) and (2) overlapping areas (the rightmost plot).

Overlapping areas. We construct a variant of ZoneEnv to introduce a more challenging setting where multiple atomic propositions can be true simultaneously, i.e., different colored regions may overlap in arbitrary configurations. Note that FlatWorld is an overly simplified case where the layout is fixed. In ZoneEnv, overlapping areas can be handled in two ways: (1) by adding separate observations for each assignment involving multiple true propositions, or (2) by using observations only for individual atomic propositions (default setting for ZoneEnv) and learning policies through interaction with the environment, during which the agent implicitly discovers the relationships among propositions and observations. The first approach does not offer meaningful advantages over the second, as our observation reduction technique can extract the relevant parts of the observation

for any target assignment and fuse them as needed. Thus, we adopt the second approach, which presents a more challenging generalization problem. To accommodate this scenario, we modify DeepLTL’s curriculum training to allow multiple atomic propositions to be true at the same time. The positions of both the agent and the regions are randomly sampled. We evaluate both methods using reach-avoid specifications with $n = 2$: $\phi_2 : \neg(\text{magenta} \vee \text{yellow}) \cup (\text{blue} \wedge \text{green})$, $\phi_3 : \neg(\text{blue} \vee \text{magenta}) \cup (\text{yellow} \wedge \text{green})$, $\phi_4 : \neg(\text{yellow} \vee \text{blue}) \cup (\text{green} \wedge \text{magenta})$, and $\phi_5 : \neg(\text{green} \vee \text{magenta}) \cup (\text{yellow} \wedge \text{blue})$. Trajectory illustrations are shown in Figure 11. We report the average rates and the results in the rightmost plot in Figure 10 show that our method achieves a higher success rate and a lower violation rate. Our method can better satisfy safety constraints due to the integration of reachability constraints.

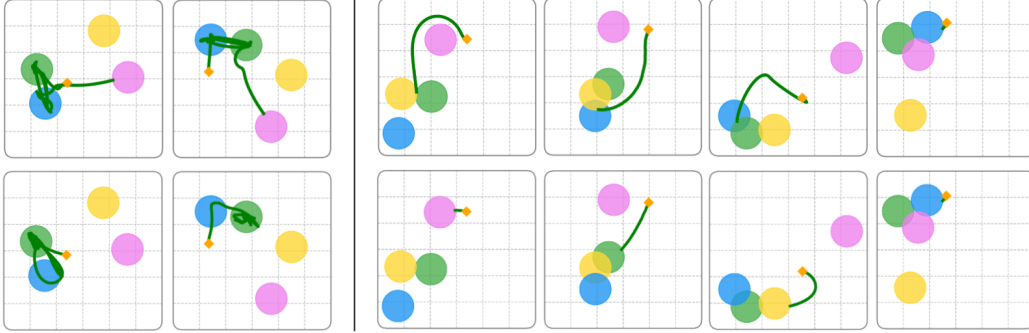


Figure 11: Illustration of trajectories in ZoneEnv: (1) unsatisfiable subgoals (left two columns, ϕ_1) and (2) overlapping areas (right four columns, ϕ_2 – ϕ_5). The top row shows trajectories generated by our method, while the bottom row shows those from DeepLTL.

Number of atomic propositions. We also evaluate performance as the number of atomic propositions increases. In this experiment, we emphasize that the observations corresponding to the additional atomic propositions satisfy the conditions outlined in Section 4.2. It is important to note that when new atomic propositions are observed through modalities not encountered during training — such as those representing the agent’s speed or heading — zero-shot adaptation remains an open challenge. The results are presented in Table 6, and example trajectories are shown in Figure 12. Among the evaluated methods, only our method can generalize in a zero-shot manner to new atomic propositions whose observations are produced by identical observation functions under output transformation, as the subgoal-induced observation reduction technique fuses these propositions into "reach" and "avoid", allowing the agent to make decisions without relying on their specific labels $L(s)$.

Table 6: Performance in ZoneEnv with increasing number of atomic propositions. We evaluate the reach-avoid specifications with sequence length of $n = 6$, using an increasing number of atomic propositions as listed in Table 5. We report success rate η_s , violation rate η_v , other rate η_o , and average steps to satisfy the specifications μ .

Metrics	$ AP = 4$	$ AP = 6$	$ AP = 8$	$ AP = 10$
$\eta_s \uparrow$	0.96 ± 0.02	0.94 ± 0.02	0.91 ± 0.03	0.93 ± 0.03
$\eta_v \downarrow$	0.02 ± 0.01	0.02 ± 0.01	0.03 ± 0.01	0.02 ± 0.01
$\eta_o \downarrow$	0.02 ± 0.02	0.04 ± 0.02	0.06 ± 0.03	0.05 ± 0.02
$\mu \downarrow$	306.58 ± 16.88	314.88 ± 17.02	313.10 ± 10.61	323.30 ± 12.57

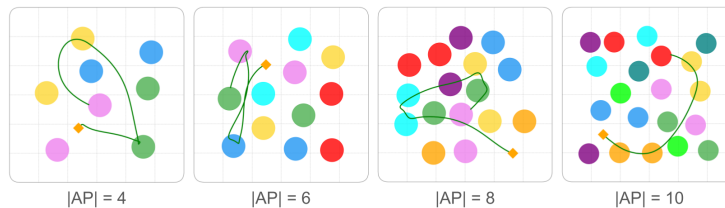


Figure 12: Illustration of ZoneEnv with an increasing number of atomic propositions. The trajectories shown correspond to the first specification for $|AP| = 4, 6, 8, 10$ in Table 5.

Deterministic v.s. stochastic policies. As discussed in Section 5, we follow the standard practice of using a deterministic policy during evaluation. We also note that DeepLTL is evaluated using a stochastic policy in LetterWorld in its original paper. To provide a comprehensive comparison, we report results for both stochastic and deterministic policies in LetterWorld, as shown in Figure 13. We observe that DeepLTL exhibits inconsistent behavior: while the success rate improves under a stochastic policy, the violation rate also increases. This trade-off is unacceptable in safety-critical applications and indicates that DeepLTL struggles to maintain safety. In contrast, our method delivers consistent performance without compromising safety for task progression, since we explicitly model the violation of subgoals as a reachability constant in a safe RL formulation. In our view, stochastic policies are not suitable for evaluation. For reach-only specifications, they contain no safety constraints; a stochastic policy may satisfy them simply by randomly exploring the environment, thereby overstating performance. For reach-avoid specifications, stochasticity further introduces inconsistency: the mean action may be safe, while a sampled action may lead to a violation, or vice versa. This randomness complicates evaluation and prevents a fair comparison between methods.

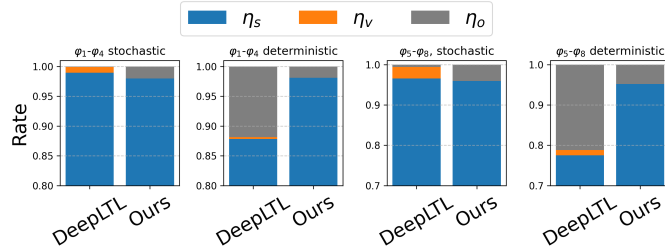


Figure 13: Evaluation results of DeepLTL and our method in LetterWorld using stochastic and deterministic policies. Each value is averaged over 5 seeds, with 100 trajectories per seed.

Table 7: Performance in ZoneEnv with Ant agent. We report the success rate η_s , violation rate η_v , and the average steps to satisfy the specifications μ . Each value is averaged over 5 seeds, with 100 trajectories per seed.

ZoneEnv	GenZ-LTL(ours)			DeepLTL			
	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	$\eta_s \uparrow$	$\eta_v \downarrow$	$\mu \downarrow$	
	φ_9	0.97 ± 0.02	0.02 ± 0.01	305.32 ± 46.50	0.00 ± 0.01	0.23 ± 0.08	865.50 ± 72.83
	φ_{10}	0.91 ± 0.03	0.09 ± 0.03	202.00 ± 53.33	0.03 ± 0.02	0.39 ± 0.11	588.58 ± 166.33
	φ_{11}	0.95 ± 0.02	0.04 ± 0.02	182.88 ± 38.66	0.07 ± 0.02	0.27 ± 0.05	604.87 ± 28.22
	φ_{12}	0.98 ± 0.01	0.00 ± 0.00	125.97 ± 35.66	0.24 ± 0.05	0.05 ± 0.04	392.04 ± 64.39
	φ_{9-12}	0.95 ± 0.04	0.04 ± 0.04	204.04 ± 77.69	0.09 ± 0.10	0.23 ± 0.14	568.15 ± 172.57
	φ_{13}	0.93 ± 0.04	0.00 ± 0.00	459.99 ± 86.69	0.00 ± 0.00	0.00 ± 0.00	-
	φ_{14}	0.95 ± 0.02	0.00 ± 0.00	434.41 ± 76.87	0.00 ± 0.00	0.00 ± 0.00	-
	φ_{15}	0.89 ± 0.02	0.08 ± 0.02	429.76 ± 75.75	0.00 ± 0.00	0.39 ± 0.13	-
φ_{16}	0.95 ± 0.03	0.00 ± 0.00	449.89 ± 83.26	0.00 ± 0.00	0.00 ± 0.00	-	
φ_{13-16}	0.93 ± 0.04	0.02 ± 0.04	443.51 ± 73.31	0.00 ± 0.00	0.10 ± 0.18	-	

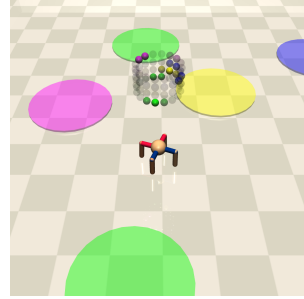


Figure 14: Visualization of the Ant agent in ZoneEnv.

Complex Dynamics. To evaluate the performance of our method in environments with more complex dynamics, we conduct additional experiments using the Ant agent [25], consisting of a torso and four legs connected by hinge joints, in the Zone environment. The state space includes the agent’s ego-state (40 dimensions, compared to 12 for the Point agent used in our main experiments) and LiDAR observations of the regions. A visualization of the Ant agent is shown in Figure 14. The action space has 8 dimensions, corresponding to the torques applied to the Ant’s joints to coordinate leg movements (compared to 2 dimensions for the Point agent). We train and evaluate our method against DeepLTL, the strongest baseline in our main results. For a fair comparison, all training procedures, training parameters, and neural network sizes were kept the same (note that the last one in particular could result in some performance drop as the state dimension and action dimension are bigger for the ant agent). Both methods are trained until convergence. The evaluation results are shown in Table 7. We can observe that our method consistently outperforms the baseline across all metrics, demonstrating an even larger performance gain in environments with more complex agent dynamics. Note that for the Ant agent, in addition to the LTL specifications, there is a built-in safety constraint that the agent should not fall headfirst, enforced as a hard constraint with immediate episode termination. Although a negative reward is assigned in such cases, as is done when a specification is

violated for DeepLTL, it does not explicitly model safety constraints as our method does through HJ reachability, making it less effective in handling such cases. This impedes the learning of coordinated locomotion and then the satisfaction of LTL specifications, leading to degraded performance.

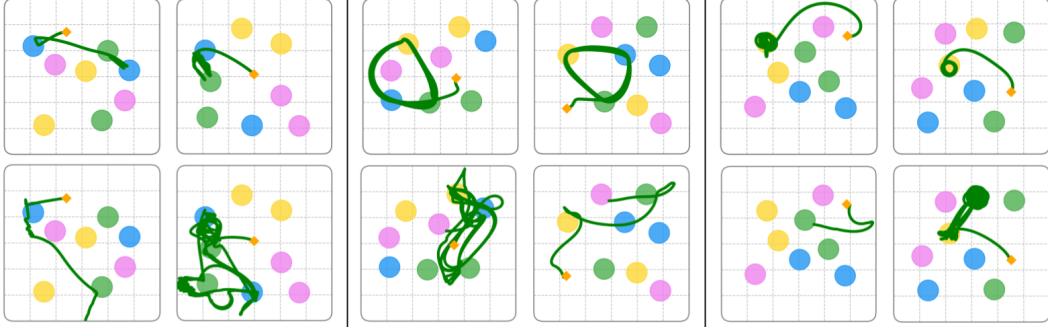


Figure 15: Illustration of infinite-horizon tasks in ZoneEnv. The trajectories correspond to specifications ψ_1 – ψ_3 , shown from left to right (two columns per specification). The top row shows trajectories generated by our method, while the bottom row shows those from DeepLTL.

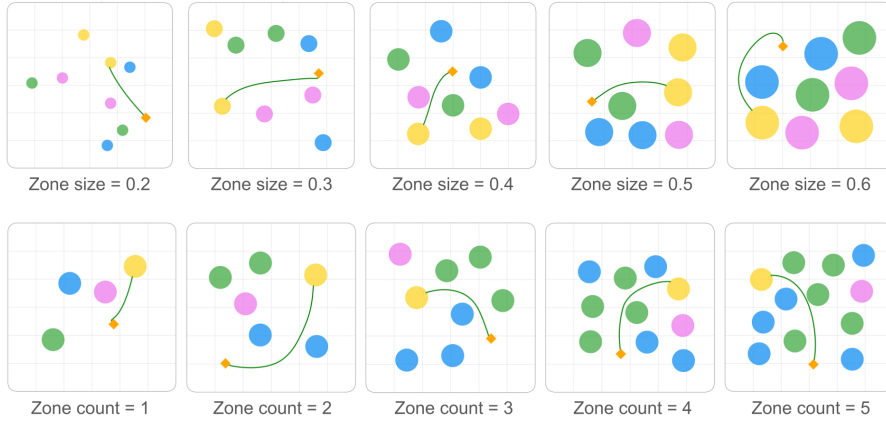


Figure 16: Illustration of ZoneEnv under increasing environment complexity, including different zone sizes and different numbers of zones per atomic proposition. The default zone size is 0.4, and the default number of zones per atomic proposition is 2. The trajectories are generated by our method and correspond to the evaluated specification $(\neg \text{blue} \wedge \neg \text{green}) \cup \text{yellow}$.

D Visualizations

We provide visualizations for the experiments in Section 5.1 and Section 5.2, including infinite-horizon tasks (Figure 15), increasing environment complexity (Figure 16) and increasing numbers of atomic propositions (Figure 12). For infinite-horizon tasks, our method exhibits a clear recurring pattern that visits the specified regions in the correct order, while the baseline struggles to do so and suffers from safety constraint violations. For different environment complexities, we vary the zone size and zone count per atomic proposition. Importantly, we increase the zone count only for propositions the agent must avoid, since doing so for propositions the agent need to satisfy would simplify the task. Larger zones or more zones per proposition force the agent to navigate around constrained regions to reach the target, and the resulting trajectories demonstrate that our method satisfies the safety constraints. For experiments with an increasing number of atomic propositions, we incrementally introduce new propositions, up to 10 in total, each corresponding to a new colored region with associated lidar observations. This makes zero-shot generalization challenging for baseline methods, as they explicitly encode atomic propositions and cannot generalize to unseen ones without retraining. Even substituting one proposition for another, for example, replacing blue with red, causes these baselines to fail, due to their dependence on fixed encodings. In contrast, our method uses subgoal-induced observation reduction, which removes this dependency when atomic

propositions share the same property. This allows our method to focus on the semantics of the propositions, i.e., "reach" or "avoid", rather than the specific proposition symbols, enabling zero-shot generalization to unseen atomic propositions.

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: The abstract and introduction clearly state the paper's goals, methods, and contributions, which are consistently supported and developed throughout the paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

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

Answer: [\[Yes\]](#)

Justification: We include the limitations in the conclusion section and the Appendix.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: [\[NA\]](#)

Justification: While the paper does not include theoretical results, it clearly states the necessary assumptions underlying the proposed method.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: The paper provides sufficient details about the experimental setup, including environment configurations, evaluation metrics, and baselines, to support reproducibility of the main results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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 provided in the supplementary materials, along with instructions sufficient to reproduce the main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: The paper specifies key details of the experimental setup, including environment settings, training and evaluation protocols, hyperparameters, and optimizer configurations in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as 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: The paper reports the mean and variance of evaluation metrics across multiple runs to support the statistical reliability of the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: The paper provides details on compute resources, including hardware specifications and runtime, in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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 have reviewed the NeurIPS Code of Ethics and confirm that the research complies with all its guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

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

Answer: [Yes]

Justification: The paper discusses potential positive and negative societal impacts of the proposed work in a section provided before the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: [\[Yes\]](#)

Justification: Our work does not involve the release of high-risk data or models. However, we note that the potential risks stem from how users specify LTL tasks, and we emphasize in the broader impacts that careful specification design is essential to avoid unintended or unsafe behaviors.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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 external assets used in this work, including code and environments, are properly cited in the paper. We respect the terms of use and licenses associated with these assets, and license information is provided where applicable.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

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

Answer: [\[Yes\]](#)

Justification: We release new code assets as part of this work. The code is accompanied by documentation that describes usage, dependencies, training details, and environment integration.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: This work does not involve research with human subjects and therefore does not require IRB approval or equivalent review.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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: LLMs were only used for writing and editing purposes and did not contribute to the core methodology or scientific content of the research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.