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` [49] and `ZoneEnv` [49, 23] for evaluation. The `LetterWorld` is a 7×7 grid world that contains 12 letters corresponding to atomic propositions $AP = \{\mathsf{a}, \mathsf{b}, \ldots, \mathsf{l}\}$. 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 $AP = \{\mathsf{blue}, \mathsf{green}, \mathsf{magenta}, \mathsf{yellow}\}$ 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 count $= \{1, 2(\text{default}), 3, 4, 5\}$; (3) and the number of atomic propositions, expanded to include new symbols $\{\mathsf{red}, \mathsf{cyan}, \mathsf{orange}, \mathsf{purple}, \mathsf{teal}, \mathsf{lime}\}$. Illustrations of the environments are provided in Figure 7 and trajectories under different LTL specifications are provided in Figure 11, 13, and 14.

We exclude the `FlatWorld`[23, 44] 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.

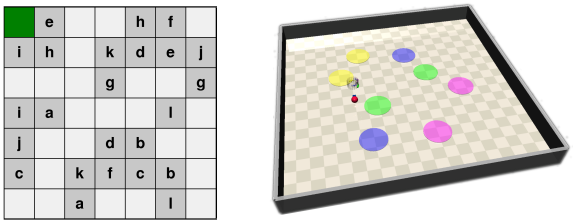

Figure 7: Visualisations of environments. Left: `LetterWorld`. Right: `ZoneEnv`

A.2 LTL Specifications

We evaluate zero-shot generalization across a range of specifications, including complex and nested finite-horizon tasks (Table 3), complex infinite-horizon tasks (Table 4), varying specification complexity with different sequence lengths of reach-only and reach-avoid specifications (Table 5), varying environment complexity in reach-avoid tasks (Table 5), and different numbers of atopic propositions (Table 6). 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, φ_5–φ_8 and φ_{13}–φ_{16}, which feature longer temporal sequences and stricter safety requirements, i.e., more atomic propositions appear under negation.

Table 3: Complex finite-horizon specifications used in our evaluation. Results are shown in Table 1.

LetterWorld	φ_1	$\mathsf{F}\,(a \wedge (\neg b\ \mathsf{U}\ c)) \wedge \mathsf{F}\,d$
	φ_2	$(\mathsf{F}\,d) \wedge (\neg f\ \mathsf{U}\ (d \wedge \mathsf{F}\,b))$
	φ_3	$\neg a\ \mathsf{U}\ (b \wedge (\neg c\ \mathsf{U}\ (d \wedge (\neg e\ \mathsf{U}\ f))))$
	φ_4	$(a \vee b \vee c \vee d \Rightarrow \mathsf{F}\,(e \wedge \mathsf{F}\,(f \wedge \mathsf{F}\,g)))\ \mathsf{U}\ (h \wedge \mathsf{F}\,i)$
	φ_5	$\mathsf{F}\,(d \wedge (\neg(a \vee b)\ \mathsf{U}\ (b \wedge (\neg e\ \mathsf{U}\ c)))) \wedge \mathsf{F}\,(\neg(f \vee g \vee h)\ \mathsf{U}\ a)$
	φ_6	$\mathsf{F}\,((k \wedge ((\neg b \vee c)\ \mathsf{U}\ f)) \wedge (\neg(a \vee e \vee h)\ \mathsf{U}\ g)) \wedge \mathsf{F}\,d$
	φ_7	$\neg(j \vee b \vee d)\ \mathsf{U}\ (a \wedge (\neg c\ \mathsf{U}\ (f \wedge \mathsf{F}\,(g \wedge (\neg d\ \mathsf{U}\ e)))))$
	φ_8	$\neg(f \vee g)\ \mathsf{U}\ (a \wedge (\neg b\ \mathsf{U}\ c) \wedge \mathsf{F}\,(d \wedge (\neg e\ \mathsf{U}\ f)))$
ZoneEnv	φ_9	$(\mathsf{F}\,\text{blue}) \wedge (\neg\text{blue}\ \mathsf{U}\ (\text{green} \wedge \mathsf{F}\,\text{yellow}))$
	φ_{10}	$\neg(\text{magenta} \vee \text{yellow})\ \mathsf{U}\ (\text{blue} \wedge \mathsf{F}\,\text{green})$
	φ_{11}	$\neg\text{green}\ \mathsf{U}\ ((\text{blue} \vee \text{magenta}) \wedge (\neg\text{green}\ \mathsf{U}\ \text{yellow}))$
	φ_{12}	$(\text{green} \vee \text{blue} \Rightarrow (\neg\text{yellow}\ \mathsf{U}\ \text{magenta}))\ \mathsf{U}\ \text{yellow}$
	φ_{13}	$\mathsf{F}\,(\text{green} \wedge (\neg(\text{blue} \vee \text{yellow})\ \mathsf{U}\ (\text{yellow} \wedge (\neg\text{magenta}\ \mathsf{U}\ \text{blue})))) \wedge \mathsf{F}\,(\neg\text{green}\ \mathsf{U}\ \text{yellow})$
	φ_{14}	$\mathsf{F}\,((\text{blue} \vee \text{green}) \wedge (\neg\text{yellow}\ \mathsf{U}\ (\text{blue} \wedge (\neg(\text{green} \vee \text{magenta})\ \mathsf{U}\ \text{magenta})))) \wedge \mathsf{F}\,(\text{yellow} \wedge (\neg\text{blue}\ \mathsf{U}\ \text{green}))$
	φ_{15}	$\neg(\text{magenta} \vee \text{yellow})\ \mathsf{U}\ (\text{blue} \wedge (\neg\text{green}\ \mathsf{U}\ (\text{yellow} \wedge \mathsf{F}\,(\text{green} \wedge (\neg\text{blue}\ \mathsf{U}\ \text{magenta})))))$
	φ_{16}	$\mathsf{F}\,(\text{blue} \wedge (\neg\text{yellow}\ \mathsf{U}\ (\text{green} \wedge \mathsf{F}\,(\text{yellow} \wedge (\neg(\text{magenta} \vee \text{green})\ \mathsf{U}\ \text{blue})))))$

Table 4: Infinite-horizon specifications used in our evaluation. Results are shown in Table 2.

LetterWorld	ψ_1	$\mathsf{G}\,\mathsf{F}\,\text{blue} \wedge \mathsf{G}\,\mathsf{F}\,\text{green} \wedge \mathsf{G}\,\neg(\text{yellow} \vee \text{magenta})$
	ψ_2	$\mathsf{G}\,\mathsf{F}\,\text{blue} \wedge \mathsf{G}\,\mathsf{F}\,\text{yellow} \wedge \mathsf{G}\,\mathsf{F}\,\text{green} \wedge \mathsf{G}\,\neg\text{magenta}$
	ψ_3	$\mathsf{F}\,\mathsf{G}\,\text{yellow} \wedge \mathsf{G}\,\neg(\text{green} \vee \text{blue} \vee \text{magenta})$
ZoneEnv	ψ_4	$\mathsf{G}\,\mathsf{F}\,(e \wedge (\neg a\ \mathsf{U}\ f)) \wedge \mathsf{G}\,\neg(c \vee d)$
	ψ_5	$\mathsf{G}\,\mathsf{F}\,a \wedge \mathsf{G}\,\mathsf{F}\,b \wedge \mathsf{G}\,\mathsf{F}\,c \wedge \mathsf{G}\,\neg(e \vee f \vee i)$
	ψ_6	$\mathsf{G}\,\mathsf{F}\,c \wedge \mathsf{G}\,\mathsf{F}\,a \wedge \mathsf{G}\,\mathsf{F}\,(e \wedge (\neg f\ \mathsf{U}\ g)) \wedge \mathsf{G}\,\mathsf{F}\,k \wedge \mathsf{G}\,\neg(i \vee j)$

Table 5: Reach-only and reach-avoid specifications. Evaluation results are shown in Figure 3.

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

Table 6: Reach-avoid specifications with more APs. Evaluation results are shown in Table 7.

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

B Implementation Details

We use the official codebases of the baselines: LTL2Action[1], GCRL-LTL[2], and DeepLTL[3]. 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 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) [41] to estimate value functions:

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

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

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

for $k = 0, 1, 2, \ldots, K$ **do**
  for $n = 0, 1, 2, \ldots, N$ **do**
    if the current subgoal is not specified **or** the current subgoal is satisfied/violated **then**
      Sample a subgoal $\sigma \sim \text{Unif}(\xi)$
    end if
    Collect trajectory $\tau = \{s_t^\sigma, a_t, r_t, h_t\}$
  end for
  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
  Compute advantage functions A_r^π, A_h^π, based on V_r and V_h using (4) and (5).
  Compute the Lagrangian multiplier λ.
  Fit value function, safety value function by regression on mean-square error.
  Update the policy parameters by maximizing (3).
  Update the multiplier parameters by minimizing (3).
end for

[1] `https://github.com/LTL2Action/LTL2Action`

[2] `https://github.com/RU-Automated-Reasoning-Group/GCRL-LTL`

[3] `https://github.com/mathiasj33/deep-ltl`

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 is not applicable. However, learning policies in such settings is inherently intractable due to the exponential number of subgoal-state combinations.

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 $\neg\mathsf{yellow}\ \mathsf{U}\ ((\mathsf{blue} \wedge \mathsf{green}) \vee \mathsf{magenta})$. 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.

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}

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

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 8. 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.

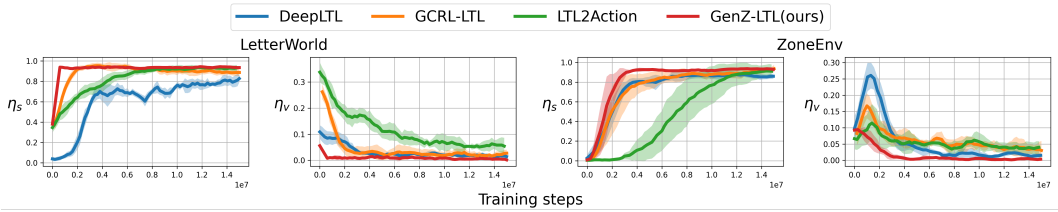

Figure 8: Evaluation curves on simple reach-only and reach-avoid specifications with $n = 3$. Each value is averaged over 5 seeds with 50 trajectories with randomly sampled tasks.

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 9, 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.

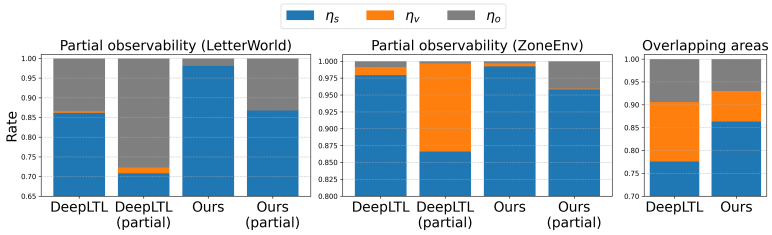

Figure 9: Evaluation results of DeepLTL and our method in two settings: (1) partial observability (left two plots) and (2) overlapping areas (the rightmost plot).

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 $n = 2$: ϕ_2 : $\neg(\mathsf{magenta} \vee \mathsf{yellow})\ \mathsf{U}\ (\mathsf{blue} \wedge \mathsf{green})$, ϕ_3 : $\neg(\mathsf{blue} \vee \mathsf{magenta})\ \mathsf{U}\ (\mathsf{yellow} \wedge \mathsf{green})$, ϕ_4 : $\neg(\mathsf{yellow} \vee \mathsf{blue})\ \mathsf{U}\ (\mathsf{green} \wedge \mathsf{magenta})$, and ϕ_5 : $\neg(\mathsf{green} \vee \mathsf{magenta})\ \mathsf{U}\ (\mathsf{yellow} \wedge \mathsf{blue})$. Trajectory illustrations are shown in Figure 10. We report the average rates and the results in the rightmost plot in Figure 9 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.

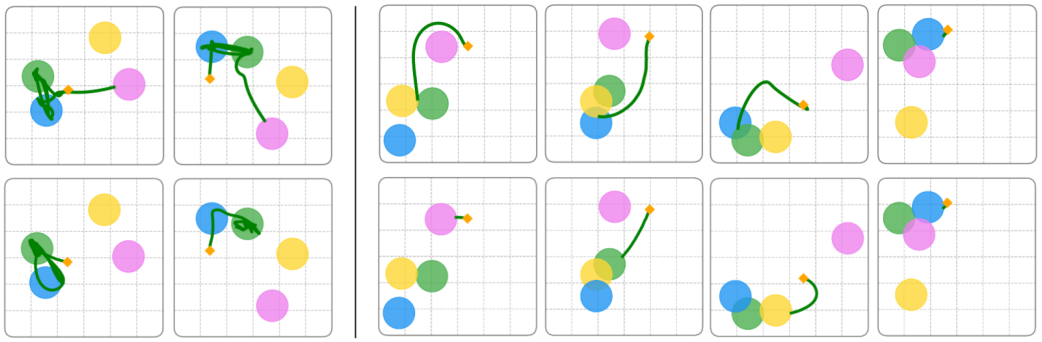

Figure 10: Illustration of trajectories in `ZoneEnv`: (1) unsatisfiable subgoals (left two columns, ϕ_1) and (2) overlapping areas (right four columns, ϕ_2–ϕ_5). The top row shows trajectories generated by our method, while the bottom row shows those from DeepLTL.

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 7, and example trajectories are shown in Figure 11. 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 7: 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 6. 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
$\eta_s \uparrow$	$0.96_{\pm 0.02}$	$0.94_{\pm 0.02}$	$0.91_{\pm 0.03}$	$0.93_{\pm 0.03}$
$\eta_v \downarrow$	$0.02_{\pm 0.01}$	$0.02_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.02_{\pm 0.01}$
$\eta_o \downarrow$	$0.02_{\pm 0.02}$	$0.04_{\pm 0.02}$	$0.06_{\pm 0.03}$	$0.05_{\pm 0.02}$
$\mu \downarrow$	$306.58_{\pm 16.88}$	$314.88_{\pm 17.02}$	$313.10_{\pm 10.61}$	$323.30_{\pm 12.57}$

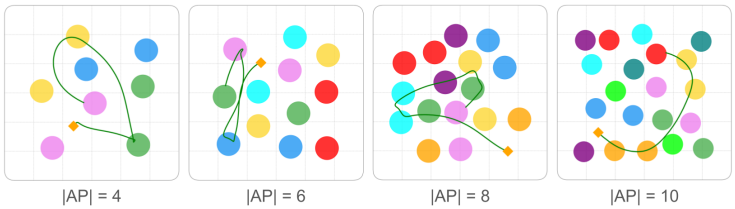

Figure 11: Illustration of `ZoneEnv` with an increasing number of atomic propositions. The trajectories shown correspond to the first specification for $|\text{AP}| = 4, 6, 8, 10$ in Table 6.

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 12. 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.

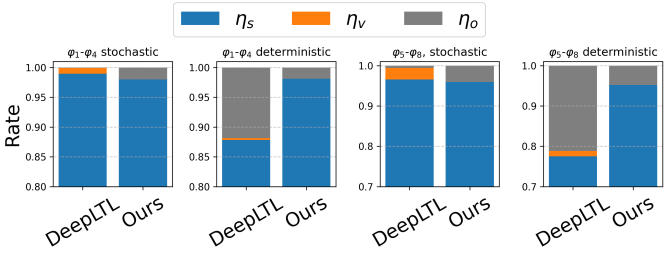

Figure 12: Evaluation results of DeepLTL and our method in `LetterWorld` using stochastic and deterministic policies. Each value is averaged over 5 seeds, with 100 trajectories per seed.

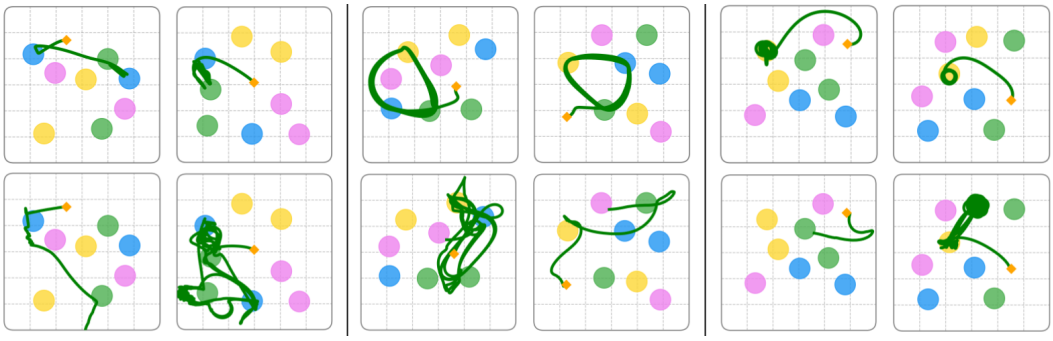

Figure 13: Illustration of infinite-horizon tasks in `ZoneEnv`. The trajectories correspond to specifications ψ_1–ψ_3, shown from left to right (two columns per specification). The top row shows trajectories generated by our method, while the bottom row shows those from DeepLTL.

D Visualizations

We provide visualizations for the experiments in Section 5.1 and Section 5.2, including infinite-horizon tasks (Figure 13), increasing environment complexity (Figure 14) and increasing numbers

of atomic propositions (Figure 11). 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.

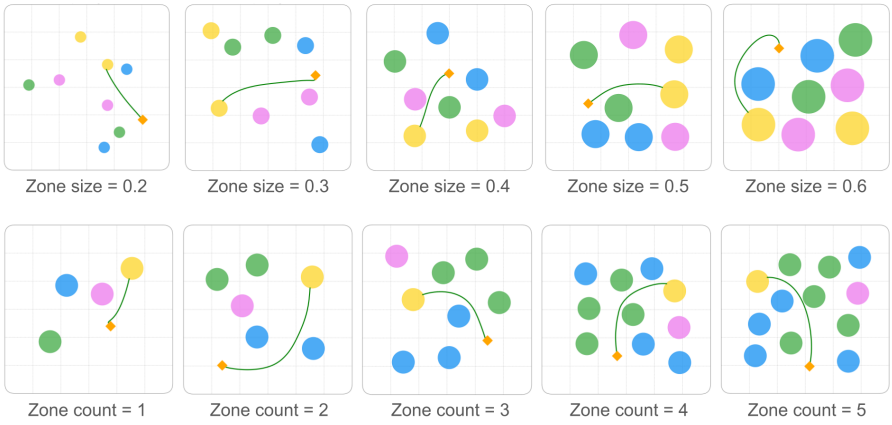

Figure 14: Illustration of `ZoneEnv` under increasing environment complexity, including different zone sizes and different numbers of zones per atomic proposition. The default zone size is 0.4, and the default number of zones per atomic proposition is 2. The trajectories are generated by our method and correspond to the evaluated specification $(\neg \text{blue} \wedge \neg \text{green})\ \mathsf{U}\ \text{yellow}$.