

On Generating Explanations for Reinforcement Learning Policies: An Empirical Study

Mikihsa Yuasa¹, Huy T. Tran¹, and Ramavarapu S. Sreenivas¹

Abstract— Explaining *reinforcement learning* policies is important for deploying them in real-world scenarios. We introduce a set of *linear temporal logic* formulae designed to provide such explanations, and an algorithm for searching through those formulae for the one that best explains a given policy. Our key idea is to compare action distributions from the target policy with those from policies optimized for candidate explanations. This comparison provides more insight into the target policy than existing methods and avoids inference of “catch-all” explanations. We demonstrate our method in a simulated game of capture-the-flag, a car-parking environment, and a robot navigation task.

Index Terms— Autonomous System, Intelligent Systems, Machine Learning, Robotics

I. INTRODUCTION

REINFORCEMENT learning (RL) has shown promising results for learning to autonomously make decisions. The integration of deep learning with RL, or *deep reinforcement learning* (DRL), has significantly advanced the field by, for example, surpassing human experts in games [1] and contributing to robotics [2]. However, the complexity of DRL models raises a significant concern: their lack of explainability [3]. As these systems advance to solve real-world problems, their decision-making becomes more opaque, making it difficult to understand the reasoning behind their choices. Such understanding could, for example, improve trust in a self-driving car, where a passenger might like to know why a car suddenly stopped, or be used by an engineer during the design process to verify that a navigation policy satisfies the desired specification.

A common approach to explaining complex policies, such as those produced by DRL, is to mine *temporal logic* (TL) explanations from observed system behaviors [4]. TL offers a formal structure and interpretable semantics for describing the evolution of system states over time. Traditionally, methods are based on a classification setup, where positive (from the target policy) and negative (from another policy) trajectories are provided, with various techniques used to optimize the classifier. For example, [5, 6] use directed acyclic graphs to simultaneously learn a *signal temporal logic* (STL) formula structure and associated parameters, while [7, 8] use tree-based algorithms. Neural networks have recently been used

to learn parameters of a given *weighted signal temporal logic* (wSTL) formula structure [9] and the structure itself [10]. However, since these methods focus on differentiating positive and negative data, their inferred explanations strongly depend on how the given data were generated. Other methods only require positive data [11, 12], but they remain biased by the given positive data. Non-TL methods have also been proposed to explain policies, including methods to identify important features [13, 14], identify important time steps [15], learn state abstractions [16], and explore counterfactuals [17], but these methods provide limited interpretability with respect to the overall objective of the policy. Inverse RL methods can infer policy objectives [18], but they simply identify feature weights and therefore cannot model temporal objectives captured by TL.

We take a different approach where we assume access to the target policy itself, rather than being given positive and/or negative data. The target policy could be accessible, for example, if one was optimizing a DRL policy for a complex task and wanted to gain insight into the underlying decision logic behind the policy to increase trust or support its verification in safety-critical applications. We are motivated by the idea that having access to the target policy will provide additional information that could help with the inference process. More specifically, we use action distributions from the target policy to gain insight into an agent’s preferences across actions for sampled states. This insight could be useful, for example, when an agent does not have a strong preference for the action taken in a given state, which could help differentiate between possible explanations for that policy. Inferring such preferences from a fixed dataset (i.e., without access to the policy itself) is challenging, particularly for stochastic policies and environments. These preferences could also be used to emphasize informative states during the inference process, which may not have been visited in a given set of trajectories.

Based on this idea, we propose a search method that infers the *linear temporal logic* (LTL) formula that best explains a target policy, where this explanation delineates the operational conditions maintained throughout execution and the ultimate objectives achieved. Our approach centers on a class of LTL formulae that is endowed with a concept of neighborhood that is amenable to a local-search (Sections III-A and III-C). Each potential LTL-explanation is then translated into an RL policy, which is compared to the target policy using a well-structured metric (Section III-B). Should a neighboring explanation have better alignment with the target policy, it supplants the current

¹The Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA. {myuasa2,huytran1,rsree}@illinois.edu

*This work was supported in part by ONR grant N00014-20-1-2249 and JASSO Study Abroad Support Program (Graduate Degree Program).

explanation. This iterative process persists until none of the neighboring explanations surpasses the present one, thus establishing a local optimum as the recommended explanation. To enhance robustness, we propose additional neighborhood expansion and extension heuristics (Section III-D), as well as a multi-start implementation of the search that generates the top- k candidate explanations. We demonstrate our method in three simulated environments (Section IV).

II. BACKGROUND

A. Reinforcement Learning

We model our problem as a *Markov decision process* (MDP) $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ [19], where \mathcal{S} is the state space, \mathcal{A} is the action space, and $\gamma \in [0, 1]$ is the discount factor. Then $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the state transition function and $r(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the reward function for states $s, s' \in \mathcal{S}$ and action $a \in \mathcal{A}$. Let $\pi(a|s) : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ be a policy for the agent. At time step t , the agent executes action $a_t \sim \pi(\cdot|s_t)$ given the current state s_t , after which the system transitions to state s_{t+1} and the agent receives reward $r(s_t, a_t, s_{t+1})$. The objective is to learn a policy that maximizes $\mathbb{E}_{p, \pi} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a]$.

B. Linear Temporal Logic and FSPA-Augmented MDPs

The syntax for LTL is defined as $\phi := \top \mid \psi(s) \mid \neg\phi \mid \phi \vee \phi' \mid \phi \wedge \phi' \mid \mathcal{G}(\phi) \mid \mathcal{F}(\phi) \mid \mathcal{U}(\phi) \mid \mathcal{X}(\phi)$ for logical formulae ϕ and ϕ' . Here, \top is the True Boolean constant, $s \in \mathcal{S}$ is an MDP state, $\psi(s) := f(s) < c$ is an atomic predicate for $c \in \mathbb{R}$, and \neg (negation), \wedge (conjunction), and \vee (disjunction) are Boolean connectives. \mathcal{F} (eventually), \mathcal{G} (globally), \mathcal{U} (until), and \mathcal{X} (next) are temporal operators. Given an LTL formula and a state or trajectory, we can assign a real-number value, the robustness, that reflects the degree to which the formula is satisfied. More formally, given a state $s \in \mathcal{S}$ and LTL formulae ϕ, ϕ' , the robustness of boolean operators is defined as,

$$\begin{aligned}\rho(s, f(s) < c) &= c - f(s), \\ \rho(s, \neg\phi) &= -\rho(s, \phi), \\ \rho(s, \phi \wedge \phi') &= \min(\rho(s, \phi), \rho(s, \phi')), \\ \rho(s, \phi \vee \phi') &= \max(\rho(s, \phi), \rho(s, \phi')).\end{aligned}$$

Given a trajectory of states $\tau := (s_0, s_1, \dots, s_k)$, the robustness of temporal operators is defined as,

$$\begin{aligned}\rho(\tau, \mathcal{F}(\phi)) &= \max_{i \in [0, k]} (\rho(s_i, \phi)), \\ \rho(\tau, \mathcal{G}(\phi)) &= \min_{i \in [0, k]} (\rho(s_i, \phi)).\end{aligned}$$

A feasible LTL formula can be transformed into a *finite state predicate automaton* (FSPA) using ω -automaton manipulation [20]. An FSPA \mathcal{A} is defined by $\langle \mathcal{Q}, \mathcal{S}, \mathcal{E}, \Phi, q_0, b, F, Tr \rangle$, where \mathcal{Q} is a finite set of automaton states, \mathcal{S} is an MDP state space, $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of edges (transitions) between automaton states, Φ is the input alphabet, $q_0 \in \mathcal{Q}$ is the initial automaton state, $b : \mathcal{E} \mapsto \Phi$ maps an edge (q_t, q_{t+1}) to its transition condition (defined as an LTL formula ϕ), $F \subseteq \mathcal{Q}$ is the set of final automaton states, and $Tr \subseteq \mathcal{Q}$ is the set of automaton trap states.

We now define an FSPA-augmented MDP $\mathcal{M}_{\mathcal{A}}$ based on [21]. Given MDP \mathcal{M} and FSPA \mathcal{A} , an FSPA-augmented MDP $\mathcal{M}_{\mathcal{A}}$ is defined by a tuple $\langle \tilde{\mathcal{S}}, \mathcal{Q}, \mathcal{A}, \tilde{p}, \tilde{r}, \mathcal{E}, \Psi, q_0, b, F, Tr \rangle$, where $\tilde{\mathcal{S}} \subseteq \mathcal{S} \times \mathcal{Q}$ is the product state space. Then $\tilde{p}(\tilde{s}'|\tilde{s}, a) : \tilde{\mathcal{S}} \times \mathcal{A} \times \tilde{\mathcal{S}} \mapsto [0, 1]$ is the state transition function and $\tilde{r}(\tilde{s}, a, \tilde{s}') : \tilde{\mathcal{S}} \times \mathcal{A} \times \tilde{\mathcal{S}} \mapsto \mathbb{R}$ is the reward function for states $\tilde{s}, \tilde{s}' \in \tilde{\mathcal{S}}$ and action $a \in \mathcal{A}$. We consider two rewards in this paper. For sparse reward settings, we define the reward \tilde{r} as,

$$\tilde{r}(\tilde{s}, a, \tilde{s}') = \begin{cases} 0 & q = q', \\ -\rho(s, b(q, q')) & q' \in Tr, \\ \rho(s, b(q, q')) & \text{otherwise,} \end{cases} \quad (1)$$

where $q, q' \in \mathcal{Q}$. For dense reward settings, we define the reward \tilde{r} as,

$$\tilde{r}(\tilde{s}, a, \tilde{s}') = \begin{cases} \beta \rho(s, b(q, q^*)) & q = q', \\ -\rho(s, b(q, q')) & q' \in Tr, \\ \rho(s, b(q, q')) & \text{otherwise,} \end{cases} \quad (2)$$

where $\beta \in [0, 1]$ is a scaling factor, $q^* = \arg \max_{q'' \in \mathcal{N}} \rho(s, b(q, q''))$, and $\mathcal{N} = \{q'' \in \mathcal{Q} \setminus Tr | (q, q'') \in \mathcal{E}\}$ is the set of non-trap automaton states that neighbors q . Intuitively, this dense reward encourages a transition to a non-trap automaton state if the next automaton state is the same as the current one.

III. METHOD

Our problem statement is as follows: given a target policy π_{tar} , MDP \mathcal{M} , and a set of atomic predicates Ψ , find an explanation ϕ that explains the objectives of the policy and the operational conditions maintained throughout execution. We address this problem by proposing a greedy local-search method, summarized in Figure 1 and Algorithms 1 and 2. We detail the key components of our method below.

A. Definition of Explanations

We consider explanations of the form $\phi = \mathcal{F}(\phi_F) \wedge \mathcal{G}(\phi_G)$, where ϕ_F and ϕ_G are *Conjunctive Normal Form* (CNF) or *Disjunctive Normal Form* (DNF) formulae that have up to two clauses within them. That is, we assume the policy tries to achieve a task represented by ϕ_F while satisfying global constraints represented by ϕ_G . For example, given atomic predicates $\Psi := \{\psi_0, \psi_1, \psi_2, \psi_3, \psi_4\}$, a possible explanation is $\mathcal{F}(\psi_0 \vee \psi_1) \wedge \mathcal{G}(\neg\psi_2 \wedge (\neg\psi_3 \vee \psi_4))$, which requires that: “*Eventually, either ψ_0 or ψ_1 is satisfied. Globally, ψ_2 is not satisfied and either ψ_3 is not satisfied or ψ_4 is satisfied.*”

We represent each explanation as a row vector of truth values whose length is $3N_{\text{pred}} + 2$, where $N_{\text{pred}} = |\Psi|$. The first N_{pred} elements define whether or not each predicate is negated (0 for no negation, 1 for negation). The next N_{pred} elements define which temporal formula, ϕ_F or ϕ_G , each predicate belongs to (0 for ϕ_F , 1 for ϕ_G). We require each temporal formula to contain at least one predicate. The following N_{pred} elements define which clause within the temporal formula each predicate belongs to (0 for the first clause, 1 for the second clause). The last two elements define whether each temporal formula, ϕ_F or ϕ_G , is in CNF or DNF form (0 for CNF, 1 for DNF).

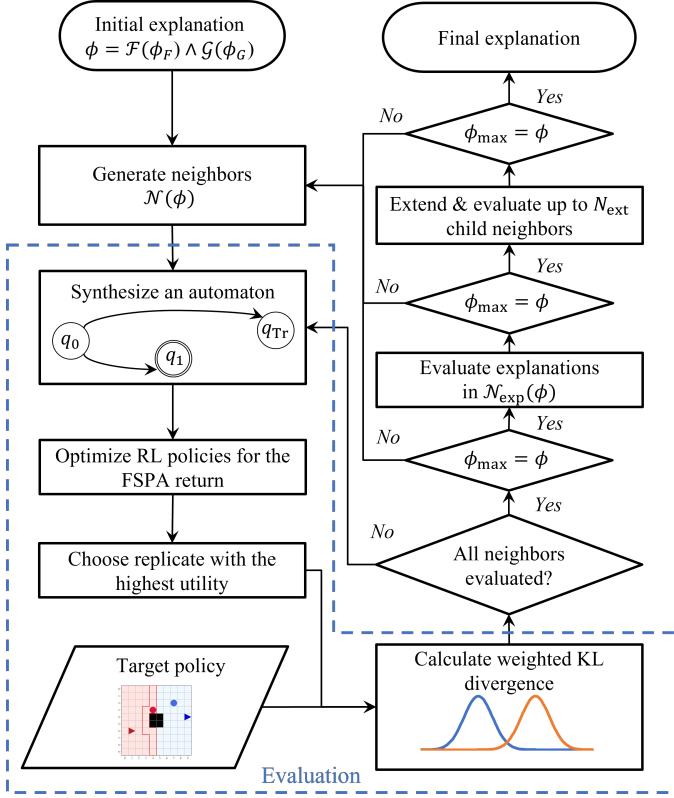


Fig. 1. Overview of our proposed search algorithm.

B. Evaluation of Explanations

We evaluate the utility of a candidate explanation by measuring the similarity between the target policy and a policy optimized for that LTL formula. A naive evaluation could simply generate a set of trajectories from the target policy and then count the trajectories that satisfy a formula. However, this approach would give high utility to “catch-all” explanations (e.g., those containing disjunctions of all predicates). We instead propose to measure utility as the weighted average of Kullback-Leibler (wKL) divergence values between the action distributions of the target policy and the policy optimized for a candidate explanation over sampled states. More specifically, we first synthesize an FSPA-augmented MDP $\mathcal{M}_{\mathcal{A}}$ using a candidate explanation ϕ and use RL to optimize a policy with respect to reward \tilde{r} (using Equation (1) or Equation (2)), following the method proposed by [21]. We address the fact that multiple optimal policies can exist by optimizing N_{rep} replicates and choosing the policy with the highest policy entropy H^{π} , calculated as,

$$H^{\pi} = \frac{-\sum_{s \in \mathcal{B}_{\text{NT}}} A_{\pi}(s) \log A_{\pi}(s)}{|\mathcal{B}_{\text{NT}}|}, \quad (3)$$

where $A_{\pi}(s) := \pi(\cdot|s)$ is the action distribution for policy π at state s and \mathcal{B}_{NT} is a set of randomly sampled non-trap states (i.e., states $s \in \mathcal{S}$ whose corresponding automaton state is in $\mathcal{Q} \setminus \mathcal{T}$).

We then measure the similarity between the selected policy and the target policy by calculating KL divergence values over action distributions of sampled states. More specifically, for a

selected policy π_{ϕ} , target policy π_{tar} , and state s , we calculate,

$$D_{\text{KL}}(A_{\pi_{\phi}}(s) || A_{\pi_{\text{tar}}}(s)) = A_{\pi_{\phi}}(s) \log \frac{A_{\pi_{\phi}}(s)}{A_{\pi_{\text{tar}}}(s)}. \quad (4)$$

If policies output multiple actions, the average of KL divergence values over action distribution pairs is used. Finally, we use Equation (4) to calculate the utility U of explanation ϕ as a wKL divergence value over states $s_i \in \mathcal{B}_{\text{NT}}$,

$$U^{\phi} = -\sum_{i=1}^{|\mathcal{B}_{\text{NT}}|} w_i D_{\text{KL}}(A_{\pi_{\phi}}(s_i) || A_{\pi_{\text{tar}}}(s_i)), \quad (5)$$

where w_i is the weight associated with s_i , calculated as,

$$w_i = \frac{\bar{H}^{\pi_{\text{tar}}}(s_i)}{\sum_{s_j \in \mathcal{B}_{\text{NT}}} \bar{H}^{\pi_{\text{tar}}}(s_j)}. \quad (6)$$

Here, $\bar{H}^{\pi_{\text{tar}}}(s)$ is a normalized entropy calculated as,

$$\bar{H}^{\pi_{\text{tar}}}(s) = 1 + \frac{A_{\pi_{\text{tar}}}(s) \log A_{\pi_{\text{tar}}}(s)}{H_{\max}}, \quad (7)$$

where H_{\max} is the maximum possible entropy. We use a weighted average to emphasize similarity between policies at states where the target policy is highly certain about what action to take. We include an augmented return filter to ignore any explanations that produce a policy with a converged return lower than a user-defined threshold R_{th} . This filter thus ignores explanations that are impossible to satisfy given the FSPA-augmented MDP.

C. Neighborhood Definition and Evaluation

After evaluating an explanation ϕ , our search proceeds by generating a neighborhood $\mathcal{N}(\phi)$ of related explanations. We define the neighborhood $\mathcal{N}(\phi)$ as the set of explanations whose row vector representations differ from that of ϕ at a single location (i.e., a single bit-flip). The class of LTL formulae we consider (i.e., of the form $\mathcal{F}(\phi_F) \wedge \mathcal{G}(\phi_G)$) are thus completely connected under this notion of a neighborhood. After generating $\mathcal{N}(\phi)$, we evaluate each of the neighboring explanations using the method discussed in Section III-B.

D. Additional Neighborhood Expansion & Extension

To address the potential existence of multiple undesired local optima, we include additional neighborhood expansion and extension steps in our search. The expansion step creates an expanded neighborhood $\mathcal{N}_{\text{exp}}(\phi)$ for a given parent explanation ϕ by flipping the values of the last two elements (i.e., the elements that define whether the temporal formulae are in CNF or DNF) of each explanation in the original neighborhood $\mathcal{N}(\phi)$. These additional explanations are then evaluated. This step is implemented if the neighborhood $\mathcal{N}(\phi)$ does not produce a better explanation than ϕ .

The extension step forces the search to generate and evaluate neighborhoods for the child explanations in $\mathcal{N}(\phi)$, even when the explanations in $\mathcal{N}(\phi)$ have a lower utility than ϕ . That is, we extend the search by evaluating neighbors of the explanations in $\mathcal{N}(\phi)$ with the highest utilities, up to N_{ext} times. This step is implemented if the original and extended neighborhoods of ϕ do not produce a better explanation than ϕ .

Algorithm 1 TL Greedy Local-Search

Input: number of searches N_{search} , number of maximum search steps N_{max} , number of replicates N_{rep} , number of sampled episodes N_{ep} , return filter threshold \tilde{R}_{th} , number of extension steps N_{ext}

Output: explanation ϕ

```

1: Construct empty buffer  $\Phi$ 
2: for  $n = 1, 2, \dots, N_{\text{search}}$  do
3:   Initialize starting explanation  $\phi$  randomly
4:   for  $m = 1, 2, \dots, N_{\text{max}}$  do
5:      $B = \text{EvalNeighbors}(\phi, N_{\text{rep}}, N_{\text{ep}}, \tilde{R}_{\text{th}})$ 
6:      $\phi_{\text{max}}, U_{\text{max}} = B[0]$ 
7:     if  $\phi = \phi_{\text{max}}$  then
8:       for  $i = 1, 2, \dots, N_{\text{ext}}$  do
9:          $\phi', U' \leftarrow B[i]$ 
10:         $B' = \text{EvalNeighbors}(\phi', N_{\text{rep}}, N_{\text{ep}}, \tilde{R}_{\text{th}})$ 
11:         $\phi'_{\text{max}}, U'_{\text{max}} \leftarrow B'[0]$ 
12:        if  $U'_{\text{max}} > U_{\text{max}}$  then
13:           $\phi, U \leftarrow \phi'_{\text{max}}, U'_{\text{max}}$ 
14:          break
15:        if  $\phi = \phi_{\text{max}}$  then break
16:        else  $\phi, U \leftarrow \phi_{\text{max}}, U_{\text{max}}$ 
17:      Store  $(\phi, U)$  to  $\Phi$ 
18:    return  $\phi$  from  $\Phi$  with the highest  $U$ 

```

Algorithm 2 EvalNeighbors()

Input: starting explanation ϕ_{in} , number of replicates N_{rep} , number of sampled episodes N_{ep} , reward filter threshold \tilde{R}_{th}

Output: Sorted buffer $B = [(\phi_i, U^{\phi_i})]$

```

1: Generate  $\mathcal{N}(\phi_{\text{in}})$  and construct sorted buffer  $B$ 
2: for  $\phi_i$  in  $\mathcal{N}(\phi_{\text{in}})$  do
3:   Synthesize automaton from  $\phi_i$ 
4:   Optimize and evaluate  $N_{\text{rep}}$  policies for  $\phi_i$ 
5:   Choose the policy with highest  $U^{\phi_i}$ 
6:   Calculate average return  $\bar{R}$  for  $N_{\text{ep}}$  episodes
7:   if  $\bar{R} > \tilde{R}_{\text{th}}$  then store  $(\phi_i, U^{\phi_i})$  to  $B$ 
8:  $\phi_{\text{max}}, U_{\text{max}} = B[0]$ 
9: if  $\phi_{\text{max}} = \phi_{\text{in}}$  then repeat Lines 2-7 for  $\mathcal{N}_{\text{exp}}(\phi_{\text{in}})$ 
10: return  $B$ 

```

IV. RESULTS

A. Test Environments and RL Training Details

We considered three test environments, shown in Figure 2. Capture-the-flag (CtF) is a discrete grid-world with adversarial dynamics, where agents try to capture their opponent's flag. If the agents are next to each other in the blue territory, the red agent is killed with 75% probability (vice versa in the red territory). We explained the blue agent policy and defined the red policy as a heuristic focused on defending its border region. We defined four atomic predicates based on euclidean distances between the blue agent (ba), blue flag (bf), blue territory (bt), red agent (ra), and red flag (rf): $\psi_{\text{ra},\text{bf}} = d_{\text{ra},\text{bf}} < 1$, $\psi_{\text{ba},\text{rf}} = d_{\text{ba},\text{rf}} < 1$, $\psi_{\text{ba},\text{ra}} = d_{\text{ba},\text{ra}} < 1.5$, $\psi_{\text{ba},\text{bt}} = d_{\text{ba},\text{bt}} < 1$. The resulting search space contained 640 explanations. Using the procedure from [21] and sparse

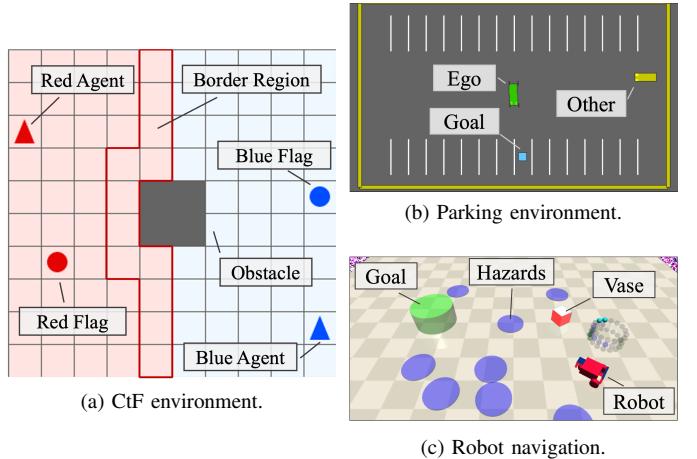


Fig. 2. Screenshots from our environments.

reward defined in Equation (1), we optimized the target policy to satisfy $\mathcal{F}(\psi_{\text{ba},\text{rf}} \wedge \neg\psi_{\text{ra},\text{bf}}) \wedge \mathcal{G}(\neg\psi_{\text{ba},\text{ra}} \vee \psi_{\text{ba},\text{bt}})$, which requires that: “*Eventually, the blue agent reaches the red flag and the red agent does not reach the blue flag. Globally, the blue agent does not encounter the red agent or it stays in the blue territory.*”

We used the car-parking scenario to consider continuous state-action spaces ($s \in \mathbb{R}^{19}$, $a \in \mathbb{R}^2$), based on a modified version of highway-env [22]. An episode ends when the ego vehicle reaches the landmark, hits a wall, or hits the other vehicle. We explained the ego vehicle policy and defined the other vehicle to use a heuristic policy that moves left or right based on a random initial location. We defined three atomic predicates: $\psi_g = d_{\text{ego},\text{goal}} < 1$, $\psi_o = d_{\text{ego},\text{other}} < 3$, $\psi_w = d_{\text{ego},\text{wall}} < 4$, where ego and other stand for the ego and other vehicles, respectively. The resulting search space contained 96 explanations. Using the dense reward defined in Equation (2), we optimized the target policy to satisfy $\mathcal{F}(\psi_g) \wedge \mathcal{G}(\neg\psi_o \wedge \neg\psi_w)$, which requires that: “*Eventually, the ego vehicle reaches the goal. Globally, the ego vehicle does not hit the other vehicle and walls.*”

We used a robot navigation task to consider more realistic and complex robot dynamics ($s \in \mathbb{R}^{72}$, $a \in \mathbb{R}^2$), based on a modified version of CarGoaly0 [23]. We explained the robot policy and defined three atomic predicates: $\psi_{\text{gl}} = d_{\text{robot},\text{goal}} < 0.3$, $\psi_{\text{hz}} = d_{\text{robot},\text{hazards}} < 0.2$, $\psi_{\text{vs}} = d_{\text{robot},\text{vase}} < 0.1$. We considered two target policies: one optimized for an LTL specification and one optimized for a non-LTL reward (to consider a target policy that was not optimized for one of the candidate explanations). Using the dense reward defined in Equation (2), we optimized the LTL target policy to satisfy $\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$, which requires that: “*Eventually, the robot reaches the goal. Globally, the robot does not enter a hazard or vase.*” We optimized the non-LTL policy using a reward based on distance to the goal and reaching the goal or a hazard.

We used StableBaselines3 [24] implementations of PPO [25] to optimize policies for CtF and robot navigation, and SAC [26] with HER [27] for parking. Hyperparameters

TABLE I

CTF & PARKING RESULTS. TARGET POLICIES WERE SUCCESSFULLY FOUND IN SEARCHES 1 AND 2 (CTF) AND 1, 2, AND 3 (PARKING).

Search	CtF explanations	wKL div. [-]	Searched specs [%]	Parking explanations	wKL div. [-]	Searched specs [%]
1	$\mathcal{F}(\psi_{ba,rf} \wedge \neg\psi_{ra,bf}) \wedge \mathcal{G}(\neg\psi_{ba,ra} \vee \psi_{ba,bt})$	8.00×10^{-8}	8.13	$\mathcal{F}(\psi_g) \wedge \mathcal{G}(\neg\psi_o \wedge \neg\psi_w)$	0.00	37.5
2	$\mathcal{F}(\psi_{ba,rf} \wedge \neg\psi_{ra,bf}) \wedge \mathcal{G}(\neg\psi_{ba,ra} \vee \psi_{ba,bt})$	8.00×10^{-8}	10.2	$\mathcal{F}(\psi_g) \wedge \mathcal{G}(\neg\psi_o \wedge \neg\psi_w)$	0.00	29.2
3	$\mathcal{F}(\neg\psi_{ba,bt}) \wedge \mathcal{G}((\neg\psi_{ba,ra} \wedge \psi_{ba,rf}) \vee (\neg\psi_{ra,bf}))$	1.46×10^{-7}	6.56	$\mathcal{F}(\psi_g) \wedge \mathcal{G}(\neg\psi_o \wedge \neg\psi_w)$	0.00	37.5
4	$\mathcal{F}(\psi_{ba,rf}) \wedge \mathcal{G}((\neg\psi_{ba,bt} \wedge \neg\psi_{ra,bf}) \vee (\neg\psi_{ba,ra}))$	7.42×10^{-7}	8.44	$\mathcal{F}(\psi_o) \wedge \mathcal{G}(\neg\psi_g \vee \neg\psi_w)$	4.61×10^{-4}	44.8
5	$\mathcal{F}(\neg\psi_{ba,bt} \wedge \neg\psi_{ra,bf}) \wedge \mathcal{G}((\neg\psi_{ba,rf}) \vee (\neg\psi_{ba,ra}))$	1.57×10^{-6}	6.56	$\mathcal{F}(\psi_o) \wedge \mathcal{G}(\neg\psi_g \vee \neg\psi_w)$	4.61×10^{-4}	33.3
6	$\mathcal{F}((\neg\psi_{ra,bf}) \vee (\psi_{ba,rf})) \wedge \mathcal{G}(\neg\psi_{ba,ra} \vee \psi_{ba,bt})$	5.95×10^{-6}	9.06	$\mathcal{F}(\psi_w) \wedge \mathcal{G}(\neg\psi_g \vee \psi_o)$	6.40×10^{-4}	30.2
7	$\mathcal{F}(\psi_{ba,ra}) \wedge \mathcal{G}((\neg\psi_{ba,bt} \wedge \neg\psi_{ra,bf}))$	1.18×10^{-5}	8.59	$\mathcal{F}(\neg\psi_o \wedge \psi_w) \wedge \mathcal{G}(\neg\psi_g)$	7.35×10^{-4}	28.1
8	$\mathcal{F}((\psi_{ba,ra}) \wedge (\neg\psi_{ba,rf})) \wedge \mathcal{G}((\psi_{ba,bt}) \vee (\neg\psi_{ra,bf}))$	1.18×10^{-5}	11.4	$\mathcal{F}(\neg\psi_o \wedge \psi_w) \wedge \mathcal{G}(\neg\psi_g)$	7.35×10^{-4}	27.1
9	$\mathcal{F}((\psi_{ra,bf}) \vee (\neg\psi_{ba,bt})) \wedge \mathcal{G}((\neg\psi_{ba,ra}) \vee (\psi_{ba,rf}))$	1.62×10^{-5}	7.19	-	-	-
10	$\mathcal{F}((\neg\psi_{ba,ra}) \vee (\neg\psi_{ra,bf})) \wedge \mathcal{G}((\psi_{ba,bt}) \vee (\psi_{ba,rf}))$	8.52×10^{-5}	6.56	-	-	-

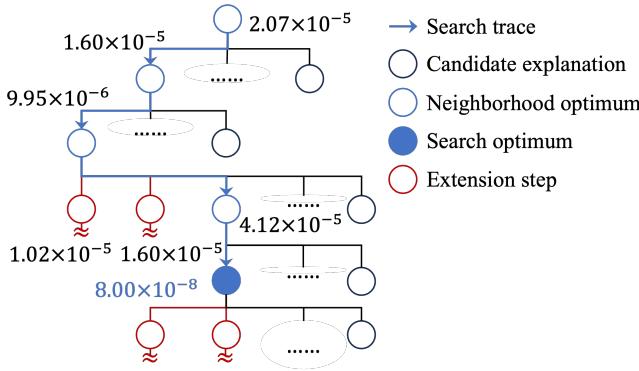


Fig. 3. A partial trace of CtF Search 5 with wKL divergence values.

were default ones or those suggested by RL Baselines3 Zoo [28]. We used the following search hyperparameters, based on a small trial-and-error process: $N_{\text{max}} = 10$, $N_{\text{rep}} = 3$ (CtF, parking) or 1 (navigation), $N_{\text{ep}} = 200$, $\tilde{R}_{\text{th}} = 0.05$, and $N_{\text{ext}} = 3$. Our code is available online¹.

B. Experiment Results

Table I shows our CtF results, where Searches 1 and 2 successfully found the target explanation as the one with the highest utility among searched explanations. The second best result was Search 6, whose explanation is: “Eventually, the blue agent captures the red flag. Globally, the blue agent is not inside the blue territory while the red agent does not capture the blue flag, or the blue agent does not encounter the red agent.” This explanation is close to the target explanation and consistent with the game dynamics since the task includes the overall objective of capturing the red flag, while the constraint plausibly captures the battle dynamics of CtF which could incentivize the blue agent to avoid the red agent if it is outside of the blue territory, but engage the red agent within the blue territory. Figure 3 shows a partial trace of the search tree produced by Search 2. We see that the extension step was used in the third step of the search to avoid an undesired local minimum early in the process. Though not shown, the extension step was also used in this search.

Table I also shows our parking results, where Searches 1-3 successfully found the target explanation. The second best

results were Searches 4 and 5, whose explanation is: “Eventually, the ego vehicle encounters the other agent. Globally, the ego vehicle does not reach the goal or does not hit the wall.” This explanation captures a failure case, where the ego vehicle sometimes does not reach the goal because it collides with the other vehicle, though it does successfully avoid walls. This result suggests that our explanations can capture unlikely, but observed, behaviors from the target policy that occur due to environment stochasticity. We also performed an ablation study, where we independently removed the extension step, expansion step, and weights in calculating KL divergence (i.e., we set $w_i = 1$ in Equation (6)), and found that none of the modified versions of our method successfully found the target explanation.

Table II shows our robot navigation results. For the LTL target policy, our method found the target explanation in Searches 1-8. For the non-LTL target policy, our method identified the best explanation as: “Eventually, the robot reaches the goal or does not hit the vase. Globally, the robot does not enter a hazard” (Search 1). While the task part of the explanation may seem unexpected, it is actually reasonable because the target policy did not always reach the goal (the success rate was around 60%) and it occasionally hit the vase (since it was not penalized for doing so) — since our method has no mechanism for ignoring predicates, the vase predicate ended up in the task clause. The constraint part of the explanation was expected, since the target policy learned to avoid hazards. This result suggests that our method can find a reasonable explanation, even for policies not optimized for a candidate explanation.

V. CONCLUSIONS

This paper presents a method for generating an explanation of a given RL policy using a connected class of LTL formulae. We employ a local-search algorithm that identifies the explanation which produces a policy that best matches the target policy, based on a weighted KL divergence metric. We verified our approach in three simulated environments. Limitations include computationally complexity (we scale exponentially with the number of predicates and require an RL optimization for searched candidates), requiring user-defined predicates, interpreting inferred LTL formulas, and assuming stationary, near-optimal policies. Potential improvements are automating

¹<https://github.com/miki-yuasa/tl-search>

TABLE II
ROBOT NAVIGATION RESULTS FOR LTL (LEFT) & NON-LTL (RIGHT) TARGET POLICIES.

Search	Robot explanations	wKL div. [-]	Searched specs [%]	Robot explanations	wKL div. [-]	Searched specs [%]
1	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	21.8	$\mathcal{F}(\psi_{\text{gl}} \vee \neg\psi_{\text{vs}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}})$	3.0501×10^{-4}	39.6
2	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	30.2	$\mathcal{F}(\neg\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \vee \neg\psi_{\text{vs}})$	3.0504×10^{-4}	29.2
3	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	26.0	$\mathcal{F}(\neg\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \vee \neg\psi_{\text{vs}})$	3.0504×10^{-4}	27.1
4	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	29.2	$\mathcal{F}(\neg\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \vee \neg\psi_{\text{vs}})$	3.0504×10^{-4}	27.1
5	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	21.9	$\mathcal{F}(\neg\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \vee \neg\psi_{\text{vs}})$	3.0504×10^{-4}	29.2
6	$\mathcal{F}(\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \wedge \neg\psi_{\text{vs}})$	0.00	24.0	$\mathcal{F}(\neg\psi_{\text{gl}}) \wedge \mathcal{G}(\neg\psi_{\text{hz}} \vee \neg\psi_{\text{vs}})$	3.0504×10^{-4}	33.3
7	$\mathcal{F}(\neg\psi_{\text{vs}}) \wedge \mathcal{G}(\psi_{\text{gl}} \vee \neg\psi_{\text{hz}})$	2.64×10^{-4}	25.0	$\mathcal{F}(\psi_{\text{gl}} \vee \psi_{\text{hz}}) \wedge \mathcal{G}(\neg\psi_{\text{vs}})$	3.0508×10^{-4}	29.2
8	$\mathcal{F}(\neg\psi_{\text{vs}}) \wedge \mathcal{G}(\psi_{\text{gl}} \vee \neg\psi_{\text{hz}})$	2.64×10^{-4}	27.1	$\mathcal{F}(\psi_{\text{gl}} \vee \psi_{\text{hz}}) \wedge \mathcal{G}(\neg\psi_{\text{vs}})$	3.0508×10^{-4}	21.9

predicate selection, translating formulae to natural language, integrating into RL algorithms for explainability-by-design, scaling through neural network LTL representations, reducing computation through transfer learning or optimization landscape exploration, and bounding the number of required searches.

REFERENCES

- [1] D. Silver *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 7, 2018.
- [2] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, “Transferring policy of deep reinforcement learning from simulation to reality for robotics,” *Nat. Mach. Intell.*, vol. 4, no. 12, pp. 1077–1087, 2022.
- [3] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, “Explainability in deep reinforcement learning,” *Knowledge-Based Syst.*, vol. 214, p. 106 685, 28, 2021.
- [4] E. Bartocci, C. Mateis, E. Nesterini, and D. Nickovic, “Survey on mining signal temporal logic specifications,” *Inf. and Computation*, vol. 289, p. 104 957, 1, 2022.
- [5] J.-R. Gaglione, D. Neider, R. Roy, U. Topcu, and Z. Xu, “Learning linear temporal properties from noisy data: A MaxSAT-based approach,” in *Automated Technology for Verification and Analysis*, Z. Hou and V. Ganesh, Eds., Cham: Springer Int. Publishing, 2021, pp. 74–90.
- [6] Z. Kong, A. Jones, and C. Belta, “Temporal logics for learning and detection of anomalous behavior,” *IEEE Trans. Automat. Control*, vol. 62, no. 3, pp. 1210–1222, 2017.
- [7] G. Bombara and C. Belta, “Offline and online learning of signal temporal logic formulae using decision trees,” *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 3, 22:1–22:23, 2021.
- [8] R. Karagulle, N. Aréchiga, A. Best, J. DeCastro, and N. Ozay, “A safe preference learning approach for personalization with applications to autonomous vehicles,” *IEEE Robot. Automat. Lett.*, vol. 9, no. 5, pp. 4226–4233, 2024.
- [9] R. Yan, A. Julius, M. Chang, A. Fokoue, T. Ma, and R. Uceda-Sosa, “STONE: Signal temporal logic neural network for time series classification,” in *2021 Int. Conf. Data Mining Workshops (ICDMW)*, 2021, pp. 778–787.
- [10] D. Li, M. Cai, C.-I. Vasile, and R. Tron, “Learning signal temporal logic through neural network for interpretable classification,” in *2023 Amer. Control Conf. (ACC)*, 2023, pp. 1907–1914.
- [11] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, “TeLEX: Passive STL learning using only positive examples,” in *Runtime Verification*, S. Lahiri and G. Reger, Eds., Cham: Springer Int. Publishing, 2017, pp. 208–224.
- [12] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, “TeLEX: Learning signal temporal logic from positive examples using tightness metric,” *Formal Methods Syst. Des.*, vol. 54, no. 3, pp. 364–387, 1, 2019.
- [13] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.
- [14] M. T. Ribeiro, S. Singh, and C. Guestrin, ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA: ACM, 2016, pp. 1135–1144.
- [15] W. Guo, X. Wu, U. Khan, and X. Xing, “EDGE: Explaining Deep Reinforcement Learning Policies,” in *35th Conf. on Neural Inf. Process. Syst. (NeurIPS 2021)*, 2021.
- [16] T. Bewley and J. Lawry, “TripleTree: A Versatile Interpretable Representation of Black Box Agents and their Environments,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 415–11 422.
- [17] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, “Explainable reinforcement learning through a causal lens,” in *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 2493–2500. arXiv: [1905.10958](https://arxiv.org/abs/1905.10958).
- [18] S. Arora and P. Doshi, “A survey of inverse reinforcement learning: Challenges, methods and progress,” *Artificial Intelligence*, vol. 297, p. 103 500, 2021. arXiv: [1806.06877](https://arxiv.org/abs/1806.06877).
- [19] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial Intell.*, vol. 112, no. 1, pp. 181–211, 1, 1999.
- [20] K. Schneider, “Improving automata generation for linear temporal logic by considering the automaton hierarchy,” in *Proc. Artificial Intell. Logic Program.*, ser. LPAR ’01, Berlin, Heidelberg: Springer-Verlag, 2001, pp. 39–54.
- [21] X. Li, Z. Serlin, G. Yang, and C. Belta, “A formal methods approach to interpretable reinforcement learning for robotic planning,” *Science Robot.*, vol. 4, no. 37, eaay6276, 18, 2019.
- [22] E. Leurent, *An environment for autonomous driving decision-making*, <https://github.com/eleurent/highway-env>, 2018.
- [23] J. Ji *et al.*, “Safety gymnasium: A unified safe reinforcement learning benchmark,” in *37th Conf. Neural Inf. Process. Syst. Datasets and Benchmarks Track*, 2023.
- [24] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.
- [25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 28, 2017. arXiv: [1707.06347\[cs\]](https://arxiv.org/abs/1707.06347).
- [26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. of the 35th Int. Conf. Mach. Learn.*, PMLR, 3, 2018, pp. 1861–1870.
- [27] M. Andrychowicz *et al.*, “Hindsight experience replay,” in *Advances in Neural Inf. Process. Syst.*, vol. 30, Curran Associates, Inc., 2017.
- [28] A. Raffin, *Rl baselines3 zoo*, <https://github.com/DLR-RM/r1-baselines3-zoo>, 2020.