IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Reinforcement Learning of Action and Query Policies with LTL Instructions under Uncertain Event Detector

Wataru Hatanaka[1,2], Ryota Yamashina[1], and Takamitsu Matsubara[2]

***Abstract*—Reinforcement learning (RL) with linear temporal logic (LTL) objectives can allow robots to carry out symbolic event plans in unknown environments. Most existing methods assume that the event detector can accurately map environmental states to symbolic events; however, uncertainty is inevitable for real-world event detectors. Such uncertainty in an event detector generates multiple branching possibilities on LTL instructions, confusing action decisions. Moreover, the queries to the uncertain event detector, necessary for the task's progress, may increase the uncertainty further. To cope with those issues, we propose an RL framework, Learning Action and Query over Belief LTL (LAQBL), to learn an agent that can consider the diversity of LTL instructions due to uncertain event detection while avoiding task failure due to the unnecessary event-detection query. Our framework simultaneously learns 1) an embedding of belief LTL, which is multiple branching possibilities on LTL instructions using a graph neural network, 2) an action policy, and 3) a query policy that decides whether or not to query for the event detector. Simulations in a 2D grid world and image-input robotic inspection environments show that our method successfully learns actions to follow LTL instructions even with uncertain event detectors.**

***Index Terms*—Reinforcement Learning, Planning under Uncertainty**

I. INTRODUCTION

SERVICE robots are expected to reduce human workload and improve the quality of human life in indoor and outdoor fields. These robots must follow diverse instructions and perform stable, even in long-horizon tasks, to coexist with human society. Learning these abilities, especially under the partial observability of the real world, is one of the challenges of co-working with humans autonomously.

Linear Temporal Logic (LTL) [1], a formal language that captures the temporal property of the task as a symbolic event representation, is widely used to construct systems that satisfy desired specifications. In recent years, reinforcement learning (RL) to maximize the probability of satisfying LTL instructions allows an agent to follow various instructions even when the environmental model is unknown [2]–[7]. However, the success of these methods is based on the unrealistic assumption that the event detector, which informs the agent of events occurring in the environment, does not fail to detect events and that the agent is always aware of the task's progress.

[1]Wataru Hatanaka and Ryota Yamashina are with Digital Strategy Division, RICOH Company,Ltd, Japan. {wataru.hatanaka, ryohta.yamashina}@jp.ricoh.com

[2]Wataru Hatanaka and Takamitsu Matsubara are with Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology (NAIST), Japan. takam-m@is.naist.jp

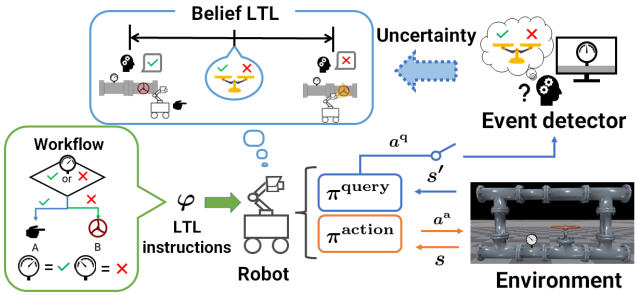


Fig. 1: Overview of our proposed framework. A robot has a query policy that controls the interaction with an event detector in addition to an action policy. Our method learns both policies that act according to the belief over LTL instructions due to uncertainty in the event detector.

When assuming uncertainty in event detection, we must face the risk of failure caused by giving the detection possibility to an event that has not occurred. For example, as shown in Fig. 1, consider the situation when the inspection result contains uncertainty in a workflow that branches depending on whether the equipment is normal or abnormal. A gradual evaluation according to the confidence or prediction score of event detection is a common approach to convey uncertainty in decision-making to the agent. However, believing only the most likely event, employed by some methods that deal with probabilistic event detectors [8], [9], may fail the task with half the probability. To avoid this, the agent needs to respond to different levels of uncertainty: if the uncertainty is low, the agent will believe in its judgment and act; otherwise, it will act conservatively. Furthermore, the agent may suffer from this uncertainty each time it queries the event detector; unnecessary queries may lead to a misunderstanding of task progress, leading to an increasingly high probability of task failure. On the other hand, since querying the event detector is essential for the agent to know when the task is accomplished, controlling the query to the event detector is required to handle this trade-off.

We propose an RL framework, Learning Action and Query over Belief LTL (LAQBL), to learn an agent that can consider the diversity of LTL instructions due to uncertain event detection while avoiding task failure due to the unnecessary event-detection query. Our framework simultaneously learns three components: 1) *embeddings of belief LTL* that embed belief LTL, a probabilistic representation of multiple LTL instructions caused by the uncertainty in the event detector, 2) *action policy*, and 3) *query policy* which decides whether

or not to query for the event detector, with that embedding as input. We exploit LTL progression for task transition and the belief LTL is embedded by a graph neural network (GNN) [7] to obtain non-myopic policy and generalisability to unknown LTL formulas. Our method is evaluated on the navigation in a grid world and the pipe inspection with high-dimensional image inputs under LTL instructions that require the agent to branch its behavior according to the subtask it achieves. Experimental results on long-horizon tasks that require action according to event detection uncertainties show that our agent outperforms a method without handling the uncertainty and achieves the best performance even when unnecessary queries to the event detectors lead to task failure.

Our main contributions are summarized as follows:

- Formulation for optimizing an agent that acts according to LTL instructions with uncertain event detectors;
- Proposal of a model for embedding multiple LTL instructions with their belief as belief LTL, inspired by LTL embedding in [7];
- Proposal of the LAQBL framework that learns the belief LTL embeddings, action, and query policies with the embeddings as input by reinforcement learning;
- Empirical evaluations in navigation with a 2D grid-world and image-input robotic inspection simulation environments.

II. RELATED WORKS

A. Multiple LTL Instructions for RL Agent

To our knowledge, there are two frameworks for instructing various LTL instructions to an agent: Reward Machines (RM) [3], [10], and LTL embedding [5]–[7]. RM provides automata representation constructed from symbolic task plans and enables designing the reward function, including non-Markovian rewards. While RM has the flexibility to represent symbolic tasks implemented by various formal languages, generalizing to unknown tasks is difficult because no information about the task structure is embedded in its state. The LTL embedding gives the agent a state that embeds part or the entire LTL instruction instead of defining explicit task states. This approach successfully generalizes to unknown LTL instructions by learning the embedding function of LTL formulas. However, most of these methods assume that the agent can detect the event that occurred in the environment and accurately track task progress.

B. Planning under Uncertain LTL Instructions

Previous works have considered addressing various uncertain situations encountered when implementing symbolic task representations in the real world to mitigate this assumption [11]–[14]. Particularly papers dealing with uncertainty regarding event detection are related to our proposal [8], [15]–[19]. The method proposed in [8] is similar to ours of modeling a belief over multiple LTL formulas for learning through RL. The belief is used to define reward functions for four different objectives in this method, while it is not assumed that a given belief will be updated. The main difference between ours and all these methods is that they do not provide a method to capture the uncertainty in the event detector; they cannot learn to act according to it. Table I summarizes the differences in belief models between our method and conventional.

Methods	Support of belief	Belief update trigger
Sharan et al. [15]	States	No update
Bouton et al. [16]	States	Observation
Hashimoto et al. [17]	States	Event detection
RMSM [19]	Automaton states	Direct learning
PUnS [8]	LTL formulas	No update
Ours	LTL formulas	Event detection

TABLE I: A comparison of modeling belief and its update trigger.

III. PRELIMINARIES

A. Linear Temporal Logic

LTL formula consists of a finite number of propositional symbols $\mathcal{P}$, boolean operators to connect propositions such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), and temporal operators to express the order of propositions such as $\cup$ (until), $\bigcirc$ (next), $\Diamond$ (eventually). For instance, the sequence "go to the kitchen and then go to the bedroom" can be described as $\varphi = \Diamond(\texttt{Kitchen} \wedge \Diamond\texttt{Bedroom})$. In this paper, we consider co-safe LTL (sc-LTL) [20], a subclass of LTL that deals with sequences of finite length, and describe sc-LTL simply as LTL. Detailed LTL syntax can be found in [21].

B. LTL-guided Policy for Taskable MDP

In common, the interaction of an agent with the environment is modeled by a Markov Decision Process (MDP) and defined by the following tuple $\mathcal{M} = (S, T, A, p, R, \gamma, \mu)$, where S is a set of states, $T \subseteq S$ is a set of terminal states, A is a set of actions, $P : S \times A \times S \to [0, 1]$ is the state transition function, $R : S \times A \to \mathbb{R}$ is a reward function, $\gamma \in (0, 1)$ is a discount factor, $\mu : S \to [0, 1]$ is a distribution over initial states.

To obtain the agent that satisfies a given LTL instruction φ through RL, the LTL task's progress and the environmental state must be managed simultaneously at every time step. We define a word $\sigma_t \in \{0, 1\}^{\mathcal{P}}$, a vector of propositions satisfied at time step t, and introduce a labeling function $L : S \to \{0, 1\}^{\mathcal{P}}$ that maps a state to the word σ_t. For RL with the LTL objective, the agent can be motivated to follow the LTL instruction φ by rewarding that φ holds in the finite trace $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_n)$, which is the mappings of the generated trajectory by the agent: $s_0, a_0, \cdots, s_n, a_n$. However, the reward function $R_\varphi(s_1, a_1, \ldots, s_t, a_t)$ is conditioned on a past trajectory and actions and becomes non-Markovian [2].

In recent works, the LTL progression is used to make the reward function Markovian [2], [7], which implements the syntactic progression rule (see details in [22]). In brief, the LTL progression diminishes an LTL formula by leaving propositions unsatisfied while preserving the original semantics and returns `true` if the LTL formula is completed and `false` if it is violated. For example, when an agent has the LTL instruction $\varphi = \neg\texttt{a} \cup \texttt{b}$ at time step t, φ is overwritten $\varphi := \mathrm{prog}(L(s_{t+1}), \varphi) = \texttt{true}$ by the progression if the event `b` is detected at $t + 1$, $\varphi := \texttt{false}$ if the event `a` is detected, and does not change if any other event is detected. *Taskable MDP* proposed in [7] is a model that allows the agent to follow

various LTL instructions while preserving Markov property by using the LTL progression and is defined as follows.

Definition 1 (*Taskable MDP* [7])**.** *Given a MDP without a reward function as a tuple $\mathcal{M} = (S, T, A, p, \gamma, \mu)$, a finite set of propositions $\mathcal{P}$, a labeling function L, a finite set of LTL formulas Φ and its probability distribution τ over Φ, a Taskable MDP is defined as a tuple $\mathcal{M}_\Phi = (S', T', A, p', R', \gamma, \mu')$, where $S' = S \times cl(\Phi)$ is a finite set of product states and $cl(\Phi)$ is the smallest set containing Φ that consists of $\varphi \in \Phi$ and its progression. $T' = \{\langle s, \varphi\rangle | s \in T' \text{ or } \varphi \in \{\mathsf{true}, \mathsf{false}\}\}$ is a terminal set of product states, p' is a transition probability of product states; $p'(\langle s', \varphi'\rangle \mid \langle s, \varphi\rangle, a) = p(s' \mid s, a)$ if $\varphi' = \mathrm{prog}(L(s), \varphi)$ otherwise zero, $\mu'(\langle s, \varphi\rangle) = \mu(s) \cdot \tau(\varphi)$ is an initial distribution of product states, and a reward function*

$$R'((s,\varphi),a) = \begin{cases} 1 & \text{if } \mathrm{prog}(L(s),\varphi) = \mathsf{true} \\ -1 & \text{if } \mathrm{prog}(L(s),\varphi) = \mathsf{false}\,. \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The LTL formula is mapped to the LTL feature space $\mathcal{E} \subseteq \mathbb{R}^n$ through the embedding $E_{LTL} : cl(\Phi) \to \mathcal{E}$, which is then given to the policy $\pi : S \times \mathcal{E} \times A \to [0, 1]$.

IV. Problem formulation

A. Modeling Event Detector and Belief over LTL formulas

We first replace a labeling function L, which deterministically provides the agent with the true events depending on the state, with a probabilistic event detector with uncertainty. In this paper, we model the uncertainty of the event detector as a function that returns probabilities for the agent satisfying events for a given state as follows.

Definition 2 (*Uncertainty of the event detector*)**.** *Given a state s_t in time step t, a finite set of propositions $\mathcal{P}$ and an LTL formula φ, the uncertainty of event detector is a probability distribution with support over $\mathcal{P}' = \mathcal{P} \cup \{\}$, where $\{\}$ denote a null proposition, with the probability mass function $O : S \times \{0, 1\}^{\mathcal{P}'} \to [0, 1]$, where $O(p|s_t)$ represents the confidence of the proposition that p holds* true *in a state s_t and $\sum_{p \in \mathcal{P}'} O(p|s_t) = 1, \forall s \in S$.*

Then, we define *possible LTLs*, a set of LTL formulas that can progress from the original LTL instruction by events for which the event detector has uncertainty, and a belief of possible LTL, as follows.

Definition 3 (*Possible LTLs and a belief of possible LTLs*)**.** *Given a state s_t in time step t and the uncertainty of event detector $O(p|s_t)$, the possible LTLs is defined as a finite set of LTL formulas $\Psi = \bigcup_{p^o \in \mathcal{P}^o} \{\mathrm{prog}(\sigma^{p^o}, \varphi)\}$ which consists of $\varphi \in \Phi$ and its progressed only propositions with positive probability $\mathcal{P}^o = \{p | O(p|s_t) > 0, p \in \mathcal{P}'\}$. A belief of the possible LTLs is defined as a probability distribution with support over the possible LTLs $B = \{b : \Psi \to [0, 1] \mid \Psi \subset cl(\Phi), \sum_{\psi \in \Psi} b(\psi) = 1\}$. The expansion of the belief in the next time step s_{t+1} is described as follows:*

$$b_{t+1}(\psi') = \sum_{p^o \in \mathcal{P}^o, \psi \in \Psi} O(p^o|s_{t+1}) b_t(\psi) \mathbb{1}[\mathrm{prog}(\sigma^{p^o}, \psi) = \psi']. \quad (2)$$

Example of updating the belief of possible LTLs. Given a LTL formula: $\varphi = \Diamond(\mathtt{a} \wedge \Diamond\mathtt{b}) \vee \Diamond(\mathtt{c} \wedge \Diamond\mathtt{d})$, which means "go to a and then b, or go to c and then d". Assume that the belief $b_t(\varphi) = 1$ and the uncertainty of the event detector in state s_{t+1} are $O(\mathtt{a}|s_{t+1}) = 0.8$, $O(\mathtt{c}|s_{t+1}) = 0.2$ and otherwise 0. The propositions that may have probability are a and c, and the possible LTLs are progressed by P^o: $\Psi = \{\varphi_\mathtt{a} = \mathrm{prog}(\sigma^\mathtt{a}, \varphi) = \Diamond\mathtt{b} \vee \Diamond(\mathtt{c} \wedge \Diamond\mathtt{d}), \varphi_\mathtt{c} = \mathrm{prog}(\sigma^\mathtt{c}, \varphi) = \Diamond(\mathtt{a} \wedge \Diamond\mathtt{b}) \vee \Diamond\mathtt{d}\}$. Then, the belief of the possible LTLs is updated by using their probabilities $b_{t+1}(\varphi_\mathtt{a}) = 0.8, b_{t+1}(\varphi_\mathtt{c}) = 0.2$.

B. Belief Taskable MDP

To model the process of the robot interacting with the environment and the event detector based on the belief, we introduce a query policy $\pi^\mathrm{q} : S \times \mathcal{E}_b \times A_\mathrm{q} \to [0, 1]$ that optimizes the query timing to the detector in addition to the action policy $\pi^\mathrm{a} : S \times \mathcal{E}_b \times A_\mathrm{a} \to [0, 1]$, where $\mathcal{E}_b$ is the feature space of the possible LTLs and its belief mapped by the embedding $E_{Belief} : \Psi \times B \to \mathcal{E}_b$. Here $A_\mathrm{q} = \{0, 1\}$, and we assume that the agent can only get uncertainty about propositions from the event detector when the query action sampled by the query policy is $a_\mathrm{q} = 1$, while the event detector returns a null proposition $O(\{\}|s) = 1$ when the query action is $a_\mathrm{q} = 0$. Finally, we construct a Belief Taskable MDP (BTMDP) as follows.

Definition 4 (*BTMDP*)**.** *Given a finite set of LTL formulas Φ and its probability distribution τ over Φ, a finite set of propositions $\mathcal{P}$, a labeling function L, an event detector O and a MDP without a reward function as a tuple $\mathcal{M} = (S, T, A, p, \gamma, \mu)$, the BTMDP is defined as $\mathcal{M}_b = (S_b, T, A_b, p_b, R_b, \gamma, \mu_b)$, where $S_b = S \times \Psi$ is a finite set of product states, $A_b = A_a \times A_q$ is a finite set of product action, $p_b((s', b'(\psi')) \mid (s, b(\psi)), a) = p(s' \mid s, a) \cdot b'(\psi')$, $\mu_b(s, b(\varphi)) = \mu(s) \cdot \tau(\varphi)$ is a initial distribution of product states, where $\tau(\varphi)$ is the probability of selecting φ from Φ and initially $b(\varphi) = 1$, and a reward function*

$$R_b((s,\Psi),a) = \begin{cases} 1 & \text{if } \Psi \cap \{\psi_{truth}\} = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

Here ψ_{truth} is the LTL formula progressed by events captured by the labeling function, which the agent cannot directly observe, and used only for reward calculation during training. The possible LTLs Ψ manage all progressions of branched LTL formulas according to the uncertainty, which means that the progress of the LTL formula is partially observable for the agent. Therefore, the policies for $\mathcal{M}_b$ can act following the belief of the possible LTLs but cannot know whether its actions follow the actual events in the environment during training. The reward function R_b addresses this situation by encouraging both $\psi \in \Psi$ and ψ_{truth} to become true

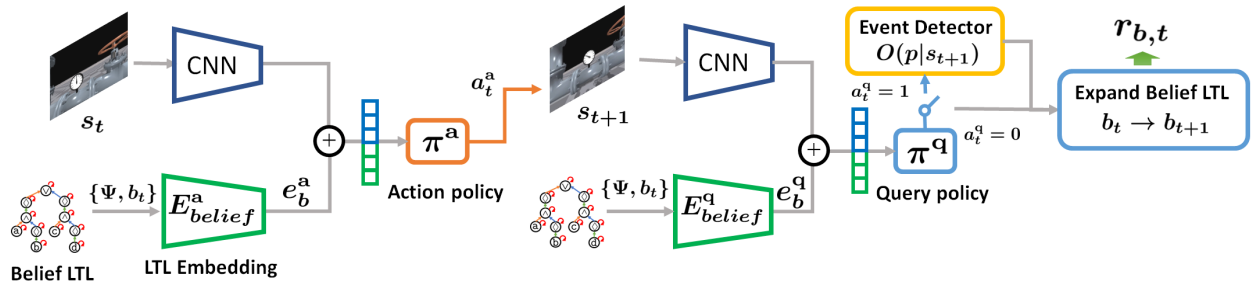


Fig. 2: Overview of the LAQBL framework. The state s and the belief LTL embeddings e_b are input to the policies, and each action is sampled sequentially. All the embeddings and policies are optimized through RL by the reward r_b.

simultaneously. On the contrary, if both $\psi \in \Psi$ and ψ_{truth} are unsatisfied or progress to false, the agent is not rewarded. **Example of reward calculation.** Consider the example in Section IV-A: $\Psi = \{\varphi_{\mathtt{a}} = \Diamond \mathtt{b} \vee \Diamond(\mathtt{c} \wedge \Diamond \mathtt{d}), \varphi_{\mathtt{c}} = \Diamond(\mathtt{a} \wedge \Diamond \mathtt{b}) \vee \Diamond \mathtt{d}\}$. At the next time step $t+1$ if $\varphi_{truth} = \varphi_{\mathtt{a}}$:

- when $L(s_{t+1}) = \mathtt{b}$ and the agent receives $O(\mathtt{b}|s_{t+1}) = 1.0$, $\mathrm{prog}(\sigma^{\mathtt{b}}, \varphi_{\mathtt{a}}) = \mathrm{prog}(\sigma^{\mathtt{b}}, \varphi_{truth}) = \mathtt{true}$ and the agent obtains a reward of 1.
- when $L(s_{t+1}) = \mathtt{d}$ and the agent receives $O(\mathtt{d}|s_{t+1})) = 1.0$, only $\mathrm{prog}(\sigma^{\mathtt{d}}, \varphi_{\mathtt{c}}) = \mathtt{true}$ and the agent obtains a reward of 0.

Note that to make a reward calculation of an episode feasible, we assume that the event detection of propositions regarding the terminal set is done with no uncertainty in this paper.

C. Problem Statement

This paper considers the problem of an agent interacting with an environment and an event detector modeled by a BTMDP. Our goal is to obtain policies that maximize the reward obtained by correctly progressing the instructed LTL under the belief state based on the uncertain detection of propositions obtained from the event detector. To this end, the problem can be described as follows.

Problem 1. *Given a BTMDP* $\mathcal{M}_b = (S_b, T_b, A_b, p_b, R_b, \gamma, \mu_b)$*, a finite set of propositions* $\mathcal{P}$*, a labeling function* L*, a finite set of LTL formulas* Φ*, a probability distribution* τ *over* Φ *and an event detector* O*, find policies* π_b^a *and* π_b^q *that optimize all tasks in the set* Φ *by maximizing the reward* R_b *for all state* $s_b \in S_b$ *and LTL instructions* $\varphi \in \Phi$*.*

V. Learning Action and Query Policies Over Belief LTL

A. Algorithm Overview

We propose a framework, namely Learning Action and Query over Belief LTL (LAQBL), that simultaneously learns the belief LTL embedding E_{belief}, which learns the representation of belief LTL, and two policies, the action policy π^{a} and the query policy π^{q}, all through RL. We use different functions E^{a}_{belief} and E^{q}_{belief} of the same architecture for each policy π^{a} and π^{q} with different objectives. An overview of the LAQBL is illustrated in Fig 2.

Given an LTL formula φ sampled from the set Φ according to a distribution τ and an initial state s_0, actions from the two policies are sequentially sampled. Firstly, the state s_0 and the feature of the belief LTL $e^{\mathrm{a}}_b = E^{\mathrm{a}}_{belief}(\varphi, b(\varphi)), b(\varphi) = 1.0$ are input to the policy $\pi^{\mathrm{a}}(a^{\mathrm{a}}|s_0, e^{\mathrm{a}}_{b,0})$ and an action a^{a}_0 is sampled. Next, the state s_1 transitioned by the state transition probability $p_b = p(s_1|s_0, a^{\mathrm{a}}_0)$ is input to the policy $\pi^{\mathrm{q}}(a^{\mathrm{q}}|s_1, e^{\mathrm{q}}_{b,0})$. The possible LTLs Ψ are computed from LTL formulas progressed by propositions with $O(p|s_1)$ greater than 0, and the belief LTL is updated by Eq. (2). Finally, the state s_1 and the belief LTL again input to the policy π^{a}. Both policies are optimized through RL using the reward r_b until the end of the episode.

B. Graph Embedding of Belief LTL

For the agent to learn behavior based on various given LTL instructions while considering the uncertainty of event detectors, it is crucial for the belief LTL embedding E_{belief} to learn discriminative features based on the differences in LTL instructions and their beliefs. A GNN-based method that embeds including future events [7] leads to non-myopic behavior and generalizability to unseen LTL formulas compared to a naive method that encodes only the next event symbols in given LTL formula [5] or RM-based method [3]. However, these methods do not provide a way of capturing uncertainty in event detection. To tackle this issue, we propose an augmentation of the GNN-based embedding that preserves LTL semantics and embeds both branched LTL formulas and their beliefs into the feature space.

We first represent the LTL formulas in the possible LTLs $\psi \in \Psi$ as a tree-structured directed graph with the tokens (propositions and operators) assigned to nodes in the same way as in [7]. As an example, the graph representation of the possible LTLs $\Psi = \{\varphi_{\mathtt{a}} = \Diamond \mathtt{b} \vee \Diamond(\mathtt{c} \wedge \Diamond \mathtt{d}), \varphi_{\mathtt{c}} = \Diamond(\mathtt{a} \wedge \Diamond \mathtt{b}) \vee \Diamond \mathtt{d}\}$ is shown in Fig. 3 (a). We add a new node connected to the generated graphs and assign the beliefs $b(\varphi_{\mathtt{a}})$ and $b(\varphi_{\mathtt{c}})$ corresponding to each LTL formula as the weight of the connected edges $W_{\varphi_{\mathtt{a}}}$ and $W_{\varphi_{\mathtt{c}}}$. We define the constructed LTL graph as a *belief LTL*, representing the set of the possible LTLs and their belief. The embedding of the belief LTL e_b by the following formula, with a slight abuse of notation:

$$\begin{aligned} e_b &= E_{belief}(\{\psi, b(\psi) | \psi \in \Psi, b(\psi) \in B\}) \\ &= f\left(\textstyle\sum_{\psi \in \Psi} x_\psi W_\psi\right), \end{aligned} \tag{4}$$

where f is a readout function, we use a 1-layer fully-connected network, and x_ψ is the node feature of the LTL graph $\psi \in \Psi$

by relational graph convolutional network (R-GCN) [23]. Since message passing by R-GCN is performed bottom-up, the feature x_ψ of the top node of each LTL graph is treated as the feature of the graph. The belief LTL can build unique graph representations according to the uncertainty of the event detector even if given the same instructions and can scale according to the belief expansion by Eq. (2).

VI. Experiments

A. Evaluation of the Belief LTL Embeddings

We first show whether the belief LTL embedding learns discriminative representation in the latent feature space according to given LTL formulas and their belief through a navigation task with a small LTL task set. A map in Fig. 3 (b) consists of 7×7 squares with 10 unique events and the "ab" placed twice. As the state, each grid has 13 dimensions (12 events and the agent position), with 1 assigned to the channel corresponding to the elements present in each grid and 0 otherwise. At the grid "ab", the agent satisfies only one of the events in each episode and receives its uncertainty from the event detector. The actions are moving up, down, left, and right. The LTL task is uniformly sampled one of six tasks including until operator $\Phi = \{\varphi_1, \cdots, \varphi_6\}$ in Fig. 3 (b). The event detector can correctly detect the satisfied event with a random probability of 0.6 to 1.0 with a resolution of 0.01, and the agent observes this probability as its uncertainty in the grid "ab".

We visualize the concatenated state vector and embedding of belief LTL through the rollout of the learned action policy by a principal component analysis (PCA). In the qualitative evaluation, LTL instructions and their uncertainty level are organized in the feature space and represented discriminatively compared to before learning. We also quantify the relation between features and belief using canonical correlation analysis (CCA) and confirm that our embeddings effectively capture the belief for each LTL instruction, as shown by the increase in canonical correlation after learning.

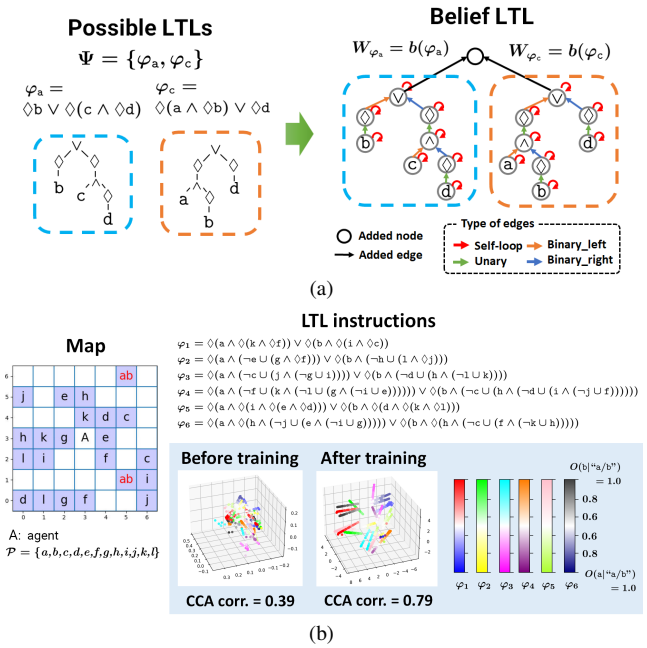


Fig. 3: (a) A graph representation of the belief LTL. A one-hot vector unique to the LTL token is assigned to each node as a feature, and each edge type has a different weight. (b) Visualization of the belief LTL embeddings. The embeddings are represented in three-dimensional space by the PCA, with each color type and intensity indicating the given LTL instructions and their level of uncertainty. The correlation coefficient of the CCA is calculated by the features reduced by the PCA and the uncertainty observed in the "ab" grid.

B. Navigation under Different Uncertainty in Event Detector

We then evaluate our method in the navigation that requires an agent to change actions depending on the event detectors' uncertainty level. We use the same map as in the experiment in VI-A. One episode consists of 200 timesteps, and each π^a and π^q policy action consumes one. Locations of events on the map are randomized for each seed, and the agent is always placed in the center.

1) LTL Task Design: Unlike the common target focusing on whether the agent can accomplish LTL instructions, evaluating what traces the agent follows to accomplish LTL instructions is necessary for our objective. Therefore, we design parallelized LTL task space to emphasize situations that motivate the agent to vary the action depending on the event detector's uncertainty. Fig. 4 (a) shows an example of the LTL task used in our experiments. The task is divided into a left and a right branch by the first disjunction operator (∨), and the left branch has a disjunction corresponding to the "ab" grid, making up the LTL task that can be satisfied by three different word traces. We denoted the disjunction containing "ab" as the *uncertain disjunction* for convenience.

While this LTL task can be progressed to true no matter which traces the agent follows, we make it feasible for the agent to learn to choose these traces depending on the uncertainty of the event detector. We use two types of event detector that probabilistically assigns uncertainty to the event: *Expert* that detects the event with high accuracy p_{high} and *Beginner* that detects with low accuracy p_{low}. This paper sets these to $p_{high} = 0.95, p_{low} = 0.5$, respectively, and fixed during training and testing. That is, if the event a occurred on the grid "ab", *Expert* assigns $O(\mathtt{a}|s) = 0.95$ and $O(\mathtt{b}|s) = 0.05$ with 95% and 5% probabilities, respectively. On the other hand, *Beginner* always assigns $O(\mathtt{a}|s) = 0.5$ and $O(\mathtt{b}|s) = 0.5$. Since we do not inform the agent which detector is set and the event detectors are uniformly sampled at the start of the episode, the agent needs to select the following reactive behavior corresponding to the event detector's uncertainty.

i) If the agent observes low uncertainty at "ab", it acts to achieve the event following the confident event (the trace indicated by blue on the LTL tree in Fig. 4 (b)).
ii) If the agent observes high uncertainty at "ab", it avoids the event following the uncertain disjunction with a high risk of failure (the trace indicated by green).

The most desirable way for the agent to address the randomness of the choice of event detectors during training is to change these behaviors according to the uncertainty obtained in the "ab" grid, which corresponds to the trace indicated by yellow arrows on the LTL tree.

Note that we provide an additional reward $\tilde{r}$ if the agent

progresses events through the trace containing the uncertain disjunction and earns a positive reward according to Eq. (3). This is necessary to balance the expected returns the agent can earn by behavior i) and ii), and we set the $\tilde{r} = 0.4$ in this experiment. The table in Fig. 4 (c) summarizes the expected returns obtained in the traces progressed by the agent.

2) Training Details: We use PPO [24] to train both policies π^{a} and π^{q}, and the hyperparameters are the same as [7]. The state is encoded by a 3-layer CNN and ReLU activations. The LTL task $\varphi \in \Phi$ is uniformly sampled in each episode, and the set Φ has the uncertain disjunction in either branch, and propositions are randomized except for `a` and `b`. The nesting depth of the LTL formula is also set randomly from 2 to 4 (Fig. 4 (a) shows a tree with depth 3), and the number of possible unique tasks is over 197 million. We focus on analyzing the agent behavior due to uncertainty in the "ab" grid; the event detector returns the same results as the labeling function L except for the "ab" grid.

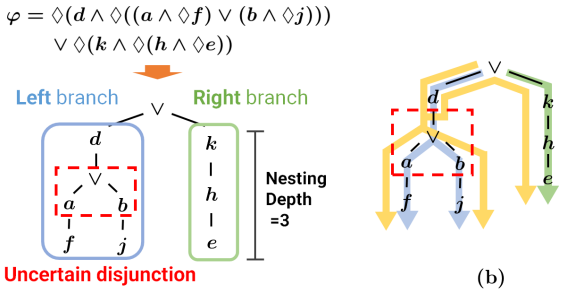
3) Comparisons: We evaluated our model against the following methods:

- **LTL2Action:** The method employs the LTL embedding the same as the GNN-based method proposed in [7] and does not use the belief over multiple LTL formulas. The LTL formulas are progressed based on the Most Likely criteria [8][1], which use a proposition with high probability to address uncertainty in the event detector.
- **belief+regular query:** The method uses the embedding of belief LTL the same as ours and performs a query to the event detector every step. This corresponds to the ablation of query policy in our method.

4) Results: **Evaluation of reactive behavior against different uncertainty in the event detector.** We first evaluate how different uncertainty in the event detector affects the behavior and performance of the agent. Fig. 5 (a) and (b) show the average return of each method during training for different probabilities of selecting *Expert* and *Beginner* event detectors, respectively. The probability of selecting *Expert* sets to 95% in Fig. 5 (a) and 50% in Fig. 5 (b). There is no difference in performance in the setting where *Expert* is selected dominantly, whereas LTL2Action, which does not consider the belief, performs significantly worse in the setting where *Expert* and *Beginner* are uniformly selected. Additionally, to quantify the reactivity of the agents to the uncertainty during training in the setting where the event detector is uniformly sampled, we count whether the agent performed the behavior i) and ii) in Section VI-B1 in Fig. 5 (c). The result that the reactivity of the belief-based method is closest to 0.5 and obtains the highest return in Fig. 5 (b) indicates that the belief-based agents achieve select behavior i) or ii) according to the uncertainty in the event detector. Alternatively, LTL2Action has a large variance of reactivity, indicating that it can not act according to the event detector.

Finally, we visualize the differences between the policies learned by LTL2Action and our method for qualitative evaluation in Fig. 6. The rollout of learned policies with the LTL instruction in Fig. 6 (a) at the map in Fig. 3 (a) shows that LTL2Action ignores the "ab" grid containing uncertainty, as shown in Fig. 6 (b). This result indicates that LTL2Action cannot capture the uncertainty and therefore learns only to achieve events not included in the uncertain disjunction. On the other hand, our policy goes to the "ab" grid first and achieves the events while following the instructions according to the uncertainty, as shown in Fig. 6 (c).

[1]The three reward functions except Most likely proposed in [8] are not in comparison because they aim to satisfy all LTLs that support the belief and are not consistent with the purpose of the experiment.



(c)

Agent behavior	Expected Reward
Always selects behavior i)	$\tilde{R}_b = (p_{high}(R_b + \tilde{r}) + (1 - p_{high})(0.0) + p_{low}(R_b + \tilde{r}) + (1 - p_{low})(0.0))/2 = 0.965$
Always selects behavior ii)	$\tilde{R}_b = R_b = 1.0$
Selects behavior i) or ii) according to the event detctor	$\tilde{R}_b = (p_{high}(R_b + \tilde{r}) + (1 - p_{high})(0.0) + R_b)/2 = 1.165$

Fig. 4: (a) An example of an LTL instruction given to an agent in the experiments (omits some operators for illustration). (b)(c) The reactive behavior of the agent on the LTL task tree and the corresponding expected returns that the agent obtains in the experiment. For the agent to obtain the highest return, it is necessary to choose a proposition to progress according to the event detector's uncertainty obtained in the "ab" grid.

Method	Depth2-4 RT	RCT(%)	NEs	QFR(%)
Ours(query policy)	**1.13**±0.30	**95.1**	9.27±17.52	2.6
LTL2Action	0.99±0.51	49.1**	6.98±9.36	2.7
belief+regular query	0.99±0.46	64.5**	0.67±0.93	-
query action	1.01±0.49	49.6**	15.33±27.55	5.6
Method	**Depth5 RT**	**RCT(%)**	**NEs**	**QFR(%)**
Ours(query policy)	**1.07**±0.37	**86.9**	18.04±24.09	3.1
LTL2Action	0.96±0.50	48.3**	13.86±18.53	2.5
belief+regular query	0.91±0.53†	54.9**	1.2±1.59	-
query action	0.68±0.66*	33.1**	57.51±653.08	6.3

†: $p < .05$, *: $p < .005$, **: $p < .0005$ by t-test.

TABLE II: Results of testing policies in the navigation task under the probabilistic error of the event detector. We report the mean and standard deviation of the returns (RT), the mean accuracy of reactivity (RCT), the mean and standard deviation of the number of times to reach the grid with no events (NEs), and the query failure rate which averaged over queries on the grid with no events (QFR). All results are averaged over 15 seeds of 1000 episodes of the rollout of the learned policies, with 500 episodes of each fixed event detector per seed. The rollouts are performed on the training distribution of tasks (Depth2-4) and out-of-distribution tasks with increased depth of sequences (Depth5).

Evaluation of the different query architecture under the probabilistic error of the event detector. To validate the effect of the query policy of the LAQBL framework, we next experiment in the environment with probabilistic false positives by the event detector. To directly implement the harmful effects of unnecessary queries, we set the task to fail immediately with a 10% probability of failure if the policy π^{q} queries the event detector on grids with no events in this

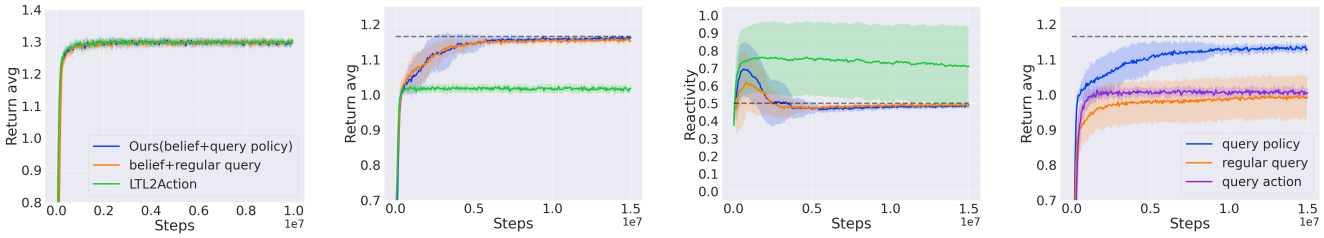


Fig. 5: (a)(b) The average returns with baselines during training in the navigation task with 15 random seeds. The probability that *Expert* event detector is sampled is 95% in (a) and 50% in (b). (c) The reactivity of the agent to uncertainty in the event detector in the training of (b). Closer to 1.0 indicates that the agent achieves the task through the uncertain disjunction, and closer to 0.0 indicates that the agent achieves by avoiding that. (d) The average returns with different policy architectures.

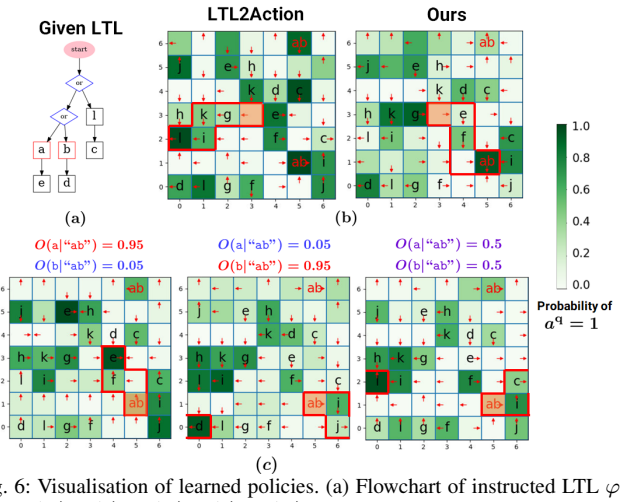


Fig. 6: Visualisation of learned policies. (a) Flowchart of instructed LTL $\varphi = \Diamond(\mathtt{a} \wedge \Diamond\mathtt{e}) \vee \Diamond(\mathtt{b} \wedge \Diamond\mathtt{d}) \vee \Diamond(\mathtt{l} \wedge \Diamond\mathtt{c})$. (b) Comparison of actions of policies learned by LTL2Action and ours at the start of the episode. (c) Actions of our policies according to different uncertainties of the event detector after a query on the "ab" grid at $(x, y) = (5, 1)$. In each grid, red arrows indicate the dominant action of the policy π^a, and green intensity indicates the probability of action $a^q = 1$ sampled by the policy π^q.

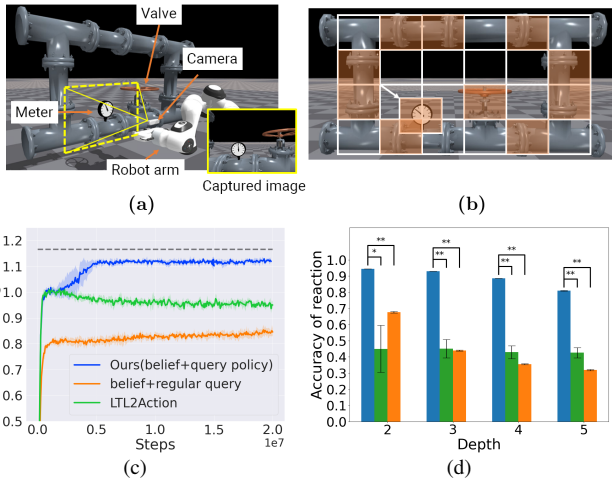


Fig. 7: (a) Overview of the piping inspection task. (b) Observation space of the robot arm. Events are registered on colored grids, and the robot moves between the center points of each grid (The arrow indicates the shift for framing the meter). (c) Comparisons of average returns during training in the piping inspection task with 5 random seeds. (d) Accuracy of the reaction for testing trained policies ($* : p < .005$, $** : p < .0005$ by t-test). The results are averaged over 5 seeds, and the test set is the same as in Table II.

experiment. We compared the LAQBL framework with two methods, the regular query, which is an ablation study of our method, and the *query action*, which adds a query action to the action policy π^a. The result in Fig. 5 (d) highlights the effectiveness of the LAQBL framework, and both the regular query and the query action performance are degraded. Contrary to expectations, the regular query learns to move as far as possible through the grid where any events exist, but performance degradation is inevitable. The query action lacks the independent query policy, making it hard to learn the embedding that optimizes both action and query policy and does not offer generalization performance.

Finally, we report the test result in Table II. Our method achieved the best performance and reactivity to uncertainty in the event detector for both LTL tasks in the training distribution and out-of-distribution with increased depth. Since we are testing with a different random seed than in training, it is possible that LTL not sampled in training will be sampled in testing. The average returns of the test are comparable to the training result, which indicates that learned policies are optimized for the overall task set Φ as we expected. Note that the query failure rate for the out-of-distribution task is slightly higher in our method, but this is because more moves and queries are required to accomplish longer tasks. See the attached video of how our agent behaves.

C. Piping Inspection by Robot Arm

To evaluate our algorithm in a realistic environment, we build the piping inspection task with a camera mounted on a robotic arm using Isaac Gym [25] shown in Fig. 7 (a). We discretized the observation space by fixing the depth motion of the arm and dividing it into a 6×4 grid on a two-dimensional plane shown in Fig. 7 (b), and 10 propositions are fixed on the grid corresponding to pipe joints, a meter, and a valve to simulate the inspection. The observation is 80×60 RGB image, and action spaces of the policies and the LTL task design are the same as the navigation task. The probability of selecting the event detector is set to 50%, while the proposition assigned to whether the meter value is normal or abnormal is set to the uncertain disjunction.

Performance comparison with high-dimensional observation. Fig. 7 (c) shows the performance comparisons with each method during training under the probabilistic error of the event detector. Fig. 7 (d) presents the average reaction accuracy of behavior i) and ii) in an environment with fixed event detectors by the rollout of the learned policies. These results show that our method has the best reactivity to uncertainty in

the event detector without performance degradation, even for high-dimensional image input, suggesting the applicability of our method to real-world tasks.

As a supplement, we present an interactive inspection with a human event detector by using learned policies to show the applicability of our system for real-world applications. We developed an operational GUI that allows a human operator to inspect the object by moving the camera with a mouse when triggered by the query policy. See the attached video for the interactive demonstration with a human detector.

VII. DISCUSSION

We define the event detector as a function that offers gradual evaluations of uncertainty, designed to work with human operators and edge AI systems, which use a multi-class classifier like softmax. Although detailed uncertainty feedback is cumbersome for humans, a workflow with several options and the corresponding uncertainty for the robot is feasible, like the demonstration video.

The limitation of this method is that the traces of tasks achieved may be biased without adjustments to the reward design since our method learns policies by sparse rewards. A future direction to mitigate this limitation is to apply the reward shaping [26] or the multi-goal RL techniques that regularise over the goals satisfied by the agents [27].

Our method leaves some topics that require further discussion as future work. First, this paper empirically shows that our method works in a setting the Possible LTL always includes the LTL progressed by true events. While this makes it possible that the policy satisfies tasks by choosing the same action as the optimal policy π^*_Φ that follows the Taskable MDP, more formal discussions on task satisfaction will be addressed in future work. Also, our experiments do not contain the operator that can be progressed to false, such as the until operator $\cup$, and further evaluation of the reward function is needed when the Possible LTL contains false.

VIII. CONCLUSIONS

This paper proposes a LAQBL framework for learning the agent to follow various LTL instructions with the uncertain event detectors. The navigation task in the grid world and the inspection task with high-dimensional image input show that our method can perform in response to the uncertainty of the event detector and that effective querying of the event detector can reduce task failures due to unnecessary queries.

REFERENCES

[1] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (FOCS)*, 1977, pp. 46–57.

[2] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an rl agent using ltl," in *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018, pp. 452–461.

[3] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "LTL and Beyond: Formal languages for reward function specification in reinforcement learning." in *International Joint Conference on Artificial Intelligence*, vol. 19, 2019, pp. 6065–6073.

[4] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Model-free reinforcement learning for stochastic games with linear temporal logic objectives," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10 649–10 655.

[5] B. G. León, M. Shanahan, and F. Belardinelli, "Systematic generalisation through task temporal logic and deep reinforcement learning," *arXiv preprint arXiv:2006.08767*, 2020.

[6] Y.-L. Kuo, B. Katz, and A. Barbu, "Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5604–5610.

[7] P. Vaezipoor, A. C. Li, R. T. Icarte, and S. A. McIlraith, "LTL2Action: Generalizing LTL instructions for Multi-Task RL," in *International Conference on Machine Learning (ICML)*, 2021, pp. 10 497–10 508.

[8] A. Shah, S. Li, and J. Shah, "Planning with uncertain specifications (PUnS)," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3414–3421, 2020.

[9] M. Ghasemi, E. Bulgur, and U. Topcu, "Task-oriented active perception and planning in environments with partially known semantics," in *International Conference on Machine Learning (ICML)*, 2020, pp. 3484–3493.

[10] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018, pp. 2107–2116.

[11] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 5338–5343.

[12] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control-based motion planning with partially infeasible LTL constraints," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1279–1284, 2020.

[13] Y. Kantaros, S. Kalluraya, Q. Jin, and G. J. Pappas, "Perception-based temporal logic planning in uncertain semantic maps," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2536–2556, 2022.

[14] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7973–7980, 2021.

[15] R. Sharan and J. Burdick, "Finite state control of pomdps with ltl specifications," in *American Control Conference*, 2014, pp. 501–508.

[16] M. Bouton, J. Tumova, and M. J. Kochenderfer, "Point-based methods for model checking in partially observable markov decision processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 061–10 068.

[17] K. Hashimoto, N. Tsumagari, and T. Ushio, "Collaborative rover-copter path planning and exploration with temporal logic specifications based on bayesian update under uncertain environments," *ACM Trans. Cyber-Phys. Syst.*, vol. 6, no. 2, apr 2022.

[18] Q. Gao, M. Pajic, and M. M. Zavlanos, "Deep imitative reinforcement learning for temporal logic robot motion planning with noisy semantic observations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8490–8496.

[19] A. C. Li, Z. Chen, P. Vaezipoor, T. Q. Klassen, R. T. Icarte, and S. A. McIlraith, "Noisy symbolic abstractions for deep RL: A case study with reward machines," *arXiv preprint arXiv:2211.10902*, 2022.

[20] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal methods in system design*, vol. 19, no. 3, pp. 291–314, 2001.

[21] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[22] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artificial intelligence*, vol. 116, no. 1-2, pp. 123–191, 2000.

[23] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[26] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] R. Zhao, X. Sun, and V. Tresp, "Maximum entropy-regularized multi-goal reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019, pp. 7553–7562.