ILCL: Inverse Logic-Constraint Learning from Temporally Constrained Demonstrations

Author Names Omitted for Anonymous Review. Paper-ID [768]

***Abstract*—We aim to solve the problem of temporal-constraint learning from demonstrations to reproduce demonstration-like logic-constrained behaviors. Learning logic constraints is challenging due to the combinatorially large space of possible specifications and the ill-posed nature of non-Markovian constraints. To figure it out, we introduce a novel temporal-constraint learning method, we call inverse logic-constraint learning (ILCL). Our method involves a two-player zero-sum game between 1) a genetic algorithm-based temporal-logic mining (GA-TL-Mining) and 2) logic-constrained reinforcement learning (Logic-CRL). GA-TL-Mining efficiently constructs a syntax tree for parameterized truncated linear temporal logic (TLTL) without relying on predefined templates. Subsequently, Logic-CRL finds an optimal policy that maximally adheres the identified TLTL constraints, employing Lagrangian relaxation for optimization. Our evaluations show ILCL outperforms state-of-the-art baselines in learning and transferring TL constraints across four temporally constrained scenarios. Furthermore, we demonstrate the transferability of our method in real-world peg-in-shallow-hole environments.**

I. Introduction

Policy learning for robots often involves constraints that account for user, task, and environmental restrictions, as shown in Fig. 1. To autonomously adopt these constraints, researchers have introduced inverse constraint learning (ICL) methods that recover constraints from demonstrations [46, 40, 36, 20, 27, 32, 49]. While these methods often focus on global state-action constraints, a number of robotic tasks require spatial or temporal restrictions. For example, mobile robots must stop at the intersection until the traffic sign turns green. To provide more structured and interpretable restrictions, researchers often adopt temporal logic (TL), such as linear temporal logic (LTL) [48] and signal temporal logic (STL) [39], in robotics.

In this context, we aim to solve the problem of TL-constraint learning from demonstrations to reproduce demonstration-like constraint-satisfying behaviors. However, TL-constraint learning is challenging due to the combinatorially large space of logic specifications in addition to the non-differentiable representation following strict syntactic rules (e.g., Backus-Naur form). For efficient searches, researchers often adopt predefined templates [61] or restrict the space of searches using specific operators only [37]. However, the limited expressivity restricts applications in tasks. Therefore, a desired approach requires to derive a free-form of logic constraints without predefined structure.

Another challenge is the ill-posed nature of non-Markovian constraints. Given the ill-posedness, conventional approaches focus on finding constraints without considering the reward maximization in policy learning. This lowers the chance of finding optimal and transferable constraints from experts. Further, temporal constraints require retaining a history of actions (i.e., long-term dependency) and assessing constraint violations after all actions have been observed. These challenges complicate the integration of TL with conventional ICL frameworks [36] via inverse and forward constraint reinforcement learning (CRL) processes.

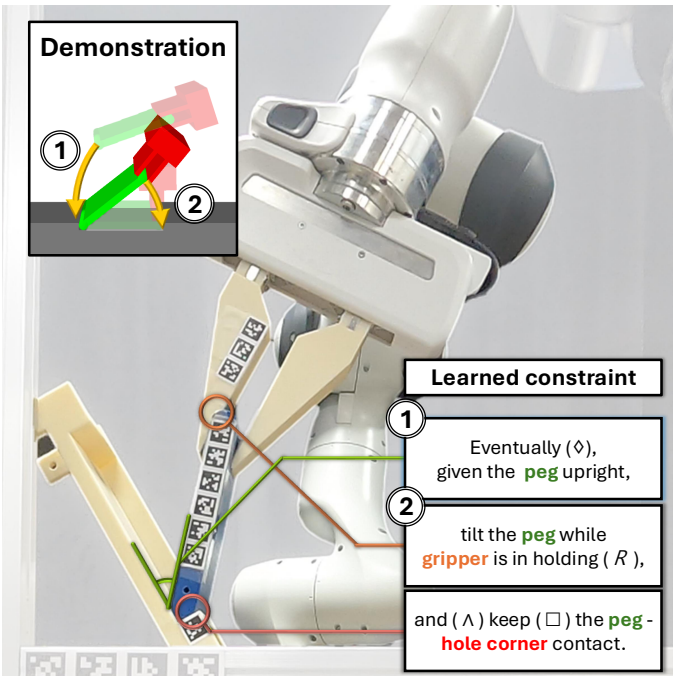

Fig. 1. An exemplar *peg-in-shallow-hole* task, which requires maintaining a peg-hole contact while inserting the peg using a parallel jaw gripper. By learning temporal logic constraints from demonstrations, our method ILCL successfully transfers demonstration-like constrained insertion behavior on a unseen tilted hole environment.

We propose a free-form transferable TL-constraint learning algorithm, which we call inverse logic-constraint learning (ILCL). Our method involves a two-player zero-sum game between 1) a genetic algorithm-based temporal-logic mining (GA-TL-Mining) and 2) logic-constrained reinforcement learning (Logic-CRL). GA-TL-Mining efficiently constructs a syntax tree for parameterized truncated linear temporal logic (TLTL) [35] without relying on predefined templates. Subsequently, Logic-CRL finds an optimal policy that maximally adheres the identified TLTL constraints, measuring the degree of logic satisfaction across a trajectory defining a robustness function. Further, to handle the late assessment of TL constraints in CRL, we introduce a constraint redistribution method, analogous to the reward redistribution [3, 18], converting trajectory-based constraints into state-action constraints. To the best of our knowledge, we provide the first TL-based CRL leveraging Lagrangian relaxation.

We perform a statistical evaluation of ILCL against state-of-the-art ICL baselines with four temporally constrained scenarios in three simulated environments—*navigation*, *wiping*, and *peg-in-shallow-hole*—that require satisfying temporal constraints for navigation and manipulation tasks. ILCL outperforms baselines in learning a diverse form of TL constraints, achieving higher task success rates and low constraint-violation scores. We also demonstrate the applicability of the learned constraints by successfully transferring a constraint to a real-world *peg-in-shallow-hole* task, thereby demonstrating the robustness and scalability of TL constraints.

Our overall contributions are as follows:

- We propose a novel GA-based logic-tree mining method finding TLTL constraints from demonstrations without predefined templates.
- We introduce the first CRL method with non-Markovian TLTL constraints applying constraint redistribution for stable learning.
- We evaluate the constraint learning capability and transferability across four challenging robotic tasks through simulations. We also demonstrate real-world applicability through the *peg-in-shallow-hole* environment.

II. Related Work

We review individual research streams in related fields.

Numerical constraint learning. Our proposed method is a variant of ICL that recovers constraints from demonstrations. Early ICL methods seek to learn numerical constraints on discretized state space [50, 13]. With the advancement of neural representations, recent ICL methods model constraints on continuous state or state-action spaces [46, 40, 45, 27, 36, 60]. To enhance model scalability, researchers introduce multi-task [32] and multi-modal [49] approaches. However, numerical representations are often hard to interpretable and transferable to new settings. Instead, we use logical representations for robust generalization.

Logic constraint learning and mining. A logic constraint is a form of formal language, such as first-order rules or TL. Early methods for inducing logic constraints from demonstrations often adopt inductive logic programming (ILP) [43], which outputs a logic program based on positive and negative examples. However, negative (i.e., anomalous) examples are rarely obtainable in general. To address this, researchers combine ILP with constraint inference to generate negative examples [6] or learn logical formulas using one-class decision tree [5]. However, these approaches often limit the outcomess to simple global propositions (e.g., $^{\forall}(x < 3)$) or timestep-specific constraints, thereby restricting generalizability. To enhance temporal representation, researchers introduce STL learning, but the algorithms still face the same limitation of relying on predefined logic forms or exact templates [61, 37]. In contrast, our method employs specification mining without relying on pre-defined structures.

Logic specification mining refers to the process of inferring potential system properties from observations (e.g., STL mining [8]). To address the combinatorially large space of specifications, researchers often fix logic structures [56, 28], simplify forms [33, 34], or discretize the search space [55, 2]. Otherwise enumerative search methods explore the full specification space, restricting the search depth to manage complexity [42]. Alternatively, by using syntax trees to represent specifications, tree-based genetic algorithms explore the space of logic trees through the exchange of subtrees [44, 47]. We extend the algorithm to compose an TLTL constraint with additional temporal operators (e.g., $\mathcal{R}, \bigcirc$).

Constrained Reinforcement Learning. CRL, also known as safety RL, aims to find a reward-maximizing policy while satisfying safety constraints. Garcia and Fernandez classify approaches into two categories: 1) modification of the exploration process and 2) the modification of the optimization criterion [19]. The first category typically restrict actions or policies within a trust region [9, 1, 14, 12]. While these connects to formal verification, the temporal constraints we use make it hard to define the trust space for action sequences.

On the other hand, the second category often employs Lagrangian relaxation techniques to convert the constrained optimization in CRL into a typical RL problem with unconstrained objectives (i.e., rewards) [53, 24, 51]. However, the long-term dependency in temporal logics may result in reward delays. To address this, researchers introduce neural networks to estimate trajectory returns [3, 38] or redistribute rewards across time steps, smoothing the reward signals [18]. We extend this reward redistribution to TL-based CRL evaluating constraint violations at the end of episodes.

In addition, researchers introduce RL-based Markovian policy learning that satisfies TL constraints, but the algorithm do not guarantee obtaining reward-optimal Markovian constrained policies [23, 58, 54, 30].

III. Preliminaries

A. Constraint Markov Decision Process (CMDP)

A CMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, C)$, where the elements are a space of states, a space of actions, a transition probability distribution $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$, a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a constraint-cost function $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, respectively. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ is a probability density function to determine the likelihood of selecting an action given a state. Then, the objective of CMDP is to find a policy π^* that generates a state-action sequence $\xi = (\mathbf{s}_0, \mathbf{a}_0, ...\mathbf{s}_T, \mathbf{a}_T)$ with a finite horizon T maximizing the expected return while the expectation of cumulative constraint cost bounded to a budget d: $\pi^* = \arg\max_\pi \mathbb{E}_{\xi\sim\pi}[R(\mathbf{s}_t, \mathbf{a}_t)]$ subject to $\mathbb{E}_{\xi\sim\pi}[\sum_{t=0}^{T} C(\mathbf{s}_t, \mathbf{a}_t)] \leq d$. In this work, we use $R(\xi) = \sum_{t=0}^{T} R(\mathbf{s}_t, \mathbf{a}_t)$ and $C(\xi) = \sum_{t=0}^{T} C(\mathbf{s}_t, \mathbf{a}_t)$ to denote the cumulative discounted rewards and constraint costs, respectively.

To generate a sequence that satisfies TL constraints, we introduce another CMDP with a TLTL constraint C_ϕ that returns a positive value when an input state sequence $\xi^{(s)} = (\mathbf{s}_0, ...\mathbf{s}_T)$ violates a TLTL formula ϕ at any time step. For notational convenience, we use the terms state-action sequence, state

sequence, and trajectory interchangeably. We also omit the arguments of $C(\mathbf{s}, \mathbf{a})$ and $C(\xi)$.

B. Truncated linear temporal logic (TLTL)

We employ TLTL [16], a predicate temporal logic that extends LTL and LTL over finite traces (LTL_f [21]). Each TLTL formula consists of predicates formulated as $a \cdot \mathbf{s}^{(i)} < b$, where $a \in \{-1, 1\}$, $b \in \mathbb{R}$ is a constant amplitude, and the superscript i refers the i-th element of the state vector $\mathbf{s}$. The syntax for defining TLTL formulas is as follows:

$$\begin{aligned} \phi ::= \top | \bot | \mu | \neg\mu | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \bigcirc \phi | \\ \Diamond\phi | \Box\phi | \phi_1 \mathcal{U} \phi_2 | \phi_1 \mathcal{R} \phi_2, \end{aligned} \tag{1}$$

where $\top$ is the Boolean True, $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{R}$ (release), and $\bigcirc$ (next) are temporal operators. Note that the syntax in Eq. (1) is based on the positive normal form, where negations are applied solely to atomic predicates. We evaluate TLTL formulas against a finite state sequence $\mathbf{s}_{t:t+\Delta} = (\mathbf{s}_t, ..., \mathbf{s}_{t+\Delta})$ from a time step t with Δ representing a finite length.

To incorporate learnable thresholds into TLTL, we introduce a parametric variant of TLTL, we call pTLTL, similar to parametric STL (pSTL) [4]. The parameterization replaces the constant c in each atomic predicate (AP) with a parameter θ. Each AP is then a parametric AP (pAP), $a \cdot \mathbf{s}^{(i)} < \theta$. For example, we represent a TLTL formula ϕ as a pTLTL formula $\phi_{\boldsymbol{\theta}}$ substituting with a parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)})$,

$$\phi = (\mathbf{s}^{(0)} < 1)\, \mathrm{U}\, (\mathbf{s}^{(1)} > 2) \qquad \text{TLTL}, \tag{2}$$

$$\phi_{\boldsymbol{\theta}} = (\mathbf{s}^{(0)} < \boldsymbol{\theta}^{(0)})\, \mathrm{U}\, (\mathbf{s}^{(1)} > \boldsymbol{\theta}^{(1)}) \qquad \text{pTLTL}, \tag{3}$$

where $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)} \in \mathbb{R}$.

Next, to represent the degree of logic satisfaction, we describe the Boolean and quantitative semantics of TLTL using a robustness function ρ. Boolean semantics is the evaluation of an input state sequence $\mathbf{s}_{t:t+\Delta}$ at time t, where the sequence $\mathbf{s}_{t:t+\Delta}$ satisfies a formula ϕ at time t if and only if $\rho(\mathbf{s}_{t:t+\Delta}, \phi, t) > 0$. The robustness function ρ assigns a large positive value given strong satisfaction and a large negative value in cases of significant violation. Likewise, the quantitative semantics is the result of the robustness function ρ. For notational simplicity, we drop time index in the robustness function, representing it as $\rho(\xi, \phi)$ to evaluate robustness across the entire trajectory. For more details, see Appendix A.

IV. Problem formulation

The objective of ILCL is to identify a TL constraint C_ϕ, represented by a TLTL formula ϕ, from expert demonstrations Ξ_E. We formulate the identification task as a constrained optimization problem that finds an optimal constraint C^*_ϕ maximizing the expected cumulative rewards, obtained by the inferred constrained policy π_{C_ϕ}, while minimizing its expectation difference from the expert policy π_E demonstrated in Ξ_E:

$$\begin{aligned} C^*_\phi = \arg\min_{C_\phi} \max_{\pi_{C_\phi}} \quad & \mathbb{E}_{\xi\sim\pi_{C_\phi}}[R(\xi)] - \mathbb{E}_{\xi\sim\pi_E}[R(\xi)] \\ \text{s.t.} \quad & \mathbb{E}_{\xi\sim\pi_{C_\phi}}\left[C_\phi(\xi)\right] = 0 \\ & \mathbb{E}_{\xi\sim\pi_E}\left[C_\phi(\xi)\right] = 0, \end{aligned} \tag{4}$$

where $C_\phi(\xi) = \mathbb{1}[\rho(\xi, \phi) < 0]$. However, this minmax optimization is challenging due to the hard constraints that both expert and optimal policies must generate state sequences with zero violations.

To figure it out, we reframe Eq. (4) as a two-player zero-sum game, following [32]. Fig. 2 shows our formulation iteratively alternates constraint and policy optimizations. The constraint optimization player seeks to identify the k-th constraint $C_{\phi,k}$ based on the previously optimized policies $\pi_1, \ldots, \pi_k$:

$$\begin{aligned} C_{\phi,k} = \arg\max_{C_\phi} \quad & \sum_{i\le k} \mathbb{E}_{\xi\sim\pi_i}\left[C_\phi(\xi)\right] \\ \text{s.t.} \quad & \mathbb{E}_{\xi\sim\pi_E}\left[C_\phi(\xi)\right] = 0. \end{aligned} \tag{5}$$

Another policy optimization player then finds the next constrained policy π_{k+1} based on the previously optimized constraint $C_{\phi,k}$:

$$\pi_{k+1} = \arg\max_{\pi} \mathbb{E}_{\xi\sim\pi}[R(\xi)] \text{ s.t. } \mathbb{E}_{\xi\sim\pi}[C_{\phi,k}(\xi)] = 0. \tag{6}$$

Starting from an arbitrary initial constraint $C_{\phi,0}$, the alternation between Eq. (5) and Eq. (6) ensures the derivation of a constraint $C_{\phi,k}$. This constraint induces a constrained policy that minimally violates the optimal constraint while gaining higher rewards than the expert policy. In the following sections, we first describe the constraint optimization player's approach as a GA-TL-Mining problem in Sec. V, followed by the policy optimization player's method as a Logic-CRL problem in Sec. VI.

V. GA-based temporal logic mining

We design the GA-TL-Mining method to identify a TLTL constraint ϕ that maximizes a fitness function $\mathcal{F}_k(\phi)$ given demonstrations Ξ_E at the k-th iteration of ILCL. We define the fitness function $\mathcal{F}_k(\phi)$ as a Monte Carlo approximation of the dual problem objective from Eq. (5); $\mathcal{F}_k(\phi)$ is

$$\frac{\sum_{\xi\in\Xi_k} \mathbb{1}[\rho(\xi, \phi) < 0]}{|\Xi_k|} \cdot \Pi_{\xi_E\in\Xi_E} \mathbb{1}[\rho(\xi_E, \phi) > 0], \tag{7}$$

where Ξ_k is the set of trajectories sampled from the policies $\pi_1, \ldots, \pi_k$ from Logic-CRL. With the fitness function, our GA-TL-Mining method iteratively improves the population of solution logic trees (i.e., pTLTLs) through selection, crossover, and mutation operations. For simplicity, we refer pTLTLs and logic trees interchangeably.

Our proposed mining method consists of three steps: 1) initial population generation, 2) parent population selection, and 3) offspring generation. We describe detail below.

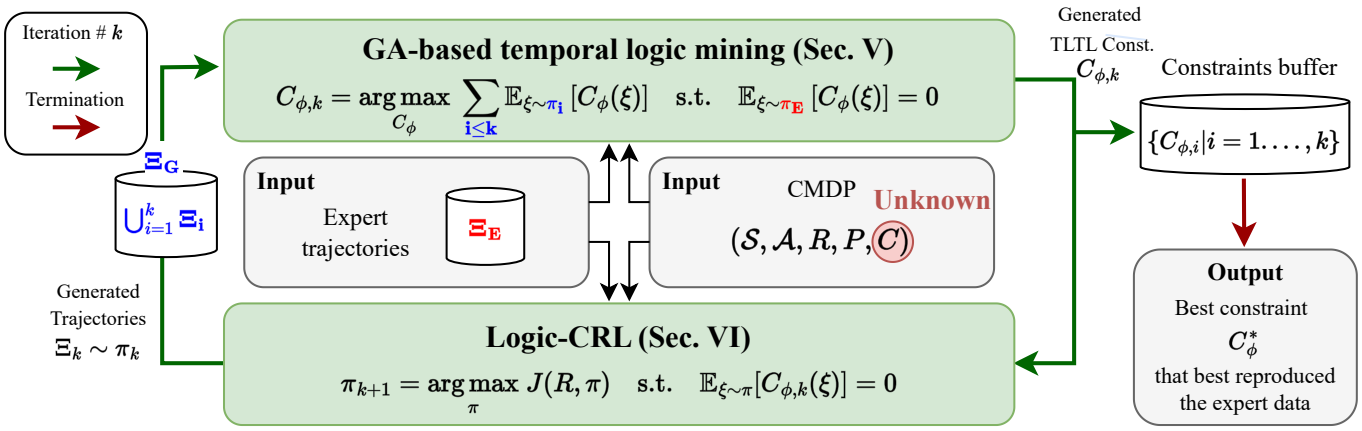

Fig. 2. Overall architecture of ILCL that finds a truncated linear temporal logic (TLTL) constraint by iteratively alternating constraint-and-policy optimizations given a set of expert trajectories Ξ_E. At the k-th iteration, the constraint optimization, **GA-based temporal-logic mining** (Top middle), seek to generate a TLTL constraint $C_{\phi,k}$ that maximizes the constraint cost of sampled trajectories. The policy optimization, **Logic-CRL** (Bottom middle), learns a policy π_{k+1} satisfying the TLTL constraint generated. We then sample and add trajectories into a buffer Ξ_G. By alternating the two optimizers, ILCL selects the best constraint C^*_ϕ, from the generated constraint set, that best explains the expert trajectories.

1) Initial population generation: GA-TL-Mining begins by creating an initial population of logic trees Φ that consists of basis and random logic trees:

$$\Phi = \underbrace{\{\phi_1, \ldots \phi_{N_{BSS}}\}}_{\text{Basis logic trees}} \cup \underbrace{\{\phi_1, \ldots \phi_{N_{RND}}\}}_{\text{Random logic trees}}, \tag{8}$$

where N_{BSS} and N_{RND} are the number of trees in each respective subset.

We generate the basis trees based on the current state $\mathbf{s} \in \mathbb{R}^\kappa$. From this state, we create 2κ pAPs using a sign a and a state element $\mathbf{s}^{(i)}$ for each $i \in [1, \kappa]$. The generation of logic trees follows forms:

$$\bigcirc\mu, \Diamond\mu, \Box\mu, \mu_1 \,\mathcal{U}\, \mu_2 \text{ and } \mu_1 \mathcal{R}\, \mu_2, \tag{9}$$

where μ, μ_1, and μ_2 are arbitrary pAPs from the state $\mathbf{s}$. In this work, to reduce computational complexity, we manually select $\kappa' < \kappa$ number of elements so that we finally generate a total of $8\kappa'^2 + 6\kappa'$ logic trees.

Next, to ensure a diverse initial population, we generate N_{RND} random trees by modifying basis trees. We start by randomly selecting target basis trees, in which we randomly choose a node. We then add a randomly selected operator as a parent node to the selected node. Note that we limit the maximum depth of trees to a predefined threshold. Finally, the size of the current population becomes $N = N_{BSS} + N_{RND}$.

2) Parent population selection: Our algorithm selects N_p logic trees with the highest regularized fitness score from the current population $\Phi = \{\phi_1, \ldots \phi_N\}$. We use the selected trees as parents for generating the next population in step 3. As ROGE [44], we define the regularized fitness score $\mathcal{F}_k^\Phi(\phi)$ as

$$\mathcal{F}_k^\Phi(\phi) = \mathcal{F}_k(\phi) - \zeta \left(\frac{1}{N_p} \sum_{j=1}^{|\Phi|} [\#\text{nodes in } \phi_j] \right)^\eta, \tag{10}$$

where ζ and η are hyperparameters. For the first term, fitness function of Eq. (7), we evaluate each ϕ with the best parameter values that maximize the fitness function $\mathcal{F}_k$. We estimate these values through dual annealing [59], as described in [37]. We apply the second term (i.e., a regularizer) to force GA-TL-Mining to learn the simplest constraint, thereby reducing ill-posedness in mining.

3) Offspring generation: Given the current population $\Phi = \{\phi_1, \ldots, \phi_{N_p}\}$, our algorithm generates offspring applying three genetic operators: `CROSSOVER`, `MUTATION-R`, and `MUTATION-A`. We first randomly select two logic trees, ϕ_1 and ϕ_2, using the tournament selection method in [41]. We then apply each operator in the following manner:

- `CROSSOVER`: we modify the selected trees, ϕ_1 and ϕ_2, by exchanging the randomly selected subtrees with each other.
- `MUTATION-R`: we first randomly select a node in the tree ϕ_1. We then either replace it into another node with the same type, such as pAP or operator, or remove it from the tree.
- `MUTATION-A`: we add a randomly selected logic operator as the parent node of a randomly chosen node in the tree ϕ_1.

Here, `MUTATION-A` enables GA-TL-Mining to efficiently explore the large search space by allowing not only semantically consistent operations but also more precise trees. For example, `MUTATION-A` changes $\mu_1 \,\mathcal{U}\, \mu_2$ to $\mu_1 \,\mathcal{U}\, \Box\mu_2$ by adding the $\Box$ operator without changing the implication (i.e., $\xi \models \mu_1 \,\mathcal{U}\, \Box\mu_2 \rightarrow \xi \models \mu_1 \,\mathcal{U}\, \mu_2$ for a state sequence ξ).

Our method alternates between steps 2 and 3. To narrow the search space, we prevent the generation of offspring that have already appeared in previous populations. Further, we ensure that all trees in the population to inherit their temporal properties by requiring all non-root nodes to have ancestors with temporal operators. Lastly, we simplify each logic tree for logical simplicity, enforcing rules such as $\Box\Box = \Box$, to avoid unnecessarily lengthy logics. We finally returns a generated TLTL constraint.

Note that we reuse previously discovered TLTL constraints for the subsequent mining process as guidance by adding them

into the initial population after populating the basis trees.

VI. Logic-Constrained Reinforcement Learning

We introduce logic-constrained RL (Logic-CRL) that learns policies to satisfy TLTL constraints obtained from GA-TL-Mining. We formulate the Logic-CRL as a Lagrangian-based CRL [53] on a product CMDP we define for TLTL constraints. Below we describe the product CMDP as well as the Lagragian approximation processes.

A. Product Constrained Markov Decision Process (PCMDP)

We construct a product CMDP, we call PCMDP, that is a synchronous structure between the transition system in the CMDP and deterministic finite automata (DFA) converted from TL constraints.

A DFA is a tuple $(Q_\phi, \Sigma_\phi, \delta_\phi, q_{0,\phi}, F_\phi)$, where Q_ϕ is a set of states, $\Sigma_\phi = 2^{\mathbf{AP}}$ is a power set of APs, $\delta_\phi : Q_\phi \times \Sigma_\phi \to Q_\phi$ is a deterministic transition function, $q_{0,\phi} \in Q_\phi$ is an initial state, and $F_\phi \subset Q_\phi$ is a set of accepting states. The product of DFA with CMDP is then a tuple $(\mathcal{S}_\phi, \mathcal{A}, \mathcal{L}_\phi, P_\phi, R, C_\phi)$, where $\mathcal{S}_\phi = \mathcal{S} \times Q_\phi$ is an augmented state space, $\mathcal{L}_\phi : \mathcal{S} \to \Sigma_\phi$ is a labeling function that assigns a set of APs to each state, and $P_\phi : \mathcal{S}_\phi \times \mathcal{A} \times \mathcal{S}_\phi \to [0, 1]$ is a transition probability function expressed as

$$P_\phi\left((\mathbf{s}_t, q_t), \mathbf{a}_t, (\mathbf{s}_{t+1}, q_{t+1})\right) = \mathbb{1}\left(\delta_\phi\left(q_t, \mathcal{L}_\phi(\mathbf{s}_{t+1})\right) = q_{t+1}\right) \cdot P(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \tag{11}$$

where q_t and q_{t+1} are DFA states ($\in Q_\phi$) at step t and $t+1$, respectively. Note that, in this work, we focus on PCMDPs with optimal finite-horizon policies independent of the remaining time horizon. For example, in finite-horizon CMDPs, optimal policies may exhibit non-Markovian behaviors, such as staying at high-reward states and transitioning to acceptance states only at the last step. To prevent these behaviors in PCMDPs, we consider tasks, where high rewards are exclusively associated with acceptance states.

Finally, our product CMDP enables the synthesis of a policy that takes the TLTL constraint ϕ into account while maximizing accumulated rewards. However, optimizing policies with TLTL constraints presents challenges due to their sparse nature and the delayed evaluation at the end of episodes.

B. Lagrangian-based Constrained Reinforcement Learning

We introduce a Lagrangian-based CRL resolving the problem of non-differentiable and temporally delayed evaluation of TLTL constraints in policy learning. The non-differentiable (i.e., sparse) nature comes from Boolean semantics of TLTLs. To figure it out, we incorporate quantitative semantics with robustness function $\rho(\xi, \phi)$ to define a bounded constraint-cost function $C'_\phi \in [0, 1]$ as:

$$C'_\phi(\xi) = \alpha C_\phi(\xi) + \frac{1-\alpha}{\Upsilon}\text{CLIP}[-\rho(\xi, \phi), 0, \Upsilon], \tag{12}$$

where $\alpha \in [0, 1]$ is an adjustable weight, $\Upsilon \in \mathbb{R}^+$ is an upper bound to limit the impact of large constraint violations. CLIP truncates the negative robustness value within the range $[0, \Upsilon]$. However, the evaluation of constraint cost function is possible at the termination of an episode ξ only.

To handle the delayed evaluation, we introduce a constraint redistribution method that spreads out the constraint-cost values of Eq. (12) across all timesteps within the trajectory ξ, similar to the reward redistribution in IRCR [18]. Our redistribution is a trajectory-space smoothing of the constraints in Eq. (6) transforming a TLTL trajectory constraint into a state-action constraint for Lagrangian-based CRL.

In detail, let P_0 be a initial state distribution and β be a behavioral policy: $\beta : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Defining a trajectory distribution $P_\beta(\xi) = P_0(\mathbf{s}_0) \prod_{t=0}^{T} P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\beta(\mathbf{a}_t|\mathbf{s}_t)$, we introduce a new state-action constraint $\tilde{C}'_\phi$ as

$$\tilde{C}'_\phi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\xi_\beta \sim p_\beta}\left[C'_\phi(\xi_\beta; \mathbf{s}_t, \mathbf{a}_t)\right], \tag{13}$$

where $\tilde{C}'_\phi(\xi_\beta; \mathbf{s}_t, \mathbf{a}_t)$ is a TLTL constraint of trajectory ξ_β containing a state-action pair, $(\mathbf{s}_t, \mathbf{a}_t)$. With this smoothed state-action constraint, we finally formulate a Lagrangian dual problem of Eq. (6):

$$\arg\min_{\lambda \geq 0} \max_{\pi} \mathop{\mathbb{E}}_{\xi \sim \pi}\left[\sum_{\mathbf{s}_t, \mathbf{a}_t \sim \xi}\left(R(\mathbf{s}_t, \mathbf{a}_t) + \lambda\left(\tilde{C}'_\phi(\mathbf{s}_t, \mathbf{a}_t) - d\right)\right)\right], \tag{14}$$

where λ is a Lagrangian multiplier and d is a non-zero constraint budget. We particularly set $d = \frac{\varepsilon\alpha}{N_\xi}$, where ε is a small positive value and N_ξ is the number of sampled trajectories used for expectations in Eq. (14). This budget d ensures that the learned policy satisfies the zero violation. After learning a constrained optimal policy π_{k+1}, we then sample zero-violation constrained trajectories Ξ_{new} as a part of Ξ_{k+1} ($= \Xi_k \cup \Xi_{new}$) for the input of GA-TL-Mining.

In this work, we sample ξ_β from the current policy π and previous policies $\pi_1, ..., \pi_k$ instead of β, following the soft actor-critic (SAC) [22] version of IRCR. We then update the Lagrangian multiplier using the PID-Lagrangian method [51]. Further, to boost the policy learning in SAC, we initialize a replay buffer with the expert trajectories and the previously sampled trajectories that already satisfy the given constraint similar to [7, 25, 57].

VII. Experimental Setup

We conduct quantitative and qualitative evaluations of the proposed ICLR algorithm against state-of-the-art baselines in simulated environments. We also demonstrate the transferability of the learned constraints to real-world settings.

A. Implementation with Simulated Environments

We introduce three benchmark simulation environments: *navigation*, *wiping*, and *peg-in-shallow-hole*.

1) *Navigation*: Fig. 3 (Left) shows a point agent at $\mathbf{x}^{Agt} \in \mathbb{R}^2$ navigating toward a goal position $\mathbf{x}^{Goal} \in \mathbb{R}^2$ satisfying temporal constraints that prevent or allow accessing to randomly distributed regions labelled with colors. Let $\mathbf{p}^R$, $\mathbf{p}^G$, and $\mathbf{p}^B$ represent the polar coordinates ($\in \mathbb{R}^2$) of the nearest red, green, and blue regions, respectively. Each coordinate includes

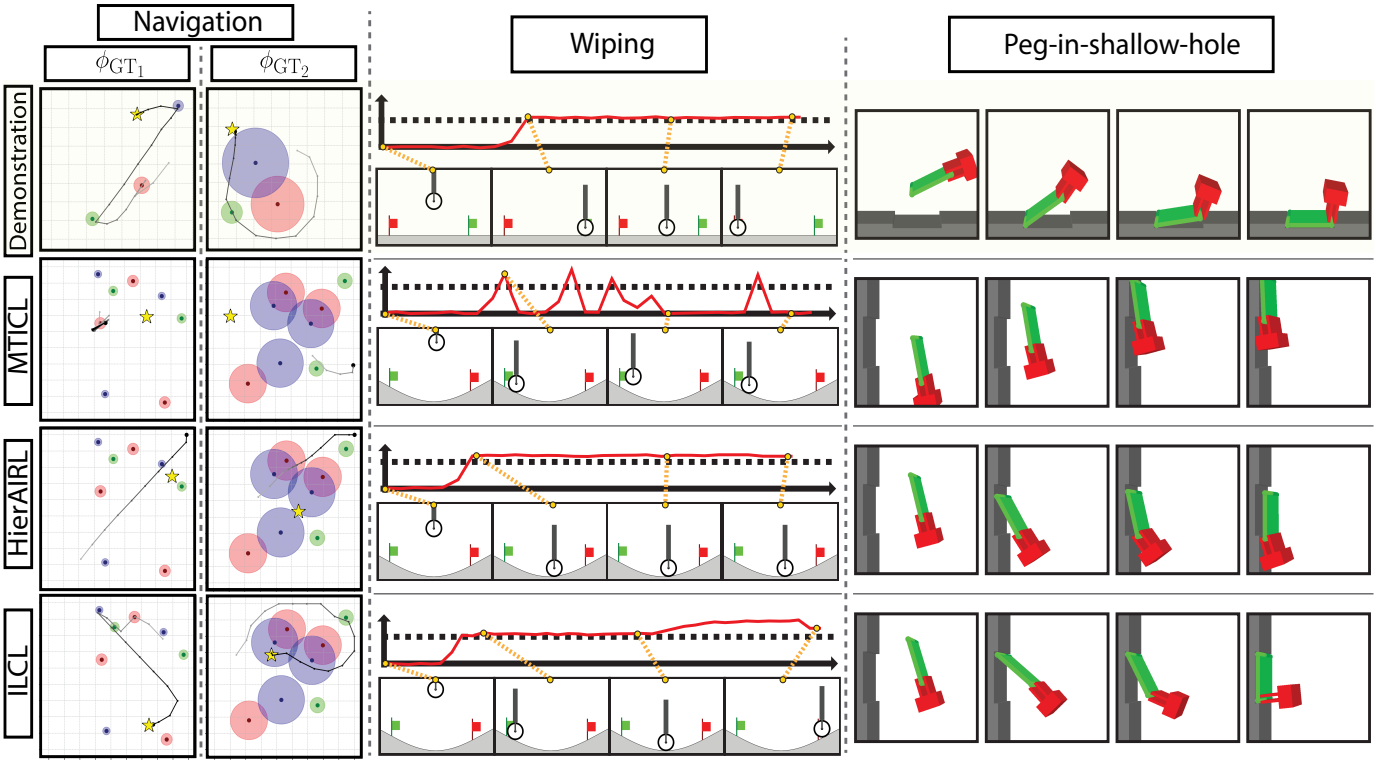

Fig. 3. Comparison of temporal constraint learning and transfer performance in four simulated navigation and manipulation tasks. In each task (column), ILCL and two baseline methods (i.e., MTICL and Hier-AIRL) first learn temporal constraints from demonstrations in the first row. Then, in novel environments, we train a behavior policy using the learned constraints or rewards to compare the reproduction performance of demonstration-like constrained behaviors. **Left**: in the *navigation* tasks, yellow shapes and black traces represent goal locations and trajectories, respectively. **Middle**: in the *wiping* task, green and red flags indicate start and goal locations, respectively. Red curve represents the observed contact force, and black-dot line represent the contact threshold from the demonstration. **Right**: in the *peg-in-shallow-hole* task, red, green, and gray objects represent a parallel jaw gripper, a peg, and a hole, respectively.

relative distance p_{dist} and angle p_{ang} from the agent. For example, $\mathbf{p}^R = (p^R_{dist}, p^R_{ang})$.

We define two ground-truth logic constraints, ϕ_{GT1} and ϕ_{GT2}, to record 20 demonstrations ($T = 25$) for each:

$$\phi_{\text{GT1}} = \underbrace{\Box(p^R_{dist} > 0.2) \wedge (p^B_{dist} > 0.25\, \mathcal{U}\, p^G_{dist} < 0.08)}_{\text{Always avoid all red regions. Blue regions become accessible only after the agent reaches any green region.}}$$

$$\phi_{\text{GT2}} = \underbrace{\Diamond(p^R_{dist} < 0.06 \wedge \Diamond(p^G_{dist} < 0.05 \wedge \Diamond(p^B_{dist} < 0.04))),}_{\text{Visit red, green, and blue regions in order}}$$

To learn the underlying constraints, we define a CMDP with the following specifications:

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = (\mathbf{x}^{Agt}, \mathbf{x}^{Goal}, \mathbf{p}^R, \mathbf{p}^G, \mathbf{p}^B) \in \mathbb{R}^{13}\}$.
- $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} = \Delta\mathbf{x}^{Agt} \in \mathbb{R}^2\}$; A space of discrete-time velocity commands within $[-0.15, 0.15]$.
- $R = 1 - \tanh(5 \cdot \|\mathbf{x}^{Agt} - \mathbf{x}^{Goal}\|_2)$; A reward function to encourage the agent to reach a goal position.

In evaluations, we vary the number, size, and location of regions. Note that each environment includes a 1.5 times larger operation space and a twice longer time horizon, $T = 50$.

2) *Wiping*: Fig. 3 (Middle) shows a roller agent at $\mathbf{x}^{Agt} \in \mathbb{R}^2$ wiping the ground surface between the start point A (green) and the goal point B (red). The roller applies a contact force f_c that exceeds a certain threshold when touching the surface.

Let ${}^A\mathbf{x}^{Agt}$ and ${}^B\mathbf{x}^{Agt}$ are the positions of the agent relative to the start and goal points, respectively. For example, ${}^A\mathbf{x}^{Agt} = ({}^Ax^{Agt}, {}^Ay^{Agt})$.

We define a ground-truth logic constraint ϕ_{GT} as

$$\underbrace{f_c < 0.05 \mathcal{U}(f_c > 1.6\, \mathcal{R}\, ({}^Ax^{Agt} > -0.1 \wedge {}^Ax^{Agt} < 0.1)),}_{\text{Keep sufficient contact while the roller is between points A and B. Avoid contact otherwise.}}$$

which is to record 100 demonstrations ($T = 50$) with randomly selected start-and-goal positions and initial agent locations. To learn the underlying constraints, we define a CMDP with the specifications below:

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = ({}^A\mathbf{x}^{Agt}, {}^B\mathbf{x}^{Agt}, \dot{\mathbf{x}}^{Agt}, w^{Agt}, d_g, f_g) \in \mathbb{R}^9\}$, where $\dot{\mathbf{x}}^{Agt}$ and w^{Agt} are the agent's linear and angular velocities, respectively. d_g and f_g are agent's sensor measurements for distance and contact force from the ground surface, respectively.
- $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} = (f_x^{Agt}, f_y^{Agt}) \in \mathbb{R}^2\}$; A space of force commands applied to the agent.
- $R = R_{goal} + R_{contact} + R_{penalty}$; $R_{goal} = \mathbb{1}(|{}^Bx^{Agt}| < 0.1) - |{}^Bx^{Agt}| \cdot \mathbb{1}(|{}^Bx^{Agt}| > 0.1)$ is a goal reward to encourage the goal achievement, $R_{contact} = 2 \cdot \mathbb{1}(f_c > 0.05)$ is a contact reward designed to encourage wiping contact, and $R_{penalty} = -0.2 \cdot f_c^2$ is to penalize large contact force.

In evaluations, we optimize constrained policies across environments featuring random frequencies of sinusoidal curves. We use the Box2D simulator [10].

3) *Peg-in-shallow-hole*: Consider a *peg-in-shallow-hole* task in SE2, also referred to as *shallow-depth insertion* [31] and illustrated in Fig. 3 (Right). Let Σ_h denote the 2D hole frame, tilted at an angle θ^h. The task uses a parallel-jaw gripper positioned at ${}^{\Sigma_h}\mathbf{x}^G$, represented by $({}^{\Sigma_h}x^{peg}, {}^{\Sigma_h}y^{peg}, {}^{\Sigma_h}\theta^{peg}) \in \mathbb{R}^3$, where θ^{peg} represents the tilt angle of the peg in the global frame. The process starts with the gripper holding a thin peg at the pose ${}^{\Sigma_h}\mathbf{x}^{peg}$ and aims to place it in its side within hole, maintaining stable contact, as a temporal constraint. For convenience, we introduce two features: d^{peg}_{hole} and d^{peg}_{jaw}, which measure the peg edge-to-hole and peg edge-to-support jaw distances, respectively.

We then define a ground-truth logic constraint ϕ_{GT} to record 100 demonstrations ($T = 30$) involving randomly tilted holes ($\theta^h \in [-10°, 10°]$):

$$\phi_{\text{GT}} = \Diamond(\underbrace{{}^{\Sigma_h}\theta^{peg} > 34.88° \wedge ({}^{\Sigma_h}\theta^{peg} < 4.217° \,\mathcal{R}\, d^{peg}_{jaw} < 0.0053) \wedge \Box d^{peg}_{hole} < 0.0072}_{\text{Given the peg upright, lower the tilt while in holding and the peg edge is always in contact with the hole corner, eventually.}}). \tag{15}$$

During the demonstration collection, we vary start-and-goal configurations as well as initial agent poses. To learn the underlying constraints, we define a CMDP as

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = ({}^{\Sigma_h}\mathbf{x}^G, {}^{\Sigma_h}\dot{\mathbf{x}}^G, {}^{\Sigma_h}\mathbf{x}^{peg}, {}^{\Sigma_h}\dot{\mathbf{x}}^{peg}, {}^{\Sigma_h}\bar{\mathbf{x}}^G, d^{jaw}, \dot{d}^{jaw}, \bar{d}^{jaw}, d^{peg}_{hole}, d^{peg}_{jaw}, \cos\theta^h, \sin\theta^h) \in \mathbb{R}^{22}\}$, where ${}^{\Sigma_h}\bar{\mathbf{x}}^B$ is the desired pose for the next time step. d^{jaw}, $\dot{d}^{jaw}$ and $\bar{d}^{jaw}$ are the position, velocity, and desired position of the jaws, respectively. The last two elements are the positional embedding of the hole slope.
- $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} = (\Delta^{\Sigma_h}\bar{\mathbf{x}}^G, \Delta\bar{d}^{finger}) \in \mathbb{R}^4\}$; A space of discrete-time velocity commands for the gripper and its finger joint.
- $R = (1 - \tanh(5|{}^{\Sigma_h}y^{peg} - {}^{\Sigma_h}y^{peg*}|)) + 5 \cdot \mathbb{1}({}^{\Sigma_h}y^{peg} - {}^{\Sigma_h}y^{peg*})$, where where ${}^{\Sigma_h}y^{peg}$ and ${}^{\Sigma_h}y^{peg*}$ are the current and desired vertical position of the peg in terms of the hole frame Σ_h.

In evaluations, we create four environments with holes tilted at $-90°$, $-45°$, $45°$ or $90°$ and increase the time horizon into $T = 60$. We use the PyBullet simulator [15].

B. Quantitative Evaluation Study with Baselines

We evaluate ILCL and baseline methods in both training (seen) and testing (unseen) environments. In the training environments, we estimate a constraint and its associate constrained policy for each random seed. We generate 10^3 trajectories from each policy by varying the start or goal positions of initial conditions. Overall, we produce 10^4, $5 \cdot 10^3$, and $5 \cdot 10^3$ trajectories using 10, 4, and 4 random seeds for the *navigation*, *wiping*, *peg-in-shallow-hole* tasks, respectively.

In the testing environments, we transfer the learned constraints (i.e., 10, 5, and 5 constraints for each task, respectively) to novel, randomly created environments. For examples, in the *navigation* task, we vary the locations of regions, and in *peg-in-shallow-hole*, we modify the slope of the hole. We create 10, 4, and 4 new environment for the three tasks, respectively. For each constraint and environment, we estimate a constrained policy and generate 10^3 trajectories by varying the start or goal positions within these environments. In total, we produce 10^6, $2 \cdot 10^4$, and $2 \cdot 10^4$ trajectories. We assess performance using the following metrics across all trajectories:

- **REW**: the average of episode rewards from trajectories.
- **VR**: the violation rate for a ground-truth constraint ϕ_{GT}. We classify any timestep of violations as an episode violation.
- **TR**: the average of truncated negative robustness values, where each value comes from $\max(-\rho(\xi, \phi_{\text{GT}}), 0)$,

We evaluate ILCL against two types of baseline methods: ICL and IRL. The ICL methods include numeric state-action (i.e., Markovian) constraint learning approaches:

- **ICRL** [40]: An inverse CRL (ICRL) method with maximum likelihood constraint inference.
- **TCL** [27]: A transferable constraint learning (TCL) method by reward decomposition.
- **MTICL** [32]: A multi-task ICL (MTICL) method with a two-player zero-sum game scheme (i.e., basis for ILCL).

The IRL methods, particularly hierarchical IRL (HIRL), generate policies that operate at multiple levels of temporal abstraction:

- **Option-GAIL** [29]: An HIRL extension of generative adversarial imitation learning (GAIL) [26] that models a task's hierarchy using options, matching option-occupancy measures of demonstrations,
- **Hier-AIRL** [11]: An HIRL extension of adversarial inverse reinforcement learning (AIRL) [17] that recover hierarchical policies using the option framework [52].

In evaluations, we use the same input features for ICL baselines as for ILCL. For IRL baselines, we provide additional reward-related features to facilitate reward-only learning.

C. Qualitative Transfer Study in Real World

We perform qualitative transfer studies in real-world *peg-in-shallow-hole* environments, as shown in Fig. 4. The environment consists of a 7-DoF Franka Emika Panda arm, an external RGB-D camera, a peg, and a tilt of shallow hole. After learning the stable-contact constraint through simulations, we enable the robot to transfer the constraint and reproduce demonstration-like constrained behaviors in novel real-world environments.

To show the transferability, we vary the slope of the hole within $[-25°, 60°]$. To detect the state of the slope and the peg, we use a AprilTag detector with attaching tags on the ground, hole, and peg. The learned constrained policy applies discrete-time velocity command and resulting desired pose into the Cartesian PD controller with 1 kHz (RL controller with 5 Hz).

VIII. Evaluation Results

A. Quantitative Analysis through Simulations

We analyze the constraint learning performance of our proposed method, ILCL, and baselines across three simulated environments with four ground-truth logic constraints. As

Method \ Env.		*Navigation* with ϕ_{GT1}			*Navigation* with ϕ_{GT2}			*Wiping*			*Peg-in-shallow-hole*		
		REW	VR [%]	TR	REW	VR [%]	TR	REW	VR [%]	TR	REW	VR [%]	TR
Expert		12.64 ± 2.83	0	0	9.52 ± 2.70	0	0	77.56 ± 10.99	0	0	56.02 ± 18.67	0	0
ICL	ICRL	5.10 ± 3.65	97.9 ± 3.7	0.379 ± 0.080	1.329 ± 0.973	100.0 ± 0.0	0.478 ± 0.047	**80.68** ± 1.76	98.9 ± 0.1	1.025 ± 0.008	**63.19** ± 32.01	100.0 ± 0.0	0.073 ± 0.009
	TCL	4.58 ± 1.95	82.9 ± 11.1	0.218 ± 0.057	3.35 ± 3.85	100.0 ± 1.0	0.385 ± 0.110	42.12 ± 34.31	99.4 ± 0.8	1.290 ± 0.197	19.78 ± 22.91	100.0 ± 0.	0.108 ± 0.013
	MTICL	5.71 ± 0.75	98.5 ± 0.7	0.264 ± 0.026	4.31 ± 1.02	100.0 ± 0.0	0.301 ± 0.011	55.67 ± 31.37	95.6 ± 7.7	0.515 ± 0.427	**142.36** ± 16.15	99.9 ± 0.2	0.039 ± 0.004
IRL	Option -GAIL	2.06 ± 0.64	95.3 ± 2.5	0.360 ± 0.100	3.43 ± 0.83	100.0 ± 0.0	0.443 ± 0.06	-11.25 ± 25.74	96.3 ± 2.7	0.929 ± 0.107	5.05 ± 5.33	93.9 ± 11.3	0.240 ± 0.173
	Hier-AIRL	2.02 ± 0.34	82.9 ± 12.2	0.078 ± 0.021	3.53 ± 2.91	100.0 ± 0.0	0.406 ± 0.018	-54.33 ± 37.96	21.0 ± 5.5	0.016 ± 0.006	3.30 ± 3.99	87.5 ± 24.1	0.178 ± 0.132
ILCL^{Human}		**12.63** ± 1.27	**7.1** ± 4.1	**0.017** ± 0.008	**5.25** ± 3.45	**7.1** ± 25.7	**0.015** ± 0.079	75.52 ± 11.77	**3.0** ± 17.0	**0.002** ± 0.034	52.22 ± 0.91	**13.1** ± 3.5	**0.0001** ± 0.0001
ILCL		**11.62** ± 4.92	**32.5** ± 46.8	**0.054** ± 0.122	**4.23** ± 3.30	**24.0** ± 42.7	**0.014** ± 0.075	**75.56** ± 12.17	**16.3** ± 36.9	**0.006** ± 0.042	55.88 ± 2.04	**26.2** ± 10.9	**0.0002** ± 0.0002

TABLE I: Comparison of the proposed ILCL and five baseline methods in four training environments. The top row, labeled 'Expert,' shows the evaluation results from human expert trajectories without constraint violations. The last two rows display our ILCL results using two types of constraint selection strategies; ILCL selects the best constraint for evaluation, aligned with expert demonstrations in the recovered constraints, while ILCL^{Human} uses a constraint chosen by a human expert. The numbers in each cell indicate the mean and standard deviation corresponding to the column's specified metric. Values that are bold and underlined denote the best statistically significant results, whereas bold values indicate the second-best results in each column.

Method \ Env.		*Navigation* with ϕ_{GT1}			*Navigation* with ϕ_{GT2}			*Wiping*			*Peg-in-shallow-hole*		
		REW	VR [%]	TR	REW	VR [%]	TR	REW	VR [%]	TR	REW	VR [%]	TR
ICL	ICRL	8.53 ± 4.03	94.2 ± 2.1	0.252 ± 0.035	2.27 ± 0.17	100.0 ± 0.1	0.208 ± 0.024	**203.47** ± 4.21	99.7 ± 0.2	1.056 ± 0.021	100.92 ± 8.51	100.0 ± 1.0	0.149 ± 0.017
	TCL	27.61 ± 3.51	97.2 ± 0.5	0.256 ± 0.005	**28.76** ± 1.66	100.0 ± 0.0	0.331 ± 0.006	**249.71** ± 23.51	100.0 ± 0.0	1.214 ± 0.149	136.95 ± 14.39	100.0 ± 0.0	0.209 ± 0.063
	MTICL	1.81 ± 0.38	82.1 ± 6.8	0.177 ± 0.082	2.48 ± 0.64	98.6 ± 1.4	0.352 ± 0.046	47.92 ± 32.00	98.0 ± 3.0	0.690 ± 0.216	13.14 ± 2.66	99.5 ± 1.0	0.290 ± 0.008
IRL	Option -GAIL	1.63 ± 0.48	95.5 ± 1.8	0.307 ± 0.060	1.23 ± 0.29	99.9 ± 0.3	0.379 ± 0.033	-82.97 ± 31.52	98.8 ± 1.2	1.076 ± 0.284	4.86 ± 3.31	98.6 ± 1.7	0.260 ± 0.090
	Hier-AIRL	1.32 ± 0.32	93.8 ± 2.4	0.255 ± 0.040	0.82 ± 0.19	99.9 ± 0.3	0.444 ± 0.030	53.96 ± 53.75	93.6 ± 7.3	0.793 ± 0.375	6.38 ± 5.06	98.9 ± 1.4	0.278 ± 0.052
ILCL^{Human}		**32.26** ± 1.09	**1.3** ± 2.0	**0.002** ± 0.001	25.45 ± 1.96	**23.0** ± 5.2	**0.025** ± 0.006	162.21 ± 37.09	**13.0** ± 15.0	**0.026** ± 0.029	**226.36** ± 20.24	**32.7** ± 10.2	**0.006** ± 0.003
ILCL		**32.13** ± 2.31	**24.4** ± 9.3	**0.028** ± 0.010	**26.53** ± 2.08	**37.2** ± 13.3	**0.0285** ± 0.008	167.71 ± 7.39	**18.2** ± 1.6	**0.022** ± 0.027	**239.97** ± 10.79	**63.3*** ± 20.2	**0.009** ± 0.002

TABLE II: Comparison of the proposed ILCL and five baseline methods in novel environments. *-the high violation is due to VR's binary nature as well as unstable contact in physics simulator during the insertion resulting in unconstrained behaviors and obtaining high goal rewards.

shown in Table I, ILCL consistently outperforms all baselines demonstrating the second-lowest violation rates (VR) in all scenarios. In contrast, conventional ICL and IRL methods exhibit significantly higher violation rates, approaching 100%. Notably, in the *navigation* scenario with ϕ_{GT1}, ILCL achieves a maximum violation rate of 32.5%, which is about 50% lower than the best performing baseline, Hier-AIRL, while securing higher episode rewards. ILCL selects the best constraint from logic constraints generated through GA-TA-Mining by assessing the robustness values against expert demonstrations. We then use this best constraint to find constrained policies and generate trajectories for evaluation metric calculations.

In addition, an ILCL method, where the best constraint is chosen by a human expert, referred to as ILCL^{Human}, achieves the lowest violation rates, with a maximum of 13.1%, across all scenarios. These rates are the lower bound for the ILCL's selection strategies. However, due to the binary nature of violation rates, our result even classifies a minor violation at a single timestep as a full binary violation. To distinguish the detail of constraint quality, a continuous measure is necessary.

Alternatively, to provide a more detailed assessment than the binary measure of violation rates, we investigate the degree of violations using the TR metric. ILCL consistently demonstrates either the lowest or second lowest mean TR values, significantly differing from those of the baselines. These results underscore the complexity of temporal-constraint learning, even in seen environments, to conventional ICL or IRL algorithms. Although the ICRL and MTICL approaches yield high episode rewards in the last two scenarios, these rewards are from the unconstrained behaviors, leading significantly high violation rates, which are undesirable in robotic tasks. Moreover, IRL shows the lowest episode rewards in

	Method	Best TLTL constraint	ALL			SATISFIED		
			REW	VR (%)	TR	REW	VR (%)	TR
Navigation with ϕ_{GT1}	ILCL$^{\text{human}}$	$\Box p^{R}_{dist}{>}0.208$ $\wedge(p^{B}_{dist}{>}0.267\,\mathcal{U}\,p^{G}_{dist}{<}0.080)$	33.33	0.4	0.0012	33.43	0	0
	ILCL	$\phi^{*}_{\text{ILCL}}{=}(p^{B}_{dist}{>}0.267$ $\wedge\Box p^{R}_{dist}{>}0.205)\mathcal{U}p^{G}_{dist}{<}0.081$	32.56	1.7	0.0015	32.51	1.31	$8\cdot10^{-6}$
Navigation with ϕ_{GT2}	ILCL$^{\text{human}}$	$(^{R}_{dist}{<}0.06\,\mathcal{R}$ $(\Diamond p^{B}_{dist}{<}0.04\,\mathcal{U}\,p^{G}_{dist}{<}0.048)$	29.49	15.6	0.0167	29.26	14.0	0.012
	ILCL	$p^{R}_{dist}{<}0.06\,\mathcal{R}$ $(\Diamond p^{B}_{dist}{<}0.04\,\mathcal{U}\,p^{G}_{dist}{<}0.048)$	30.32	24.2	0.0234	25.52	18.5	0.013
Wiping	ILCL$^{\text{human}}$	$f_c{<}0.05\,\mathcal{U}\,(\Box p8{>}1.674\,\mathcal{R}$ $(^{A}\mathbf{x}^{Agt}{<}0.099'\,p0{>}-0.077)))$	133.23	2.5	0.0025	173.20	0	0
	ILCL	$\Box p8{>}1.661\,\mathcal{R}$ $(p8{<}0.062\,\mathcal{U}\,(p0{<}0.099{\wedge}p0{>}-0.076)))$	168.97	6	0.0064	167.34	0	0
Peg-in-shallow-hole	ILCL$^{\text{human}}$	$\Diamond(^{\Sigma h}\,\theta^{peg}{>}34.87°$ $\wedge(^{\Sigma h}\,\theta^{peg}{<}4.296°\,\mathcal{R}\,d^{peg}_{jaw}{<}0.0053)$ $\wedge\Box d^{peg}_{hole}{<}0.0072)$	248.72	19.0	0.0024	255.25	1.4	$1.7\cdot10^{-6}$
	ILCL	$\Diamond(^{\Sigma h}\,\theta^{peg}{>}34.68°\wedge{}^{\Sigma h}\,\theta^{peg}{<}39.44°$ $\wedge(^{\Sigma h}\,\theta^{peg}{<}0.241°\,\mathcal{R}\,d^{peg}_{jaw}{<}0.0055)$ $\wedge\Box d^{peg}_{hole}{<}0.0072)$	229.54	22.0	0.0027	235.44	12.0	$5.0\cdot10^{-5}$

TABLE III: Comparison of TLTL constraints estimated by ILCL$^{\text{Human}}$ and ILCL. We assess the constraints using evaluation metrics applied to **ALL** trajectories and **SATISFIED** trajectories generated by the constrained policy through Logic-CRL. The **SATISFIED** trajectories represent the subset that does not violate the discovered constraints, allowing us to isolate the impact of CRL failures on constraint quality evaluation.

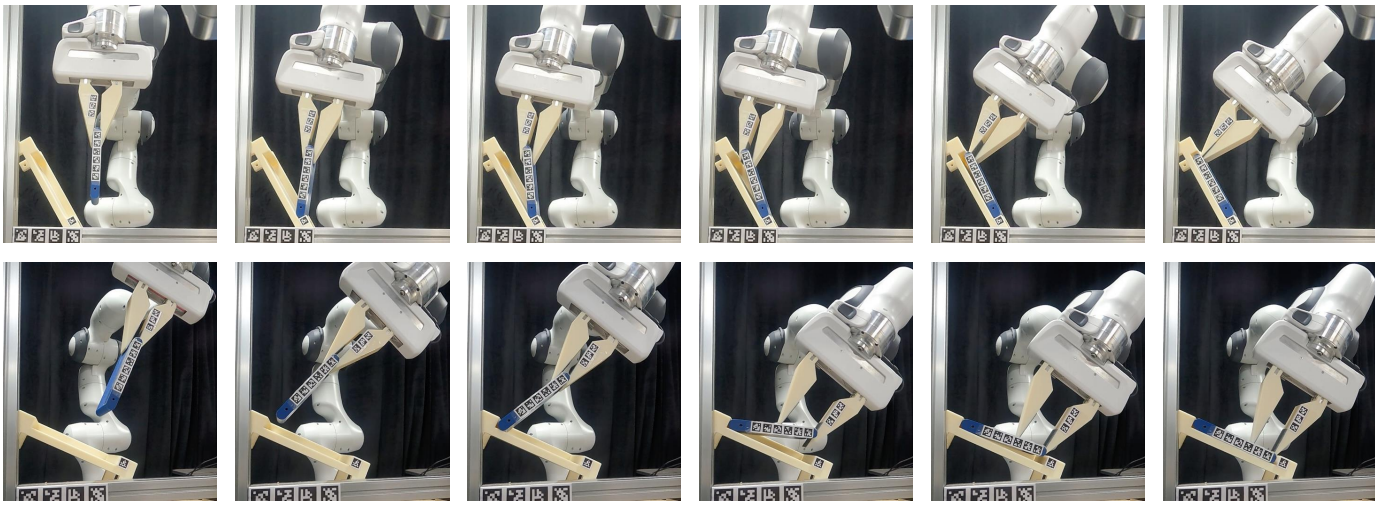

Fig. 4. Demonstration of ILCL's constraint transfer capability in the *peg-in-shallow-hole* environments. By creating real-world environments with holes tilted at 60° and −25°, we enable the Panda gripper to find a constrained policy using the learned constraints and reproduce a demonstration-like robust insertion with stable contact in each unseen environment.

most scenarios, reflecting the difficulty of optimizing both task rewards and constraints simultaneously.

In Table II, we assess the transferability of constraints learned from ILCL across novel configuration of environments. ILCL consistently shows either the lowest or second lowest violation rates as well as negative robustness values, while receiving high episode rewards, similar to the training results. In contrast, all baseline approaches exhibit violation rates exceeding 82.1%, highlighting the challenges associated with transferring temporal constraints. Further, in the *peg-in-shallow-hole* scenario, ILCL shows relatively high violation rates of 63.3%, even though the negative robustness is significantly low. This discrepancy arises not only from the binary nature of the violation rates but also from the Lagrangian approximation used in Logic-CRL, which softens constraints to favor high-reward behaviors at the expense of increasing violations in novel settings. In our simulations, we observed cases where the gripper reverses and successfully inserts the peg into the hole. Furthermore, methodologies often yield higher rewards than those reported in Table I. This occurs when approaches learn that is not fully constrained, thereby achieving high goal rewards due to the use of soft constraints in ICL or the lack of constraints in IRL.

Lastly, we evaluate the quality of TLTL constraints esti-

mated by ILCL$^{\text{Human}}$ and ILCL reducing the effect of Logic-CRL. We generate two types of trajectories: all trajectories, referred to as **ALL**, produced through Logic-CRL and its subset, referred to as **SATISFIED**, that do not violate the discovered constraints. Table III shows both ILCL$^{\text{Human}}$ and ILCL return ground-truth or similar structure of TLTL constraints while resulting in certain violation rates across all environments (see the **ALL** columns in Table III). As shown in the **SATISFIED** columns, all methods demonstrate significantly reduced violation rates indicating the suboptimal performance of CRL. ILCL$^{\text{Human}}$ particularly shows zero or near-zero rates except *Navigation* with ϕ_{GT2}. This indicates ILCL's capability of searching ground-truth like TL constraints.

B. Qualitative Demonstrations in Real World

Fig. 4 demonstrates the transferability of ILCL in novel *peg-in-shallow-hole* environments, where the Panda arm with a parallel jaw gripper successfully inserts a peg into unforeseen holes tilted at $60°$ and $-25°$, while maintaining stable contact between the hole and the peg held by the gripper. These demonstrations indicate ILCL can learn abstract logic constraints that are robustly transferable to real-world setups without human intervention. For each experiment, we optimized the constrained policy for 4 hours within 10000 steps of environment interactions.

IX. Conclusion

We introduced a inverse logic-constraint learning (ILCL) algorithm that identifies free-form transferable truncated linear temporal-logic (TLTL) constraint from demonstrations. Adopting a two-player zero-sum game learning scheme, ILCL provides a genetic algorithm-based TL mining (GA-TL-Mining) and logic-constraint reinforcement learning (Logic-CRL). GA-TL-Mining allows learning TLTL constraints from demonstrations without predefined templates, while Logic-CRL optimizes constrained policies with non-Markovian nature of constraints. Through quantitative and qualitative studies, our method outperformed state-of-the-art baseline methods resulting in demonstration-like behaviors with lower constraint violations even in novel environments. We also demonstrate the proposed ILCL's applicabilities in real-world peg-in-shallow-hole tasks.

X. Limitations

- **Time complexity**: A major limitation of ILCL is the cost-expensive logic-constraint search in GA-TA-Mining, due to the extremely large space of temporal logic specifications. Although this may lower the usability of the framework, to the best of our knowledge, this is the first implementation of TL-based constrain learning from demonstrations using CRL.
- **Soft constraint**: We introduce a Lagrangian-based Logic-CRL that softens constraints to encourage high-reward behaviors. However, this does not guarantee satisfaction of the discovered constraints as shown in Table I and Table II.
- **Ill-posedness**: The ill-posed nature of logical specifications may result in generating a logic constraint that is either narrow or broad compared to the ground-truth, leading to failures when applied to new settings. To reduce the ill-posedness, ILCL requires enough expert demonstrations that may not be feasible to obtain in real-world scenarios.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[2] Edgar A Aguilar, Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Ničković. Mining specification parameters for multi-class classification. In *International Conference on Runtime Verification*, pages 86–105. Springer, 2023.

[3] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *Runtime Verification: Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers 2*, pages 147–160. Springer, 2012.

[5] Mathijs Baert, Sam Leroux, and Pieter Simoens. Learning safety constraints from demonstration using one-class decision trees. In *Proceedings of NucLeaR Workshop, Neuro-Symbolic Learning and Reasoning in the Era of Large Language Models at AAAI 2024*, 2024.

[6] Mattijs Baert, Sam Leroux, and Pieter Simoens. Learning logic constraints from demonstration. In *NeSy2023: 17th International Workshop on Neural-Symbolic Learning and Reasoning*, pages 78–84, 2023.

[7] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[8] Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Nickovic. Survey on mining signal temporal logic specifications. *Information and Computation*, 289: 104957, 2022.

[9] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[10] Erin Catto. Box2d: A 2d physics engine for games. URL http://www.box2d.org.

[11] Jiayu Chen, Tian Lan, and Vaneet Aggarwal. Option-aware adversarial inverse reinforcement learning for robotic control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5902–5908. IEEE, 2023.

[12] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3387–3395, 2019.

[13] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[14] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[15] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[16] Giuseppe De Giacomo, Moshe Y Vardi, et al. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pages 854–860, 2013.

[17] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[18] Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. *Advances in Neural Information Processing Systems*, 33: 822–832, 2020.

[19] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[20] Ashish Gaurav, Kasra Rezaee, Guiliang Liu, and Pascal Poupart. Learning soft constraints from constrained expert demonstrations. In *The Eleventh International Conference on Learning Representations*.

[21] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *International Conference on Automated Planning and Scheduling*, 2018.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[23] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

[24] Tairan He, Weiye Zhao, and Changliu Liu. Autocost: Evolving intrinsic cost for zero-violation reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14847–14855, 2023.

[25] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[27] Jaehwi Jang, Minjae Song, and Daehyung Park. Inverse constraint learning and generalization by transferable reward decomposition. *IEEE Robotics and Automation Letters*, 9(1):279–286, 2023.

[28] Susmit Jha, Ashish Tiwari, Sanjit A Seshia, Tuhin Sahai, and Natarajan Shankar. Telex: Passive stl learning using only positive examples. In *International Conference on Runtime Verification*, pages 208–224. Springer, 2017.

[29] Mingxuan Jing, Wenbing Huang, Fuchun Sun, Xiaojian Ma, Tao Kong, Chuang Gan, and Lei Li. Adversarial option-aware hierarchical imitation learning. In *International Conference on Machine Learning*, pages 5097–5106. PMLR, 2021.

[30] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *ArXiv*, abs/2106.13906, 2021.

[31] Chung Hee Kim and Jungwon Seo. Shallow-depth insertion: Peg in shallow hole through robotic in-hand manipulation. *IEEE Robotics and Automation Letters*, 4 (2):383–390, 2019.

[32] Konwoo Kim, Gokul Swamy, Zuxin Liu, Ding Zhao, Sanjiban Choudhury, and Steven Z Wu. Learning shared safety constraints from multi-task demonstrations. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. Learning signal temporal logic through neural network for interpretable classification. In *2023 American Control Conference (ACC)*, pages 1907–1914. IEEE, 2023.

[34] Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. Tlinet: Differentiable neural network temporal logic inference. *arXiv preprint arXiv:2405.06670*, 2024.

[35] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.

[36] Guiliang Liu, Yudong Luo, Ashish Gaurav, Kasra Rezaee, and Pascal Poupart. Benchmarking constraint inference in inverse reinforcement learning. *arXiv preprint arXiv:2206.09670*, 2022.

[37] Wenliang Liu, Danyang Li, Erfan Aasi, Roberto Tron, and Calin Belta. Interpretable generative adversarial imitation learning. *arXiv preprint arXiv:2402.10310*, 2024.

[38] Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

[39] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-*

Tolerant Systems, pages 152–166. Springer, 2004.

[40] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International conference on machine learning*, pages 7390–7399. PMLR, 2021.

[41] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[42] Sara Mohammadinejad, Jyotirmoy V Deshmukh, Aniruddh G Puranic, Marcell Vazquez-Chanlatte, and Alexandre Donzé. Interpretable classification of time-series data using efficient enumerative techniques. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2020.

[43] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8:295–318, 1991.

[44] Laura Nenzi, Simone Silvetti, Ezio Bartocci, and Luca Bortolussi. A robust genetic algorithm for learning temporal specifications from data. In *Quantitative Evaluation of Systems: 15th International Conference, QEST 2018, Beijing, China, September 4-7, 2018, Proceedings 15*, pages 323–338. Springer, 2018.

[45] Dimitris Papadimitriou, Usman Anwar, and Daniel S Brown. Bayesian inverse constrained reinforcement learning. In *Workshop on Safe and Robust Control of Uncertain Systems (NeurIPS)*, 2021.

[46] Daehyung Park, Michael Noseworthy, Rohan Paul, Subhro Roy, and Nicholas Roy. Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning. In *Conference on robot learning*, pages 1005–1014. PMLR, 2020.

[47] Federico Pigozzi, Laura Nenzi, and Eric Medvet. Bustle: a versatile tool for the evolutionary learning of stl specifications from data. *Evolutionary Computation*, pages 1–24, 2024.

[48] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

[49] Guanren Qiao, Guiliang Liu, Pascal Poupart, and Zhiqiang Xu. Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations. *Advances in Neural Information Processing Systems*, 36, 2024.

[50] Dexter RR Scobee and S Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations*, 2020.

[51] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[52] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[53] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[54] Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

[55] Prashant Vaidyanathan, Rachael Ivison, Giuseppe Bombara, Nicholas A DeLateur, Ron Weiss, Douglas Densmore, and Calin Belta. Grid-based temporal logic inference. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5354–5359. IEEE, 2017.

[56] Marcell Vazquez-Chanlatte, Jyotirmoy V Deshmukh, Xiaoqing Jin, and Sanjit A Seshia. Logical clustering and learning for time-series data. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 305–325. Springer, 2017.

[57] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[58] Cameron Voloshin, Abhinav Verma, and Yisong Yue. Eventual discounting temporal logic counterfactual experience replay. In *International Conference on Machine Learning*, pages 35137–35150. PMLR, 2023.

[59] Yang Xiang, DY Sun, W Fan, and XG Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220, 1997.

[60] Sheng Xu and Guiliang Liu. Uncertainty-aware constraint inference in inverse constrained reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[61] Lunet Yifru and Ali Baheri. Concurrent learning of control policy and unknown safety specifications in reinforcement learning. *IEEE Open Journal of Control Systems*, 2024.