ILCL: Inverse Logic-Constraint Learning from Temporally Constrained Demonstrations

Author Names Omitted for Anonymous Review.

***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, which we call inverse logic-constraint learning (ILCL). Our method frames ICL as 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 syntax trees for parameterized truncated linear temporal logic (TLTL) without predefined templates. Subsequently, Logic-CRL finds a policy that maximizes task rewards under the constructed TLTL constraints via a novel constraint redistribution scheme. Our evaluations show ILCL outperforms state-of-the-art baselines in learning and transferring TL constraints on four temporally constrained tasks. We also demonstrate successful transfer to real-world peg-in-shallow-hole tasks.**

I. Introduction

Robot policy learning often involves constraints that account for user, task, and environmental restrictions, as shown in Fig. 1. To autonomously adopt these constraints, inverse constraint learning (ICL) methods recover constraints from demonstration [1]. While ICL typically targets global constraints, most tasks require spatial or temporal restrictions. For example, mobile robots must wait until the traffic sign turns green. To express such structured and interpretable restrictions, researchers often adopt temporal logic (TL), such as linear temporal logic (LTL) [2] and signal temporal logic (STL) [3].

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. For efficient searches, researchers often rely on predefined logic templates [4] or restrict operator sets [5]. However, the limited expressivity restricts applications in tasks. Therefore, a desired approach requires to derive free-form logic constraints.

Another challenge arises from the ill-posed nature of non-Markovian constraints. Given this ill-posedness, conventional approaches often focus on finding constraints without simultaneously considering the reward maximization in policy learning. This reduces the chance of finding optimal and transferable constraints from experts. Further, the non-Markovian property requires evaluating the history of state-action sequences, which yields sparse feedback during policy learning. These combined challenges complicate the integration of TL with conventional ICL frameworks [1] via inverse and forward constraint reinforcement learning (CRL) processes.

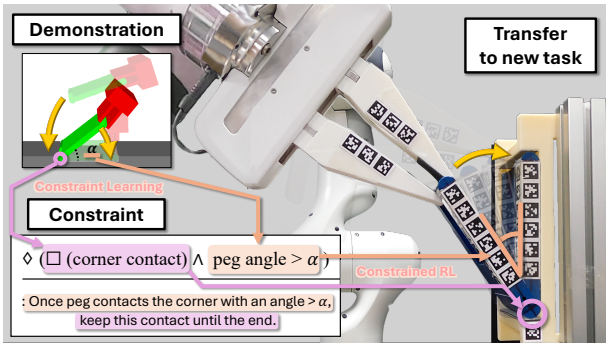


Fig. 1. An exemplar *peg-in-shallow-hole* task, which requires maintaining a peg-hole contact while inserting the peg using a parallel jaw gripper. By learning temporal logic constraints from demonstrations, our method ILCL successfully transfers demonstration-like constrained insertion behavior on a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

We propose inverse logic-constraint learning (ILCL), a free-form transferable TL-constraint learning algorithm. Our method involves constraint learning as a two-player zero-sum game, similar to [6], 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) [7] without predefined templates. Subsequently, Logic-CRL finds a policy that maximizes task rewards while adhering to the constructed TLTL constraints, measuring the degree of logic satisfaction across a trajectory using a robustness function. To handle the history dependency inherent in TL constraints in CRL, we introduce a constraint redistribution method, analogous to reward redistribution approaches [8], [9], converting trajectory-based constraints into state-action constraints. To the best of our knowledge, we provide the first CRL approach that generalizes to arbitrary TL constraints.

We perform a statistical evaluation of ILCL against state-of-the-art ICL baselines. Across four simulated navigation and manipulation benchmark environments, ILCL outperforms baselines in learning diverse forms of TL constraints, achieving the lowest constraint-violation scores while yielding high task reward returns comparable to expert behaviors. 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.

II. Related Work

Constraint learning and mining. ICL methods recover constraints from demonstrations. Early methods seek to learn numerical constraints in a discretized state space [10], while, with the advancement of neural representations, recent methods model constraints on continuous state or state-action

spaces [11], [12]. To enhance model scalability, researchers introduce multi-task [6] and multi-modal [13] approaches. However, numerical representations are often hard to interpret and transfer to new settings. Instead, we use logical representations for robust generalization.

Logic constraint learning is a variant of ICL. Early methods identify logic formulas adopting inductive logic programming [14], one-class decision tree [15], or data-driven rule induction [16]. However, these approaches often limit the outcomes 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 [4] or limited temporal expressions [5]. 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 [17]). For effective inference in the combinatorially large specification space, researchers often fix logic structures [18], simplify forms [19], discretize the search space [20], or restrict the search depth [21]. Alternatively, researchers enable tree-based genetic algorithms to explore the space of logic syntax trees exchanging subtrees without restrictions [22]. We extend the algorithm to compose a TL 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 [23]. The first category typically restrict actions or policies within a trust region [24]. While these connect 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) [25]. However, the long-term dependency in temporal logics may result in reward delays. To address this, researchers introduce neural networks to estimate trajectory returns [8] or redistribute rewards across time steps, smoothing the reward signals [9]. We extend this reward redistribution to TL-based CRL evaluating constraint violations at the end of episodes.

In addition, TL-based CRL approaches often prioritize logic satisfaction, neglecting the primary task objective within environments. To mitigate this, approaches weigh logic-based rewards [26], apply logic fragments only [27], or discretize the environment into problem-specific state spaces [28]. These strategies limit their generalizability across novel environments and the adoption of complex TL specifications. In contrast, our method operates on arbitrary TL in a model-free manner.

III. Preliminaries

A. Constraint Markov Decision Process (CMDP)

A CMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, C)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the constraint-cost function. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ defines 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 rewards and constraint costs, respectively.

To generate a sequence satisfying TL constraints, we introduce another CMDP with a TLTL constraint C_ϕ, which returns a positive value when a state sequence $\xi^{(s)} = (\mathbf{s}_0, ..., \mathbf{s}_T)$ violates the TLTL formula ϕ at any time. For brevity, we use state-action sequence, state sequence, and trajectory interchangeably and omit the arguments of $C(\mathbf{s}, \mathbf{a})$ and $C(\xi)$.

B. Truncated Linear Temporal Logic (TLTL)

We employ TLTL [7], a predicate temporal logic that extends LTL [2] and LTL over finite traces (LTL_f [30]). Each TLTL formula consists of atomic predicates (AP) of the form $a \cdot \mathbf{s}^{(i)} < b$, where $a \in \{-1, 1\}$, $b \in \mathbb{R}$ is a constant, and $\mathbf{s}^{(i)}$ refers to the i-th element of the state vector $\mathbf{s}$. The syntax of TLTL formulas with an AP μ is defined as follows:

$$\phi ::= \top \mid \bot \mid \mu \mid \neg\mu \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc \phi \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \mathcal{U} \phi_2 \mid \phi_1 \mathcal{R} \phi_2, \quad (1)$$

where $\top$ and $\bot$ denote Boolean True and False; $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are Boolean connectives; and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{R}$ (release), and $\bigcirc$ (next) are temporal operators. The syntax in Eq. (1) follows the positive normal form, where negations apply only to APs. We evaluate TLTL formulas against a finite state sequence $\mathbf{s}_{t:t+\Delta} = (\mathbf{s}_t, \ldots, \mathbf{s}_{t+\Delta})$ from time step t with the finite length Δ.

To incorporate learnable thresholds into TLTL, we introduce a parametric variant, we call pTLTL, inspired by parametric STL (pSTL) [31]. The parameterization replaces the constant b in each AP with a parameter θ, yielding a parametric AP (pAP) of the form $a \cdot \mathbf{s}^{(i)} < \theta$. For example, we represent a TLTL formula ϕ as a pTLTL formula $\phi_{\boldsymbol{\theta}}$:

$$\begin{aligned} \phi &= (\mathbf{s}^{(0)} < 1)\, \mathcal{U}\, (\mathbf{s}^{(1)} > 2) && \ldots \quad \text{(TLTL)}, \\ \phi_{\boldsymbol{\theta}} &= (\mathbf{s}^{(0)} < \boldsymbol{\theta}^{(0)})\, \mathcal{U}\, (\mathbf{s}^{(1)} > \boldsymbol{\theta}^{(1)}) && \ldots \quad \text{(pTLTL)}, \end{aligned} \quad (2)$$

where a parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}) \in \mathbb{R}^2$. In this work, we use the learnable pTLTL; for simplicity, we use TLTL ϕ and pTLTL $\phi_{\boldsymbol{\theta}}$ interchangeably and denote them as ϕ.

Then, the Boolean semantics of a formula ϕ defines a state sequence $\mathbf{s}_{t:t+\Delta}$ satisfies ϕ at time t if and only if $\rho(\mathbf{s}_{t:t+\Delta}, \phi, t) > 0$, where ρ is the robustness function [7]. This function provides quantitative semantics, assigning large positive values for strong satisfaction and large negative values for significant violation. For simplicity, we denote robustness as $\rho(\xi, \phi)$, omitting the time index.

IV. Methodology

A. Problem formulation

The objective of ILCL is to identify a constraint C_ϕ, represented by a TLTL formula ϕ, from expert trajectories Ξ_E.

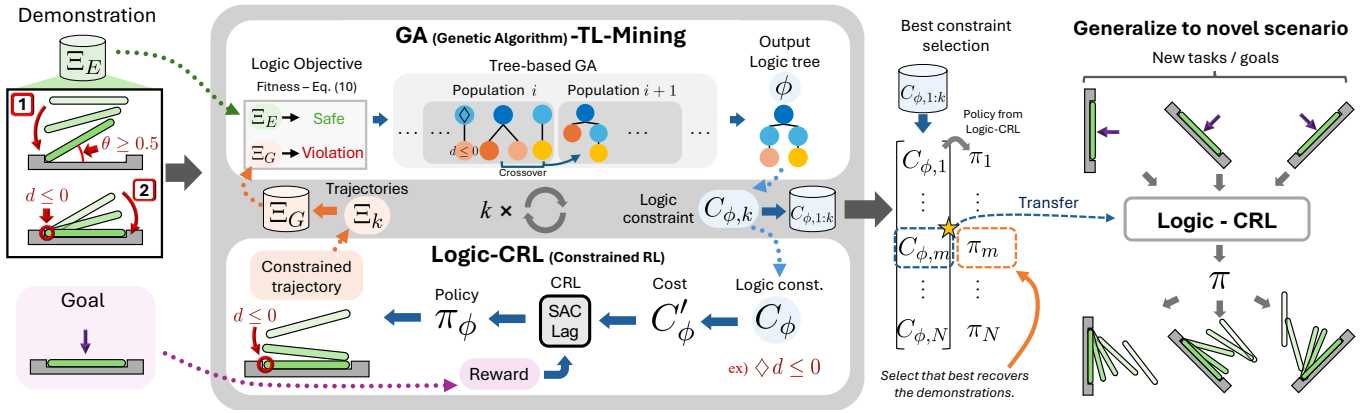


Fig. 2. Illustration of ILCL that learns TLTL constraint from demonstration to generalize to novel scenario. ILCL consists of two parts: **GA-TL-Mining** (Middle top) which generates a TLTL constraint that distinguishes demonstration trajectories from generated trajectories through a genetic algorithm on the TL syntax tree; and **Logic-CRL**, which optimizes the policy for task rewards under the generated TL constraint using SAC-Lag [29]. Once ILCL identifies a TL constraint, Logic-CRL derives a new policy applicable to novel scenarios.

Then, we formulate the identification as a constrained min-max optimization that finds an optimal constraint C^*_ϕ maximizing the expected cumulative rewards of the the inferred constrained policy π, while minimizing its expectation difference from the expert policy π_E demonstrated in Ξ_E:

$$\begin{aligned} C^*_\phi = \arg\min_{C_\phi} \max_{\pi} \quad & \mathbb{E}_{\xi\sim\pi}[R(\xi)] - \mathbb{E}_{\xi\sim\pi_E}[R(\xi)] \\ \text{s.t.} \quad & \mathbb{E}_{\xi\sim\pi}[C_\phi(\xi)] = 0 \\ & \mathbb{E}_{\xi\sim\pi_E}[C_\phi(\xi)] = 0, \end{aligned} \tag{3}$$

where $C_\phi(\xi) = \mathbb{1}[\rho(\xi,\phi) < 0]$. However, this problem is ill-posed and also intractable due to the computation of π for every candidate C_ϕ.

Alternatively, we reformulate Eq. (3) as a two-player zero-sum game, inspired by inverse state-action constraint learning in [6]. This approach is to mitigate degenerate conservative solutions in ICL. Our key distinction from [6] is the use of a logic-based constraint C_ϕ defined over state-action sequences. This formulation alternates between constraint and policy optimization players as shown in Fig. 2. In each k-th iteration, the constraint optimization (i.e., GA-TL-Mining) identifies a constraint $C_{\phi,k}$ that maximally penalizes the previously learned policies $\pi_1,\dots,\pi_k$ relative to the expert policy π_E:

$$\begin{aligned} C_{\phi,k} = \arg\max_{C_\phi} \quad & \sum_{i\leq k} \mathbb{E}_{\xi\sim\pi_i}\left[C_\phi(\xi)\right] \\ \text{s.t.} \quad & \mathbb{E}_{\xi\sim\pi_E}\left[C_\phi(\xi)\right] = 0. \end{aligned} \tag{4}$$

The policy optimization (i.e., Logic-CRL) then finds a new optimal policy π_{k+1} that satisfies the identified constraint $C_{\phi,k}$:

$$\pi_{k+1} = \arg\max_{\pi} \mathbb{E}_{\xi\sim\pi}[R(\xi)] \ \text{ s.t. } \mathbb{E}_{\xi\sim\pi}[C_{\phi,k}(\xi)] = 0. \tag{5}$$

These adversarial alternations progressively generate better-fitting constraints, narrowing the gap to the ground truth. The game's convergence details are available in [6]. The identified constraint then induces a policy that minimally violates the constraint while gaining higher rewards than the expert policy. Below, we describe each player in detail.

B. Constraint player: GA-based temporal-logic mining

At each k-th iteration of ILCL, the constraint player—GA-TL-Mining—identifies a TLTL constraint ϕ that maximizes a fitness score $\mathcal{F}_k(\phi)$ given demonstrations Ξ_E. The score $\mathcal{F}_k(\phi)$, a Monte Carlo approximation of the dual objective in Eq. (4), increases with the constraint ϕ that penalizes violations by prior policies $\pi_{1:k}$, while assigning zero if violated by expert trajectories Ξ_E; We define the fitness score $\mathcal{F}_k(\phi)$ as

$$\frac{\sum_{\xi\in\Xi_{1:k}} \mathbb{1}[\rho(\xi,\phi) < 0]}{|\Xi_{1:k}|} \cdot \Pi_{\xi_E\in\Xi_E} \mathbb{1}[\rho(\xi_E,\phi) > 0], \tag{6}$$

where $\Xi_{1:k} = \{\xi \,|\, \xi \sim \pi_i, i \in [1,k]\}$ and π_i denotes policies generated by Logic-CRL. Using the fitness function, GA-TL-Mining iteratively improves a population of TLTL constraints, represented as logic trees, through selection, crossover, and mutation, following the tree-based GA framework, ROGE [22]. For simplicity, we refer TLTL constraints and logic trees interchangeably.

GA-TL-Mining consists of three steps—1) initial population generation, 2) parent selection, and 3) offspring generation. We describe detail below.

1) Initial population generation: GA-TL-Mining begins with an initial population of logic trees Φ, which consists of N_B basis logic trees $\{\phi^\text{B}_i\}_{i=1}^{N_\text{B}}$ and N_R random logic trees $\{\phi^\text{R}_i\}_{i=1}^{N_\text{R}}$:

$$\Phi = \{\phi^\text{B}_1,\dots,\phi^\text{B}_{N_\text{B}}\} \cup \{\phi^\text{R}_1,\dots,\phi^\text{R}_{N_\text{R}}\}. \tag{7}$$

We construct the basis logic trees over the state space $\mathcal{S} \subseteq \mathbb{R}^\kappa$ where κ is the dimensionality. For each dimension $i \in [1,\kappa]$, we generate two pAPs of the form $a\cdot\mathbf{s}^{(i)} < \theta$ with $a \in \{-1,1\}$, yielding a total of 2κ pAPs. Note that the parameter θ has not yet been determined. We then use the pAPs to compose logic trees in five temporal forms:

$$\bigcirc\mu,\ \Diamond\mu,\ \Box\mu,\ \mu_1\,\mathcal{U}\,\mu_2, \text{ and } \mu_1\mathcal{R}\,\mu_2, \tag{8}$$

where μ, μ_1, and μ_2 denote pAPs. This results in 6κ unary formulas, applying three unary operators to 2κ pAPs, and $8\kappa^2$ binary formulas, applying two binary operators to $4\kappa^2$ pAP pairs. We finally obtain a total of $6\kappa+8\kappa^2$ basis logic trees. In this work, to reduce the construction overhead, we subsample basis logic trees by selecting a subset of state dimensions.

We then construct N_R random logic trees by modifying the basis trees sampled to enhance the diversity of the initial population. For each, we begin by randomly selecting a basis tree and iteratively injecting a randomly chosen operator as the

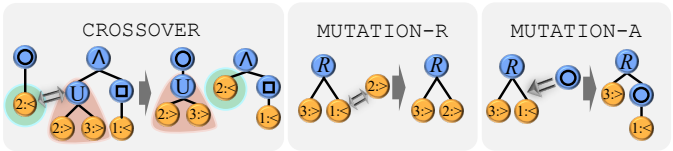


Fig. 3. The illustration of the operations in the offspring generation. For each pTLTL tree in the figure, the blue nodes represent logical operators, and the orange nodes represent pAP nodes. The labels of the pAP node indicate the semantics of pAP; for example, the label '2:<' refers to $\mathbf{s}^{(2)} < \theta$ for an undetermined parameter θ.

parent of a randomly selected node. This process continues until the tree exceeds a predefined depth d_R, or terminates early with probability p_R. In this work, we set $d_R = 3$ and $p_R = 0.1$. Finally, the size of the population becomes $N = N_{\mathrm{B}} + N_{\mathrm{R}}$. Note that, when $k > 1$, the initial population includes the previously discovered $k-1$ TLTL constraints, replacing $k-1$ random trees to guide subsequent mining.

2) Parent selection: This step selects N_p parent logic trees based on the highest regularized fitness scores within the current population $\Phi = \{\phi_1, \ldots, \phi_N\}$ to generate offspring for the next generation. Let $\mathcal{V}(\phi_i)$ denote the node set of each tree ϕ_i. The regularized fitness score $\mathcal{F}_k^{\Phi}(\phi)$ is

$$\mathcal{F}_k^{\Phi}(\phi) = \mathcal{F}_k(\phi) - \zeta \left(\frac{\sum_{i=1}^{|\Phi|} \mathcal{F}_k(\phi_i)}{|\Phi|} \right) \left(|\mathcal{V}(\phi)| - 2 \right)^2, \qquad (9)$$

where ζ is a hyperparameter and the subtraction of 2 accounts for the minimum number of nodes in a tree. To evaluate $\mathcal{F}_k^{\Phi}(\phi)$, we first compute the fitness score $\mathcal{F}_k$ by optimizing the parameters $\boldsymbol{\theta}$ of ϕ using dual annealing , as described in [5]. We then regularize the score by penalizing large tree sizes, encouraging the discovery of simpler constraints in this ill-posed mining problem [22].

3) Offspring generation: We iteratively generate N offspring of the current population $\Phi = \{\phi_1, \ldots, \phi_{N_p}\}$ applying three genetic operators: `CROSSOVER`, `MUTATION-R`, and `MUTATION-A` (see Fig. 3). For each iteration, we first randomly select two parent trees, ϕ_i and ϕ_j, using tournament selection We then randomly apply an genetic operator to generate a new offspring in the following manner:

- `CROSSOVER`: we modify the selected trees, ϕ_i and ϕ_j, by exchanging the randomly selected subtrees with each other.
- `MUTATION-R`: we first randomly select a node in the tree ϕ_i. We then either replace it with another node of the same type—pAP or logic operator—or remove it along with its descendants.
- `MUTATION-A`: we insert a randomly selected logic operator as the parent of a randomly chosen node in the tree ϕ_1 preserving semantics. For example, we mutate $\mu_1 \,\mathcal{U}\, \mu_2$ into $\mu_1 \,\mathcal{U}\, \Box\mu_2$.

Our method alternates parent selection and offspring generation, avoiding duplicates and enforcing temporal consistency by making every non-root node to have an ancestor with a temporal operator. For example, we disallow $\mu_1 \wedge \Box\mu_2$, since μ_1 lacks a temporal operator ancestor. We simplify logic trees using rules such as $\Box\Box = \Box$ to encourage logical simplicity. Mining stops when the current logic accepts all expert demonstrations and rejects all other trajectories. In practice, we limit the number of mining steps to 4 for the *wiping* task and 5 for the other tasks.

C. Policy player: logic-constrained reinforcement learning

We introduce logic-constrained RL (Logic-CRL), which trains a policy to satisfy the TLTL constraints discovered by GA-TL-Mining. A main challenge is that TLTL constraints are non-Markovian; Their truths depend on entire state sequences rather than individual states. To resolve the mismatch, we embed each constraint into a product CMDP (PCMDP) by augmenting states with the TLTL constraint-based automaton states, thereby restoring the Markov property. Another challenge is the non-differentiable nature of logic constraints. To figure it out, we adopt Lagrangian CRL [25] with a novel state-wise constraint redistribution scheme, enabling conventional CRL frameworks to simplify the non-differentiable problem while still satisfying the non-Markovian TLTL constraint. We detail both processes below.

Product Constrained Markov Decision Process: We construct a product CMDP, we call PCMDP, by synchronously combining the CMDP's transition system with a deterministic finite automata (DFA) converted from a TLTL constraint ϕ The DFA is the tuple $(Q_\phi, \Sigma_\phi, \delta_\phi, q_{0,\phi}, F_\phi)$, where Q_ϕ is the set of states, $\Sigma_\phi = 2^{\mathbf{AP}}$ is the power set of APs, $\delta_\phi : Q_\phi \times \Sigma_\phi \to Q_\phi$ is the deterministic transition function, $q_{0,\phi} \in Q_\phi$ is the initial state, and $F_\phi \subset Q_\phi$ is the set of accepting states.

The resulting PCMDP is the tuple, $(\mathcal{S}_\phi, \mathcal{A}, \mathcal{L}_\phi, P_\phi, R, C_\phi)$, where $\mathcal{S}_\phi = \mathcal{S} \times Q_\phi$ is the augmented state space, $\mathcal{L}_\phi : \mathcal{S} \to \Sigma_\phi$ is a labeling function assigning APs to states, and $P_\phi : \mathcal{S}_\phi \times \mathcal{A} \times \mathcal{S}_\phi \to \{0, 1\}$ is the transition function defined as the product of the DFA's transitions and the CMDP's transition probabilities. The transition function deterministically advances the DFA state, starting from $q_{0,\phi}$, by evaluating the label $\mathcal{L}_\phi(\mathbf{s}_{t+1})$ assigned to each next state $\mathbf{s}_{t+1}$. Note that, in this work, we consider PCMDPs with optimal finite-horizon policies that prevent non-Markovian reward-seeking behaviors—such as staying in high-reward states and transitioning to acceptance states only at the last step—by assigning high rewards exclusively to acceptance states.

Finally, PCMDP 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.

Lagrangian-based Constrained Reinforcement Learning: We introduce a Lagrangian CRL resolving the problem of sparse and temporally delayed evaluation of TLTL constraints in policy learning. The 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 but dense constraint-cost function $C'_\phi \in [0, 1]$, updating the discrete function C_ϕ in Eq. 5, as:

$$C'_\phi(\xi) = \alpha C_\phi(\xi) + \frac{1-\alpha}{\Upsilon} \mathrm{CLIP}[-\rho(\xi, \phi), 0, \Upsilon], \qquad (10)$$

where $\alpha \in [0, 1]$ is a mixing weight, $\Upsilon \in \mathbb{R}^+$ limits the impact of large constraint violations, and CLIP truncates the negative robustness value within the range $[0, \Upsilon]$.

To address the delayed evaluation, we introduce a redistribution scheme that spreads out the trajectory-level constraint cost $C'_\phi(\xi)$ uniformly across all state-action pairs $(\mathbf{s}_t, \mathbf{a}_t) \in \xi$,

analogous to the uniform reward redistribution in IRCR [9]. The resulting stepwise constraint cost $C'_\phi(\mathbf{s}_t, \mathbf{q}_t, \mathbf{a}_t)$ is on the augmented state space $(\mathbf{s}_t, \mathbf{q}_t) \in \mathcal{S}_\phi$ as

$$C'_\phi(\mathbf{s}_t, \mathbf{q}_t, \mathbf{a}_t) = \mathbb{E}_{\xi\sim\mathcal{D}}[C'_\phi(\xi)|((\mathbf{s}_t, \mathbf{q}_t), \mathbf{a}_t) \in \xi], \tag{11}$$

where $\mathcal{D}$ is the replay buffer containing previously collected trajectories in the augmented state space. This trajectory-space smoothing converts the trajectory constraint in Eq. (5) into a state-action constraint, enabling its use in Lagrangian CRL.

We finally formulate the Lagrangian dual problem of Eq. (5) that finds a constrained optimal policy π_k in the k-th Logic-CRL process:

$$\arg\min_{\lambda\geq 0}\max_{\pi} \mathop{\mathbb{E}}_{\substack{\xi\sim\pi \\ \mathbf{q}_t=\delta_\phi(\mathbf{q}_{t-1},\mathcal{L}_\phi(\mathbf{s}_t))}} \sum_{\mathbf{s}_t,\mathbf{a}_t\sim\xi} \Big(R(\mathbf{s}_t, \mathbf{a}_t) - \lambda\left(C'_\phi(\mathbf{s}_t, \mathbf{q}_t, \mathbf{a}_t) - d\right)\Big), \tag{12}$$

where λ is a learnable Lagrangian multiplier and d is a constraint-cost threshold. We define a large threshold, $d = \frac{\varepsilon\alpha}{N_\xi}$, for stable optimization while enforcing the satisfaction of the binary logic constraint C_ϕ, where $\varepsilon < 1$ is a positive constant and N_ξ is the number of trajectories sampled from the current policy π. To solve Eq. (12), we adopt SAC-Lag [29]—a soft actor-critic (SAC) [32] version of a Lagrangian CRL method.

Further, to enhance off-policy learning in Eq. (12), we initialize the replay buffer $\mathcal{D}$ with the expert trajectories Ξ_E and the previously sampled trajectories $\Xi_{1:k-1}$ that already satisfy the given constraint After obtaining π_k, we sample zero-violation trajectories Ξ_k and update the trajectory set as $\Xi_{1:k} = \Xi_{1:k-1} \cup \Xi_k$, as input to the next GA-TL-Mining step.

Lastly, after N alternations of GA-TL-Mining and Logic-CRL, ILCL returns the best constraint C^*_ϕ. Although the two-player game ideally converges, the limited iterations and adversarial instability often yield a suboptimal constraint. Thus, we choose the most expert-like constraint $C_{\phi,i}$ from the obtained constraint set $\{C_{\phi,1}, \ldots, C_{\phi,N}\}$. For each candidate constraint, we evaluate its associated policy π_i on the expert trajectories Ξ_E and select the policy that minimizes the error to the expert actions,

$$\pi_i = \mathop{\arg\min}_{\pi_i\in[\pi_1,\ldots,\pi_N]} \mathop{\mathbb{E}}_{\substack{(s_E,a_E)\sim\Xi_E \\ a\sim\pi_i(\cdot|s_E)}} ||a_E - a||_2. \tag{13}$$

We then report $C_{\phi,i}$ corresponding to π_i.

V. Experimental Setup

A. Benchmark Setups

1) *Navigation*: This task requires a point agent at $\mathbf{x}^{Agt} \in \mathbb{R}^2$ navigating to a goal position $\mathbf{x}^{Goal} \in \mathbb{R}^2$ while satisfying temporal constraints related to colored regions randomly placed in the workspace (see Fig. 4 (Left)). Let $\mathbf{p}^R$, $\mathbf{p}^G$, and $\mathbf{p}^B$ denote the polar coordinates $(p_{dist}, p_{ang}) \in \mathbb{R}^2$ of the nearest red, green, and blue regions, respectively, measured relative to the agent's coordinate. For example, $\mathbf{p}^R = (p^R_{dist}, p^R_{ang})$. We model this environment as two CMDPs, each subject to a distinct constraint, ϕ_{GT1} or ϕ_{GT2}:

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = (\mathbf{x}^{Agt}, \mathbf{x}^{Goal}, \mathbf{p}^R, \mathbf{p}^G, \mathbf{p}^B) \in \mathbb{R}^{10}\}$.
- $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} = \Delta\mathbf{x}^{Agt} \in \mathbb{R}^2\}$, the space of velocity commands.
- $R = 1 - \tanh(5 \cdot \|\mathbf{x}^{Agt} - \mathbf{x}^{Goal}\|_2)$
- $C = C_{\phi_{\text{GT1}}}$ or $C_{\phi_{\text{GT2}}}$; ϕ_{GT1} requires the agent to always avoid red and avoid blue until reaching green. ϕ_{GT2} requires the agent to visit red, green, and blue in that order.

We train each method on 20 demonstrations ($T = 25$), each constrained by ϕ_{GT1} and ϕ_{GT2} (see Table I). Then, we evaluate the learned constraints on test environments with varying number, size, and placement of regions. Each test uses a 1.5 times larger workspace and a time horizon of $T = 50$.

2) *Wiping*: This task requires 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), as illustrated in Fig. 4 (Middle). Let f_c be the contact force between the roller and the surface. ${}^A\mathbf{x}^{Agt}$ and ${}^B\mathbf{x}^{Agt}$ denote the positions of the agent relative to the start and goal points, respectively. For example, ${}^A\mathbf{x}^{Agt} = ({}^Ax^{Agt}, {}^Ay^{Agt})$. We model this environment as a CMDP:

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = ({}^A\mathbf{x}^{Agt}, {}^B\mathbf{x}^{Agt}, \dot{\mathbf{x}}^{Agt}, w^{Agt}, d_g, f_g) \in \mathbb{R}^9\}$, 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 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, $R_{contact} = 2 \cdot \mathbb{1}(f_c > 0.05)$ is a contact reward to encourage wiping contact, and $R_{penalty} = -0.2 \cdot f_c^2$ is to penalize large contact.
- $C = C_{\phi_{\text{GT}}}$ where the agent requires to keep sufficient contact force while the roller is between points A and B, while avoiding contact otherwise.

We train each method on 100 demonstrations ($T = 50$), with randomized start, goal, and agent positions under constraint ϕ_{GT}, and evaluate on test environments with sinusoidal surfaces of varying frequencies.

3) *Peg-in-shallow-hole*: This task requires a gripper agent to insert a thin peg into a shallow-angled hole maintaining its side contact as shown in Fig. 4 (Right). Let Σ_h denote the 2D hole frame, tilted by angle θ^h. The task uses a parallel-jaw gripper positioned at ${}^{\Sigma_h}\mathbf{x}^{grip}$, represented by $({}^{\Sigma_h}x^{grip}, {}^{\Sigma_h}y^{grip}, {}^{\Sigma_h}\theta^{grip}) \in \mathbb{R}^3$, where θ^{grip} represents the tilt angle of the gripper in the hole frame. Likewise, a peg is at ${}^{\Sigma_h}\mathbf{x}^{peg}$. In addition, 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 model this environment as a CMDP:

- $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} = ({}^{\Sigma_h}\mathbf{x}^{grip}, {}^{\Sigma_h}\dot{\mathbf{x}}^{grip}, {}^{\Sigma_h}\mathbf{x}^{peg}, {}^{\Sigma_h}\dot{\mathbf{x}}^{peg}, {}^{\Sigma_h}\bar{\mathbf{x}}^{grip}, d^{jaw}, \dot{d}^{jaw}, \bar{d}^{jaw}, d^{peg}_{hole}, d^{peg}_{jaw}, \cos\theta^h, \sin\theta^h) \in \mathbb{R}^{22}\}$, 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.
- $\mathcal{A} = \{\mathbf{a} \mid \mathbf{a} = (\Delta^{\Sigma_h}\bar{\mathbf{x}}^{grip}, \Delta\bar{d}^{finger}) \in \mathbb{R}^4\}$; A space of discrete-time velocity commands for the gripper and fingers.
- $R = (1 - \tanh(5|{}^{\Sigma_h}y^{peg} - {}^{\Sigma_h}y^{peg*}|)) + 5 \cdot \mathbb{1}({}^{\Sigma_h}y^{peg} - {}^{\Sigma_h}y^{peg*})$, where ${}^{\Sigma_h}y^{peg}$ and ${}^{\Sigma_h}y^{peg*}$ are the current and desired vertical position of the peg in terms of the hole frame Σ_h.
- $C = C_{\phi_{\text{GT}}}$ where the agent requires maintaining peg-to-hole contact once tilting the peg above a certain angle, while keeping the peg-to-jaw contact until full insertion.

We train each method on 100 demonstrations ($T = 30$), with randomized start, goal, agent poses as well as randomly tilted holes ($\theta^h \in [-10°, 10°]$), under the constraint ϕ_{GT}. Then, we

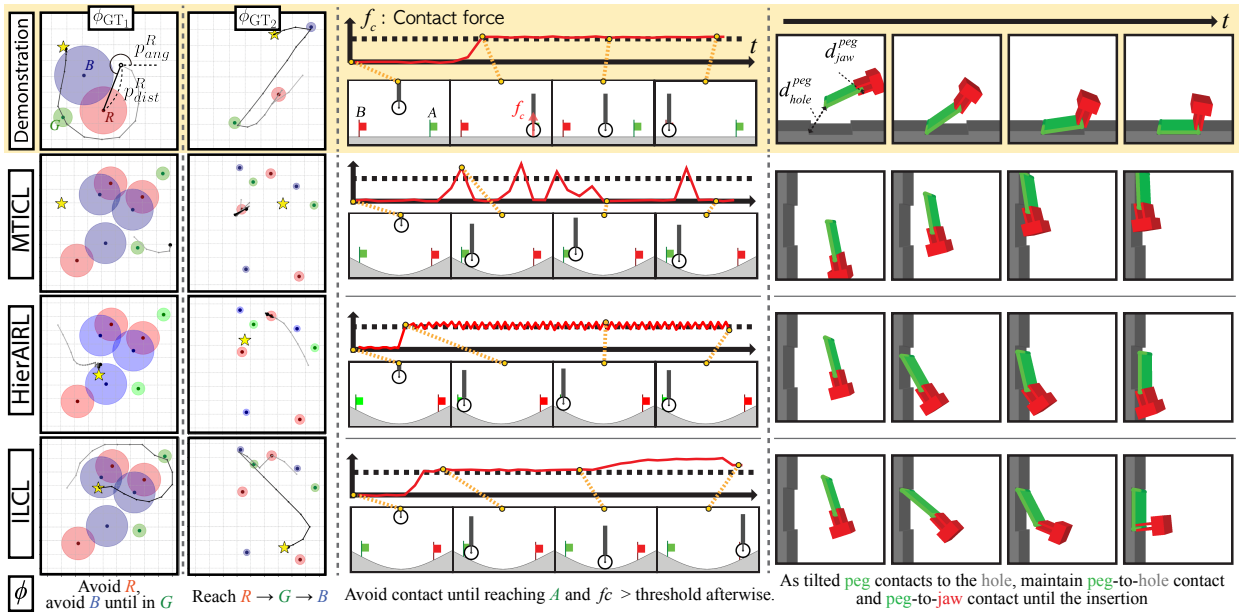


Fig. 4. Comparison of temporal constraint learning and transfer performance in four simulated tasks. In each task, ILCL, MTICL, and HierAIRL first learn temporal constraints from demonstrations (top), satisfying the ground-truth constraint ϕ (bottom). In novel environments, we train policies with the learned constraints or rewards to reproduce demonstration-like constrained behaviors. **Left**: in the *navigation* tasks, yellow shapes and black traces represent goals 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 gripper, a peg, and a hole, respectively.

evaluate on test environments with tilt angles $\theta_h \in \{-90°. -45°, 45°, 90°\}$ increasing the time horizon into $T = 60$.

B. Quantitative Evaluation Study with Baselines

We evaluate each methodology in both training (seen) and testing (unseen) environments. For training, we estimate a constraint and the corresponding constrained policy for each random seed. From each policy, we generate 10^3 trajectories by randomizing initial conditions. In total, we produce 10^4, $5 \cdot 10^3$, and $5 \cdot 10^3$ trajectories using 10, 5, and 5 random seeds, respectively, for the three tasks presented.

For testing, we transfer the learned constraints (i.e., 10, 5, and 5 constraints for each task, respectively) to novel, randomly generated environments. Creating 10, 4, and 4 new environment for the three tasks, we train a constrained policy per environment. We generate 10^3 trajectories varying the start or goal conditions. In total, we produce 10^5, $2 \cdot 10^4$, and $2 \cdot 10^4$ trajectories. We then assess performance using the following metrics across all trajectories:

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

We compare ILCL against ICL and inverse reinforcement learning (IRL) methods. The ICL methods include numeric state-action (i.e., Markovian) constraint learning approaches:

- **ICRL** [11]: An inverse CRL (ICRL) method with maximum likelihood constraint inference.
- **TCL** [12]: A transferable constraint learning (TCL) method by reward decomposition.
- **MTICL** [6]: 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 with multiple level temporal abstraction:

- **Option-GAIL** [33]: An HIRL extension of generative adversarial imitation learning (GAIL).
- **Hier-AIRL** [34]: An HIRL extension of adversarial inverse reinforcement learning (AIRL).

For evaluations, we provide all methods with identical features related to the ground-truth constraints. However, we give IRL baselines additional reward-related features to facilitate learning their reward functions.

C. Qualitative Transfer Study in Real World

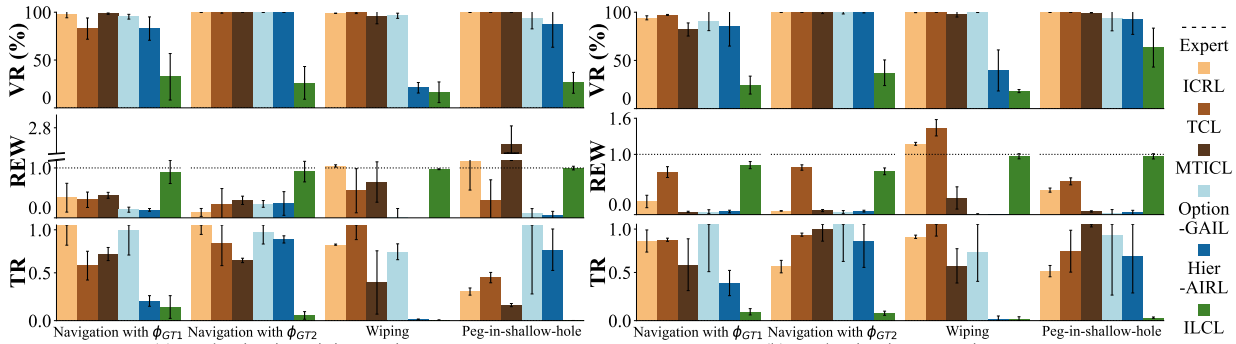
We perform qualitative transfer studies in real-world *peg-in-shallow-hole* environments, as shown in Fig. 1. The environment consists of a 7-DoF Franka Emika Panda arm, an external RGB-D camera, a peg, and a tilt of shallow hole. After learning the stable-contact constraint through simulations, we enable the robot to reproduce demonstration-like constrained behaviors via Logic-CRL in novel environments.

To show the transferability, we vary the hole's slope within $\{-25°, 60°, 90°\}$. To detect slope and peg states, we use AprilTags on the gripper, hole, and peg. The learned policy applies discrete-time velocity commands, sending desired poses to a Cartesian PD controller at 1 kHz and RL controller at 5 Hz.

VI. Evaluation Results

A. Quantitative Analysis through Simulations

We evaluate the constraint learning performance of ILCL against baselines across four benchmark environments with distinct ground-truth logic constraints. As shown in Fig. 5 (a), ILCL consistently outperforms baselines in learning constraints, achieving the lowest VRs in all training environments while obtaining expert-like REWs. We normalize all REW values to the expert demonstrations' scores, ranging



(a) Evaluation in training environments. (b) Evaluation in test environments.

Fig. 5. Comparison of the proposed ILCL and five baseline methods in training and test environments. 'Expert' denotes the result of expert demonstrations without constraint violations. We normalize all REW values to the 'Expert' score, resulting in a range of $[0, 1]$.

Table I. Examples of ground-truth (GT) and TLTL constraints estimated by ILCL. V, R, and T denote VR, REW, and TR, respectively. The reported statistics are the evaluation results of fully constrained trajectories generated by Logic-CRL, using the same REW and TR scales as in Fig. 5.

Env.	Algo.	TLTL Const.	V (%)	R	T
Nav. ϕ_{GT1}	GT	$\square p^R_{\text{dist}} > 0.2 \wedge (p^B_{\text{dist}} > 0.25 \,\mathcal{U}\, p^G_{\text{dist}} < 0.08)$	0	1	0
	ILCL	$(\square p^R_{\text{dist}} > 0.205 \wedge p^B_{\text{dist}} > 0.267) \mathcal{U} p^G_{\text{dist}} < 0.081$	1.3	0.83	2.6E-5
Nav. ϕ_{GT2}	GT	$\Diamond(p^R_{\text{dist}} < 0.06 \wedge \Diamond(p^G_{\text{dist}} < 0.05 \wedge \Diamond(p^B_{\text{dist}} < 0.04)))$	0	1	0
	ILCL	$(\Diamond p^B_{\text{dist}} < 0.041 \,\mathcal{U}\, p^G_{\text{dist}} < 0.049) \,\mathcal{U}\, p^R_{\text{dist}} < 0.061$	18.5	0.69	3.5E-2
Wiping	GT	$f_c < 0.05 \mathcal{U} (\square f_c > 1.6 \,\mathcal{R}$ $({}^A x^{Agt} > -0.1 \wedge {}^A x^{Agt} < 0.1))$	0	1	0
	ILCL	$\square f_c > 1.661 \,\mathcal{R}\, (f_c < 0.062 \,\mathcal{U}$ $({}^A x^{Agt} < 0.099 \wedge {}^A x^{Agt} > -0.076))$	**0**	0.97	**0**
Peg-in-shallow-hole	GT	$\Diamond({}^{\Sigma_h}\theta^{peg} > 34.88° \wedge ({}^{\Sigma_h}\theta^{peg} < 4.217°$ $\mathcal{R}\, d^{peg}_{jaw} < 0.0053) \wedge \square d^{peg}_{hole} < 0.0072)$	0	1	0
	ILCL	$\Diamond({}^{\Sigma_h}\theta^{\text{peg}} > 34.68° \wedge {}^{\Sigma_h}\theta^{\text{peg}} < 39.44° \wedge ({}^{\Sigma_h}\theta^{\text{peg}}$ $< 0.241° \,\mathcal{R}\, d^{\text{peg}}_{\text{jaw}} < 0.0055) \wedge \square d^{\text{peg}}_{\text{hole}} < 0.0072)$	11.2	0.95	2.1E-4

from $[0, 1]$. Notably, in the *navigation* scenario with ϕ_{GT1}, ILCL's maximum violation rate of 32.5% is 50.4% lower than Hier-AIRL's, the best performing baseline. In contrast, ICL baselines exhibit significantly higher VRs due to their inability to learn temporal structure. Although IRL baselines learn the hierarchical structure in demonstrations, the approaches frequently fail to capture either the task objective or the embedded constraint. For example, Hier-AIRL in the *wiping* task achieves a VR of 21%, lower than other baselines, but still yields low REWs due to a deficient task reward signal. In contrast, REWs exceeding 1.0 indicate less- or un-constrained goal-seeking behaviors due to failures in constraint learning or CRL. Note that the binary nature of VRs classifies even minor, single timestep violations as full binary violations. A continuous measure is necessary to provide a more detailed assessment of constraint quality.

Alternatively, to provide a more detailed assessment beyond the binary measure of violations, we investigate the degree of violations using the TR metric. ILCL consistently demonstrates the lowest mean TR values, significantly outperforming all baselines. These results underscore the complexity of temporal-constraint learning, even in seen environments, for conventional ICL or IRL algorithms. Although the ICRL and MTICL approaches yield high REWs in the last two tasks, these rewards are from unconstrained behaviors, resulting in significantly high violation rates, which are undesirable in robotic tasks. In addition, IRL shows the lowest REWs across most tasks, reflecting the difficulty of simultaneously optimizing both task rewards and constraints.

We then assess the transferability of the learned constraints to the novel environments as shown in Fig. 5 (b). ILCL consistently achieves the lowest VR and TR values with high REWs, similar to its performance during training. In contrast, all baselines exhibit VRs exceeding 85.9%, with the sole exception of Hier-AIRL in the *wiping* task, highlighting the significant challenges in transferring temporal constraints. Although Hier-AIRL yields a modest VR in *wiping*, it frequently gets stuck at the start point as illustrated in Fig. 4 and fails to collect task rewards in test environments. Further, in the *peg-in-shallow-hole* task, ILCL shows a relatively high VR of 63.3% despite a significantly low TR value. This discrepancy arises not only from the binary nature of VR but also from the Lagrangian approximation used in Logic-CRL, which softens constraints to prioritize high-reward behaviors, increasing violations in novel settings.

Table I shows examples of ground-truth and estimated constraints as well as the exemplar performance of their corresponding constrained policies per test environment. The examples show that ILCL returns a constraint similar to the ground truth. Although the estimated constraints exhibit syntactic variations from the ground truth, they consistently achieve significantly low VRs, averaging $7.8 \pm 8.7\%$, and high REWs, averaging 0.86 ± 0.13, similar to ground-truth performance across all tasks. To ensure accurate comparison, we evaluate $1,000$ trajectories that do not violate the target constraints used in Logic-CRL, thereby reducing its potential negative impact on this analysis.

B. Qualitative Demonstrations in Real World

Fig. 6 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 90°, 60°, and −25°. During the insertions, the gripper maintained stable contact between the hole and the peg. These demonstrations indicate that ILCL learns abstract logic constraints that are robustly transferable to real-world setups. For each experiment, we optimized the constrained policy for 4 hours within $10,000$ steps of real-world interactions.

VII. Conclusion

We introduced an inverse logic-constraint learning (ILCL) algorithm that identifies free-form transferable TLTL constraint from demonstrations. Adopting a two-player zero-sum game, ILCL combines GA-TL-Mining—learning TLTL

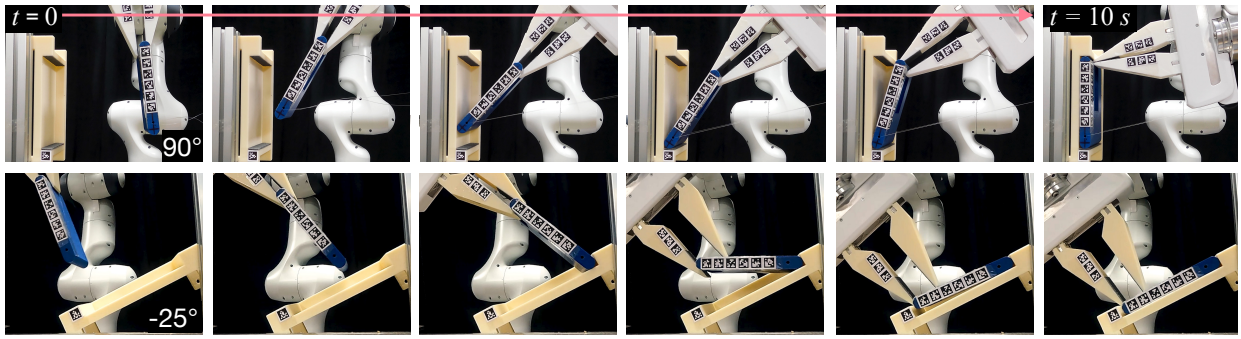


Fig. 6. Demonstration of ILCL's constraint transfer capability in the *peg-in-shallow-hole* environments. By creating real-world environments with holes tilted at 90°, 60°, and −25°, we enable the Panda gripper to find a constrained policy using the learned constraints and reproduce a demonstration-like robust insertion with stable contact in each unseen environment. Demonstration videos are available in the supplementary materials including 60° tilted hole insertion.

constraints from demonstrations without predefined templates—and Logic-CRL—optimizing policies under non-Markovian constraints. In quantitative and qualitative studies, ILCL outperforms state-of-the-art baselines, reproducing demonstration-like behaviors with fewer violations in novel environments, and generalizes to real-world peg-in-shallow-hole tasks.

REFERENCES

[1] G. Liu, S. Xu, S. Liu, A. Gaurav, S. G. Subramanian, and P. Poupart, "A comprehensive survey on inverse constrained reinforcement learning: Definitions, progress and challenges," *Trans. on Machine Learning Research*, 2025.

[2] A. Pnueli, "The temporal logic of programs," in *Annual Symp. on Foundations of Computer Science*, 1977.

[3] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Int'l. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, 2004.

[4] L. Yifru and A. Baheri, "Concurrent learning of control policy and unknown safety specifications in reinforcement learning," *IEEE Open J. of Control Systems*, 2024.

[5] W. Liu, D. Li, E. Aasi, R. Tron, and C. Belta, "Interpretable generative adversarial imitation learning," *arXiv preprint arXiv:2402.10310*, 2024.

[6] K. Kim, G. Swamy, Z. Liu, D. Zhao, S. Choudhury, and Z. S. Wu, "Learning shared safety constraints from multi-task demonstrations," in *Proc. Int'l Conf. on Neural Information Processing Systems*, 2023.

[7] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. RSJ Int'l Conf. on Intelligent Robots and Systems*, 2017.

[8] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter, "RUDDER: return decomposition for delayed rewards," in *Proc. Int'l Conf. on Neural Information Processing Systems*, 2019.

[9] T. Gangwani, Y. Zhou, and J. Peng, "Learning guidance rewards with trajectory-space smoothing," in *Proc. Int'l Conf. on Neural Information Processing Systems*, 2020.

[10] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *Proc. Int'l. Conf. on Learning Representations*, 2020.

[11] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *Proc. Int'l Conf. on Machine Learning*, 2021.

[12] J. Jang, M. Song, and D. Park, "Inverse constraint learning and generalization by transferable reward decomposition," *IEEE Robotics and Automation Letters*, 2023.

[13] G. Qiao, G. Liu, P. Poupart, and Z. Xu, "Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations," in *Proc. Int'l Conf. on Neural Information Processing Systems*, 2023.

[14] M. Baert, S. Leroux, and P. Simoens, "Learning logic constraints from demonstration," in *Proc. Int'l Workshop on Neural-Symbolic Learning and Reasoning*, 2023.

[15] M. Baert, S. Leroux, and P. Simoens, "Learning safety constraints from demonstration using one-class decision trees," in *AAAI Workshop on Neuro-Symbolic Learning and Reasoning in the Era of Large Language Models*, 2024.

[16] R. Kusters, Y. Kim, M. Collery, C. de Sainte Marie, and S. Gupta, "Differentiable rule induction with learned relational features," in *Proc. Int'l Workshop on Neural-Symbolic Learning and Reasoning*, 2022.

[17] E. Bartocci, C. Mateis, E. Nesterini, and D. Nickovic, "Survey on mining signal temporal logic specifications," *Information and Computation*, 2022.

[18] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "Telex: Passive stl learning using only positive examples," in *Proc. Int'l Conf. on Runtime Verification*, 2017.

[19] D. Li, M. Cai, C.-I. Vasile, and R. Tron, "Learning signal temporal logic through neural network for interpretable classification," in *Proc. American Control Conf.*, 2023.

[20] P. Vaidyanathan, R. Ivison, G. Bombara, N. A. DeLateur, R. Weiss, D. Densmore, and C. Belta, "Grid-based temporal logic inference," in *Proc. IEEE Conf. on Decision and Control*, 2017.

[21] S. Mohammadinejad, J. V. Deshmukh, A. G. Puranic, M. Vazquez-Chanlatte, and A. Donzé, "Interpretable classification of time-series data using efficient enumerative techniques," in *Proc. of the Int'l. Conf. on Hybrid Systems: Computation and Control*, 2020.

[22] L. Nenzi, S. Silvetti, E. Bartocci, and L. Bortolussi, "A robust genetic algorithm for learning temporal specifications from data," in *Proc. Int'l Conf. on Quantitative Evaluation of Systems*, 2018.

[23] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. of Machine Learning Research*, 2015.

[24] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int'l Conf. on Machine Learning*, 2017.

[25] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *Proc. Int'l. Conf. on Learning Representations*, 2019.

[26] A. Shah, C. Voloshin, C. Yang, A. Verma, S. Chaudhuri, and S. A. Seshia, "LTL-constrained policy optimization with cycle experience replay," *Trans. on Machine Learning Research*, 2025.

[27] M. Wen, R. Ehlers, and U. Topcu, "Correct-by-synthesis reinforcement learning with temporal logic constraints," in *Proc. RSJ Int'l Conf. on Intelligent Robots and Systems*, 2015.

[28] J. Jiang, S. Coogan, and Y. Zhao, "A unified approach to multi-task legged navigation: Temporal logic meets reinforcement learning," *arXiv preprint arXiv:2407.06931*, 2024.

[29] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," in *Proc. Conf. on Robot Learning*, 2021.

[30] G. De Giacomo, M. Y. Vardi, *et al.*, "Linear temporal logic and linear dynamic logic on finite traces." in *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2013.

[31] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Proc. Int'l Conf. on Runtime Verification*, 2012.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int'l Conf. on Machine Learning*, 2018.

[33] M. Jing, W. Huang, F. Sun, X. Ma, T. Kong, C. Gan, and L. Li, "Adversarial option-aware hierarchical imitation learning," in *Proc. Int'l Conf. on Machine Learning*, 2021.

[34] J. Chen, T. Lan, and V. Aggarwal, "Option-aware adversarial inverse reinforcement learning for robotic control," in *Proc. Int'l Conf. on Robotics and Automation*, 2023.