Explaining Multi-stage Tasks by Learning Temporal Logic Formulas from Suboptimal Demonstrations

Glen Chou, Necmiye Ozay, and Dmitry Berenson
Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109
Email: {gchou, necmiye, dmitryb}@umich.edu

***Abstract*—We present a method for learning multi-stage tasks from demonstrations by learning the logical structure and atomic propositions of a consistent linear temporal logic (LTL) formula. The learner is given successful but potentially suboptimal demonstrations, where the demonstrator is optimizing a cost function while satisfying the LTL formula, and the cost function is uncertain to the learner. Our algorithm uses the Karush-Kuhn-Tucker (KKT) optimality conditions of the demonstrations together with a counterexample-guided falsification strategy to learn the atomic proposition parameters and logical structure of the LTL formula, respectively. We provide theoretical guarantees on the conservativeness of the recovered atomic proposition sets, as well as completeness in the search for finding an LTL formula consistent with the demonstrations. We evaluate our method on high-dimensional nonlinear systems by learning LTL formulas explaining multi-stage tasks on 7-DOF arm and quadrotor systems and show that it outperforms competing methods for learning LTL formulas from positive examples.**

I. Introduction

Imagine demonstrating a multi-stage task to a robot arm barista, such as preparing a drink for a customer (Fig. 1). How should the robot understand and generalize the demonstration? One popular method is inverse reinforcement learning (IRL), which assumes a level of optimality on the demonstrations, and aims to learn a reward function that replicates the demonstrator's behavior when optimized [1, 4, 31, 35]. Due to this representation, IRL works well on short-horizon tasks, but can struggle to scale to multi-stage, constrained tasks [14, 27, 39]. Transferring reward functions across environments (i.e. from one kitchen to another) can also be difficult, as IRL may overfit to aspects of the training environment. It may instead be fruitful to decouple the high- and low-level task structure, learning a logical/temporal abstraction of the task that is valid for different environments which can combine low-level, environment-dependent skills. Linear temporal logic (LTL) is well-suited for representing this abstraction, since it can unambiguously specify high-level temporally-extended constraints [5] as a function of atomic propositions (APs), which can be used to describe salient low-level state-space regions. To this end, a growing community in controls and anomaly detection has focused on learning linear temporal logic (LTL) formulas to explain trajectory data. However, the vast majority of these methods require both positive and negative examples in order to regularize the learning problem. While this is acceptable in anomaly detection, where one expects to observe formula-violating trajectories, in the context of robotics, it can be unsafe to ask a demonstrator to execute formula-violating behavior, such as spilling the drink or crashing into obstacles.

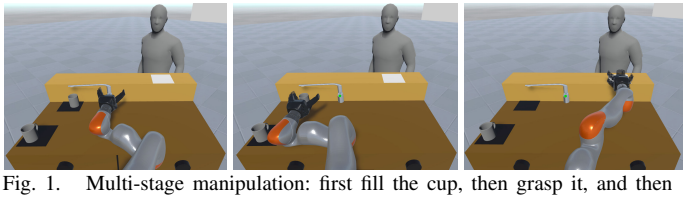

Fig. 1. Multi-stage manipulation: first fill the cup, then grasp it, and then deliver it. To avoid spills, a pose constraint is enforced after the cup is grasped.

In this paper, our insight is that by assuming that demonstrators are goal-directed (i.e. approximately optimize an objective function that may be uncertain to the learner), we can regularize the LTL learning problem without being given formula-violating behavior. In particular, we learn LTL formulas which are parameterized by their high-level logical structure and low-level AP regions, and we show that to do so, it is important to consider demonstration optimality both in terms of the quality of the discrete high-level logical decisions and the continuous low-level control actions. We use the Karush-Kuhn-Tucker (KKT) optimality conditions from continuous optimization to learn the shape of the low-level APs, along with notions of discrete optimality to learn the high-level task structure. We solve a mixed integer linear program (MILP) to jointly recover LTL and cost function parameters which are consistent with the demonstrations. We make the following contributions:

1) We develop a method for time-varying, constrained inverse optimal control, where the demonstrator optimizes a cost function while respecting an LTL formula, where the parameters of the atomic propositions, formula structure, and an uncertain cost function are to be learned. We require only positive demonstrations, can handle demonstration suboptimality, and for fixed formula structure, can extract guaranteed conservative estimates of the AP regions.
2) We develop conditions on demonstrator optimality needed to learn high- and low-level task structure: AP regions can be learned with discrete feasibility, while logical structure requires various levels of discrete optimality. We develop variants of our method under these different assumptions.
3) We provide theoretical analysis of our method, showing that under mild assumptions, it is guaranteed to return the shortest LTL formula which is consistent with the demonstrations, if one exists. We also prove various results on our method's conservativeness and on formula learnability.
4) We evaluate our method on learning complex LTL formulas demonstrated on nonlinear, high-dimensional systems, show that we can use demonstrations of the same task on different environments to learn shared high-level task structure, and show that we outperform previous approaches.

II. Related Work

There is extensive literature on inferring temporal logic formulas from data via decision trees [9], genetic algorithms [11], and Bayesian inference [37, 39]. However, most of these methods require positive and negative examples as input [13, 25, 26, 30], while our method is designed to only use positive examples. Other methods require a space-discretization [3, 38, 39], while our approach learns LTL formulas in the original continuous space. Some methods learn AP parameters, but do not learn logical structure or perform an incomplete search, relying on formula templates [6, 28, 41], while other methods learn structure but not AP parameters [37]. Perhaps the method most similar to ours is [22], which learns parametric signal temporal logic (pSTL) formulas from positive examples by fitting formulas that the data tightly satisfies. However, the search over logical structure in [22] is incomplete, and tightness may not be the most informative metric given goal-directed demonstrations (c.f. Sec. VIII). To our knowledge, this is the first method for learning LTL formula structure and parameters in continuous spaces on high-dimensional systems from only positive examples.

IRL [1, 18, 23, 24, 35] searches for a reward function that replicates a demonstrator's behavior when optimized, but these methods can struggle to represent multi-stage, long-horizon tasks [27]. To alleviate this, [27, 34] learn sequences of reward functions, but in contrast to temporal logic, these methods are restricted to learning tasks which can be described by a single fixed sequence. Temporal logic generalizes this, being able to represent tasks that involve more choices and can be completed with multiple different sequences. Some work [33, 42] aims to learn a reward function given that the demonstrator satisfies a known temporal logic formula; we will learn both jointly.

Finally, there is relevant work in constraint learning. These methods generally focus on learning time-invariant constraints [12, 14, 15, 16] or a fixed sequence of task constraints [32], which our method subsumes by learning time-dependent constraints that can be satisfied by different sequences.

III. Preliminaries and Problem Statement

We consider discrete-time nonlinear systems $x_{t+1} = f(x_t, u_t, t)$, with state $x \in \mathcal{X}$ and control $u \in \mathcal{U}$, where we denote state/control trajectories of the system as $\xi_{xu} \doteq (\xi_x, \xi_u)$.

We use linear temporal logic (LTL) [5], which augments standard propositional logic to express properties holding on trajectories over (potentially infinite) periods of time. In this paper, we will be given finite-length trajectories demonstrating tasks that can be completed in finite time. To ensure that the formulas we learn can be evaluated on finite trajectories, we focus on learning formulas, given in positive normal form, which are described in a parametric temporal logic similar to bounded LTL [20], and which can be written with the grammar

$$\varphi ::= p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \Box_{[t_1,t_2]}\varphi \mid \varphi_1 \,\mathcal{U}_{[t_1,t_2]}\, \varphi_2, \quad (1)$$

where $p \in \mathcal{P} \doteq \{p_i\}_{i=1}^{N_{\text{AP}}}$ are atomic propositions (APs) and N_{AP} is known to the learner. $t_1 \leq t_2$ are nonnegative integers. Here, $\neg\varphi$ denotes the negation of formula φ, $\varphi_1 \vee \varphi_2$ denotes the disjunction of formulas φ_1 and φ_2, $\varphi_1 \wedge \varphi_2$ denotes the conjunction of formulas φ_1 and φ_2, the "always" operator $\Box_{[t_1,t_2]}\varphi$ denotes that φ "always" has to hold, and the "until" operator $\varphi_1 \,\mathcal{U}_{[t_1,t_2]}\, \varphi_2$ denotes that φ_2 must eventually hold, and φ_1 must hold up until that time. The semantics, describing satisfaction of an LTL formula φ by a trajectory ξ_{xu}, denoted $\varphi \models \xi_{xu}$, are given in App. A. Note that negation only appears directly before APs. Let the size of the grammar be $N_{\text{g}} = N_{\text{AP}} + N_{\text{o}}$, where N_{o} is the number of temporal/boolean operators in the grammar. A useful derived operator is "eventually" $\Diamond_{[t_1,t_2]}\varphi \doteq \top \,\mathcal{U}_{[t_1,t_2]}\, \varphi$. We consider tasks that involve optimizing a parametric cost function (encoding efficiency concerns, etc.), while satisfying an LTL formula $\varphi(\theta^s, \theta^p)$ (encoding constraints for task completion):

Problem 1 (Forward problem):

$$\begin{aligned} &\underset{\xi_{xu}}{\text{minimize}} && c(\xi_{xu}, \theta^c) \\ &\text{subject to} && \xi_{xu} \models \varphi(\theta^s, \theta^p) \\ & && \bar{\eta}(\xi_{xu}) \in \bar{\mathcal{S}} \subseteq \mathcal{C} \end{aligned}$$

where $c(\cdot, \theta^c)$ is a potentially non-convex cost function, parameterized by $\theta^c \in \Theta^c$. The LTL formula $\varphi(\theta^s, \theta^p)$ is parameterized by $\theta^s \in \Theta^s$, encoding the logical and temporal structure of the formula, and by $\theta^p \doteq \{\theta_i^p\}_{i=1}^{N_{\text{AP}}}$, where $\theta_i^p \in \Theta_i^p$ defines the shape of the region where p_i holds. Specifically, we consider APs of the form: $x \models p_i \Leftrightarrow \mathbf{g}_i(\eta_i(x), \theta_i^p) \leq \mathbf{0}$, where $\eta_i(\cdot) : \mathcal{X} \to \mathcal{C}$ is a known nonlinear function, $\mathbf{g}_i(\cdot,\cdot) \doteq [g_{i,1}(\cdot,\cdot), \ldots, g_{i,N_i^{\text{ineq}}}(\cdot,\cdot)]^\top$ is a vector-valued parametric function, and $\mathcal{C}$ is the space in which the constraint is evaluated, elements of which are denoted *constraint states* $\kappa \in \mathcal{C}$. In the manipulation example, the joint angles are x, the end effector pose is κ, and $\eta(\cdot)$ are the forward kinematics. As shorthand, let $G_i(\kappa, \theta_i^p) \doteq \max_{m \in \{1,\ldots,N_i^{\text{ineq}}\}} \big(g_{i,m}(\kappa, \theta_i^p)\big)$. Define the subset of $\mathcal{C}$ where p_i holds/does not hold, as

$$\mathcal{S}_i(\theta_i^p) \doteq \{\kappa \mid G_i(\kappa, \theta_i^p) \leq 0\} \quad (2)$$

$$\mathcal{A}_i(\theta_i^p) \doteq \text{cl}(\{\kappa \mid G_i(\kappa, \theta_i^p) > 0\}) = \text{cl}(\mathcal{S}_i(\theta_i^p)^c) \quad (3)$$

To ensure that Problem 1 admits an optimum, we have defined $\mathcal{A}_i(\theta_i^p)$ to be closed; that is, states on the boundary of an AP can be considered either inside or outside. For these boundary states, our learning algorithm can automatically detect if the demonstrator intended to visit or avoid the AP (c.f. Sec. IV-B). Any *a priori* known constraints are encoded in $\bar{\mathcal{S}}$, where $\bar{\eta}(\cdot)$ is known. In this paper, we encode in $\bar{\mathcal{S}}$ the system dynamics, start state, and if needed, a goal state separate from the APs.

We are given N_s demonstrations $\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}$ of duration T_j, which approximately solve Prob. 1, in that they are feasible (satisfy the LTL formula and known constraints) and achieve a possibly suboptimal cost. Note that Prob. 1 can be modeled with continuous (ξ_{xu}) and boolean decision variables (referred to collectively as $\mathbf{Z}$) [40]; the boolean variables determine the high-level plan, constraining the trajectory to obey boolean decisions that satisfy $\varphi(\theta^s, \theta^p)$, while the continuous component synthesizes a low-level trajectory implementing the plan. We will use different assumptions of demonstrator optimality on

the continuous/boolean parts of the problem, depending on if θ^p (Sec. IV), θ^s (Sec. V), or θ^c (Sec. VI) are being learned, and discuss how these different degrees of optimality can affect the learnability of LTL formulas (Sec. VII).

Our goal is to learn the unknown structure θ^s and AP parameters θ^p of the LTL formula $\varphi(\theta^s, \theta^p)$, as well as unknown cost function parameters θ^c, given demonstrations $\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}$ and the *a priori* known safe set $\bar{\mathcal{S}}$.

IV. Learning Atomic Proposition Parameters (θ^p)

We develop methods for learning unknown AP parameters θ^p when the cost function parameters θ^c and formula structure θ^s are known. We first review recent results [16] on learning time-invariant constraints via the KKT conditions (Sec. IV-A), show how the framework can be extended to learn θ^p (Sec. IV-B), and develop a method for extracting states which are guaranteed to satisfy/violate p_i (Sec. IV-C). In this section, we will assume that demonstrations are *locally-optimal for the continuous component* and *feasible for the discrete component*.

A. Learning time-invariant constraints via KKT

Consider a simplified variant of Prob. 1 that only involves always satisfying a single AP; this reduces Prob. 1 to a standard trajectory optimization problem:

$$\begin{aligned} \underset{\xi_{xu}}{\text{minimize}} \quad & c(\xi_{xu}) \\ \text{subject to} \quad & \mathbf{g}(\eta(x), \theta^p) \leq \mathbf{0}, \quad \forall x \in \xi_{xu} \\ & \bar{\eta}(\xi_{xu}) \in \bar{\mathcal{S}} \subseteq \mathcal{C} \end{aligned} \tag{4}$$

To ease notation, θ^c is assumed known in Sec. IV-V and reintroduced in Sec. VI. Suppose we rewrite the constraints of (4) as $\mathbf{h}^k(\eta(\xi_{xu})) = \mathbf{0}$, $\mathbf{g}^k(\eta(\xi_{xu})) \leq \mathbf{0}$, and $\mathbf{g}^{\neg k}(\eta(\xi_{xu}), \theta^p) \leq \mathbf{0}$, where $k/\neg k$ group together known/unknown constraints. Then, with Lagrange multipliers λ and ν, the KKT conditions (first-order necessary conditions for local optimality [10]) of the jth demonstration ξ_j^{dem}, denoted $\text{KKT}(\xi_j^{\text{dem}})$, are:

$$\begin{aligned} \text{Primal} \quad & \mathbf{h}^k(\eta(x_t^j)) = \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5a)} \\ \text{feasibility:} \quad & \mathbf{g}^k(\eta(x_t^j)) \leq \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5b)} \\ & \mathbf{g}^{\neg k}(\eta(x_t^j), \theta^p) \leq \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5c)} \\ \text{Lagrange mult.} \quad & \boldsymbol{\lambda}_t^{j,k} \geq \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5d)} \\ \text{nonnegativity:} \quad & \boldsymbol{\lambda}_t^{j,\neg k} \geq \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5e)} \\ \text{Complementary} \quad & \boldsymbol{\lambda}_t^{j,k} \odot \mathbf{g}^k(\eta(x_t^j)) = \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5f)} \\ \text{slackness:} \quad & \boldsymbol{\lambda}_t^{j,\neg k} \odot \mathbf{g}^{\neg k}(\eta(x_t^j), \theta^p) = \mathbf{0}, \quad t = 1, \dots, T_j && \text{(5g)} \\ \text{Stationarity:} \quad & \nabla_{x_t} c(\xi_j^{\text{dem}}) + \boldsymbol{\lambda}_t^{j,k\top} \nabla_{x_t} \mathbf{g}^k(\eta(x_t^j)) \\ & \quad + \boldsymbol{\lambda}_t^{j,\neg k\top} \nabla_{x_t} \mathbf{g}^{\neg k}(\eta(x_t^j), \theta^p) && \text{(5h)} \\ & \quad + \boldsymbol{\nu}_t^{j,k\top} \nabla_{x_t} \mathbf{h}^k(\eta(x_t^j)) = \mathbf{0}, \quad t = 1, \dots, T_j \end{aligned}$$

where $\odot$ denotes elementwise multiplication. We vectorize the multipliers $\boldsymbol{\lambda}_t^{j,k} \in \mathbb{R}^{N_k^{\text{ineq}}}$, $\boldsymbol{\lambda}_t^{j,\neg k} \in \mathbb{R}^{N_{\neg k}^{\text{ineq}}}$, and $\boldsymbol{\nu}_t^{j,k} \in \mathbb{R}^{N_k^{\text{ineq}}}$, i.e. $\boldsymbol{\lambda}_t^{j,k} = [\lambda_{t,1}^{j,k}, \dots, \lambda_{t,N_k^{\text{ineq}}}^{j,k}]^\top$. We drop (5a)-(5b), as they involve no decision variables. Then, we can find a constraint

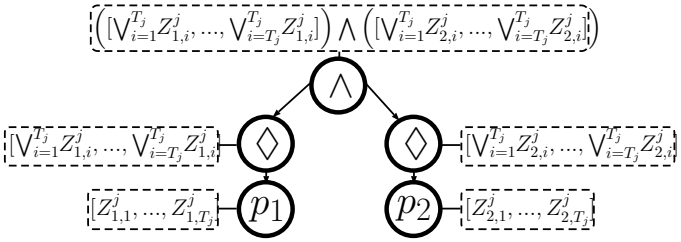

Fig. 2. A directed acyclic graph (DAG) model of the LTL formula $\varphi = (\Diamond_{[0,T_j-1]} p_1) \wedge (\Diamond_{[0,T_j-1]} p_2)$ (eventually satisfy p_1 and eventually satisfy p_2). The DAG representation can be interpreted as a parse tree for φ (c.f. Sec. V-A). The T_j boolean values for each node represent the truth value of the formula associated with the DAG subtree when evaluated on ξ_j^{dem}, starting at times $t = 1, \dots, T_j$, respectively. Each $\xi_j^{\text{dem}} \models \varphi$ iff the first entry at the root node, $(\bigvee_{i=1}^{T_j} Z_{1,i}^j) \wedge (\bigvee_{i=1}^{T_j} Z_{2,i}^j)$, is true.

which makes the N_s demonstrations locally-optimal by finding a θ^p that satisfies the KKT conditions for each demonstration:

Problem 2 (KKT, exact):

$$\begin{aligned} \text{find} \quad & \theta^p, \{\boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_t^{j,\neg k}, \boldsymbol{\nu}_t^{j,k}\}_{t=1}^{T_j}, \quad j = 1, ..., N_s \\ \text{subject to} \quad & \{\text{KKT}(\xi_j^{\text{dem}})\}_{j=1}^{N_s} \end{aligned}$$

If the demonstrations are only approximately locally-optimal, Prob. 2 may become infeasible. In this case, we can relax stationarity and complementary slackness to cost penalties:

Problem 3 (KKT, suboptimal):

$$\begin{aligned} \underset{\theta^p, \boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_t^{j,\neg k}, \boldsymbol{\nu}_t^{j,k}}{\text{minimize}} \quad & \sum_{j=1}^{N_s} \left(\|\text{stat}(\xi_j^{\text{dem}})\|_1 + \|\text{comp}(\xi_j^{\text{dem}})\|_1 \right) \\ \text{subject to} \quad & \text{(5c)} - \text{(5e)}, \ \forall \xi_j^{\text{dem}}, \ j = 1, \dots, N_s \end{aligned}$$

where $\text{stat}(\xi_j^{\text{dem}})$ denotes the LHS of Eq. (5h) and $\text{comp}(\xi_j^{\text{dem}})$ denotes the concatenated LHSs of Eqs. (5f) and (5g). For some constraint parameterizations (i.e. unions of boxes or ellipsoids [16]), Prob. 2-3 are MILP-representable and can be efficiently solved; we discuss this in further detail in Sec. IV-B.

B. Modifying KKT for multiple atomic propositions

Having built intuition with the single AP case, we return to Prob. 1 and discuss how the KKT conditions change in the multiple-AP setting. We first adjust the primal feasibility condition (5c). Recall from Sec. III that we can solve Prob. 1 by finding a continuous trajectory ξ_{xu} and a set of boolean variables $\mathbf{Z}$ enforcing that $\xi_{xu} \models \varphi(\theta^s, \theta^p)$. For each ξ_j^{dem}, let $\mathbf{Z}^j(\theta_i^p) \in \{0,1\}^{N_{\text{AP}} \times T_j}$, and let the (i,t)th index $Z_{i,t}^j(\theta_i^p)$ indicate if on ξ_j^{dem}, constraint state $\kappa_t \models p_i$ for parameters θ_i^p:

$$\begin{aligned} Z_{i,t}^j(\theta_i^p) = 1 \Leftrightarrow \kappa_t \in \mathcal{S}_i(\theta_i^p) \\ Z_{i,t}^j(\theta_i^p) = 0 \Leftrightarrow \kappa_t \in \mathcal{A}_i(\theta_i^p) \end{aligned} \tag{6}$$

Since LTL operators have equivalent boolean encodings [40], the truth value of $\varphi(\theta^s, \theta^p)$ can be evaluated as a function of $\mathbf{Z}^j$, θ^p, and θ^s, denoted as $\Phi(\mathbf{Z}^j, \theta^p, \theta^s)$ (we suppress θ^s, as it is assumed known for now). For example, consider the LTL formula $\varphi(\theta^s, \theta^p) = (\Diamond_{[0,T_j-1]} p_1) \wedge (\Diamond_{[0,T_j-1]} p_2)$, which enforces that the system must eventually satisfy p_1 and eventually satisfy p_2. We can evaluate the truth value of

$\varphi(\theta^s, \theta^p)$ ξ_j^{dem} by calculating $\Phi(\mathbf{Z}^j, \theta^p) = (\bigvee_{t=1}^{T_j} Z_{1,t}^j(\theta_1^p)) \wedge (\bigvee_{t=1}^{T_j} Z_{2,t}^j(\theta_2^p))$ (c.f. Fig. 2). Boolean encodings of common temporal and logical operators can be found in [8]. Enforcing that $Z_{i,t}^j(\theta_i^p)$ satisfies (6) can be done with a big-M formulation and binary variables $\mathbf{s}_{i,t}^j \in \{0,1\}^{N_i^{\text{ineq}}}$ [7]:

$$\begin{aligned} \mathbf{g}_i(\kappa_t^j, \theta_i^p) &\leq M(\mathbf{1}_{N_i^{\text{ineq}}} - \mathbf{s}_{i,t}^j) \\ \mathbf{1}_{N_i^{\text{ineq}}}^\top \mathbf{s}_{i,t}^j - N_i^{\text{ineq}} &\leq M Z_{i,t}^j - M_\epsilon \\ \mathbf{g}_i(\kappa_t^j, \theta_i^p) &\geq -M \mathbf{s}_{i,t}^j \\ \mathbf{1}_{N_i^{\text{ineq}}}^\top \mathbf{s}_{i,t}^j - N_i^{\text{ineq}} &\geq -M(1 - Z_{i,t}^j) \end{aligned} \tag{7}$$

where $\mathbf{1}_d$ is a d-dimensional ones vector, M is a large positive number, and $M_\epsilon \in (0,1)$. In practice, M and M_ϵ can be carefully chosen to improve the solver's performance. Note that $s_{i,m,t}^j$, the mth component of $\mathbf{s}_{i,t}^j$, encodes if κ_t^j satisfies a negated $g_{i,m}(\kappa_t^j, \theta_i^p)$, i.e. if $s_{i,m,t}^j = 1$ or 0, then κ_t^j satisfies $g_{i,m}(\kappa_t^j, \theta_i^p) \leq$ or ≥ 0. We can rewrite the enforced constraint as $\mathbf{g}_i(\kappa_t^j, \theta_i^p) \odot (2\mathbf{s}_{i,t}^j - \mathbf{1}_{N_i^{\text{ineq}}}) \leq \mathbf{0}$ for each i, t; we use this form to adapt the remaining KKT conditions. While enforcing (7) is hard in general, if $\mathbf{g}_i(\kappa, \theta_i^p)$ is affine in θ_i^p for fixed κ, (7) is MILP-representable; henceforth, we assume $\mathbf{g}_i(\kappa, \theta_i^p)$ is of this form. Note that this can still describe non-convex regions, as the dependency on κ can be nonlinear. To modify complementary slackness (5g) for the multi-AP case, we note that the elementwise product in (5g) is MILP-representable:

$$\begin{aligned} \left[\boldsymbol{\lambda}_{i,t}^{j,\neg k},\ -\mathbf{g}_i(\kappa_t^j, \theta_i^p) \odot (2\mathbf{s}_{i,t}^j - \mathbf{1}_{N_i^{\text{ineq}}})\right] &\leq M \mathbf{Q}_{i,t}^j \\ \mathbf{Q}_{i,t}^j \mathbf{1}_2 &\leq \mathbf{1}_{N_i^{\text{ineq}}} \end{aligned} \tag{8}$$

where $\mathbf{Q}_{i,t}^j \in \{0,1\}^{N_i^{\text{ineq}} \times 2}$. Intuitively, (8) enforces that either 1) the Lagrange multiplier is zero and the constraint is inactive, i.e. $g_{i,m}(\kappa, \theta_i^p) \in [-M, 0]$ or $\in [0, M]$ if $s_{i,m,t}^j = 0$ or 1, 2) the Lagrange multiplier is nonzero and $g_{i,m}(\kappa_t, \theta_i^p) = 0$, or both. The stationarity condition (5h) must also be modified to consider whether a particular constraint is negated; this can be done by modifying the second line of (5h) to terms of the form $\left(\boldsymbol{\lambda}_{i,t}^{j,\neg k\top} \odot (2\mathbf{s}_{i,t}^j - 1)\right) \nabla_{x_t} \mathbf{g}_i^{\neg k}(\eta(x_t), \theta^p)$. The KKT conditions for the multi-AP case, denoted $\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})$, then are:

$$\begin{aligned} \text{Primal} \quad & \text{Equations (5a)} - \text{(5b)}, \quad t = 1, \dots, T_j && \text{(9a)} \\ \text{feasibility:} \quad & \text{Equation (7)}, \quad i = 1, \dots, N_{\text{AP}},\ t = 1, \dots, T_j && \text{(9b)} \\ \text{Lagrange} \quad & \text{Equation (5d)}, \quad t = 1, \dots, T_j && \text{(9c)} \\ \text{nonneg.:} \quad & \boldsymbol{\lambda}_{i,t}^{j,\neg k} \geq \mathbf{0}, \quad i = 1, \dots, N_{\text{AP}},\ t = 1, \dots, T_j && \text{(9d)} \\ \text{Complem.} \quad & \text{Equation (5f)}, \quad t = 1, \dots, T_j && \text{(9e)} \\ \text{slackness:} \quad & \text{Equation (8)}, \quad i = 1, \dots, N_{\text{AP}},\ t = 1, \dots, T_j && \text{(9f)} \\ \text{Stationarity:} \quad & \nabla_{x_t} c(\xi_j^{\text{dem}}) + \boldsymbol{\lambda}_t^{j,k\top} \nabla_{x_t} \mathbf{g}^k(\eta(x_t^j)) \\ & + \sum_{i=1}^{N_{\text{ineq}}} \left[\left(\boldsymbol{\lambda}_{i,t}^{j,\neg k\top} \odot (2\mathbf{s}_{i,t}^j - 1)\right) \nabla_{x_t} \mathbf{g}_i^{\neg k}(\eta(x_t^j), \theta_i^p)\right] && \text{(9g)} \\ & + \boldsymbol{\nu}_t^{j,k\top} \nabla_{x_t} \mathbf{h}^k(\eta(x_t^j)) = \mathbf{0}, \quad t = 1, \dots, T_j \end{aligned}$$

As mentioned in Sec. III, if κ_t^j lies on the boundary of AP i, the KKT conditions will automatically determine if $\kappa_t^j \in \mathcal{S}_i(\theta_i^p)$ or $\kappa_t^j \in \mathcal{A}_i(\theta_i^p)$ based on whichever option enables $\mathbf{s}_{i,t}^j$ to take values that satisfy (9). To summarize, our approach is to 1) find $\mathbf{Z}^j$, which determines the feasibility of ξ_j^{dem} for $\varphi(\theta^s, \theta^p)$, 2) find $s_{i,m,t}^j$, which link the value of $\mathbf{Z}^j$ from the AP-containment level (i.e. $\kappa_t^j \in \mathcal{S}_i(\theta_i^p)$) to the single-constraint level (i.e. $g_{i,m}(\kappa_t^j, \theta_i^p) \leq 0$), and 3) enforce that ξ_j^{dem} satisfies the KKT conditions for the continuous optimization problem defined by θ^p and fixed values of $\mathbf{s}_{i,t}^j$. Finally, we can write the problem of recovering θ^p for a fixed θ^s as:

Problem 4 (Fixed template):

$$\begin{aligned} \text{find} \quad & \theta^p, \boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_{i,t}^{j,\neg k}, \boldsymbol{\nu}_t^{j,k}, \mathbf{s}_{i,t}^j, \mathbf{Q}_{i,t}^j, \mathbf{Z}^j, \ \forall i, j, t \\ \text{subject to} \quad & \{\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})\}_{j=1}^{N_s} \end{aligned}$$

We can also encode prior knowledge in Prob. 4, i.e. known AP labels or a prior on θ_i^p, which we discuss in App. B.

C. Extraction of guaranteed learned AP

As with the constraint learning problem, the LTL learning problem is also ill-posed: there can be many θ^p which explain the demonstrations. Despite this, we can measure our confidence in the learned APs by checking if a constraint state κ is guaranteed to satisfy/not satisfy p_i. Denote $\mathcal{F}_i$ as the feasible set of Prob. 4, projected onto Θ_i^p (feasible set of θ_i^p). Then, we say κ is learned to be guaranteed contained in/excluded from $\mathcal{S}_i(\theta_i^p)$ if for all $\theta_i^p \in \mathcal{F}_i$, $G_i(\kappa) \leq 0$ / ≥ 0. Denote by:

$$\mathcal{G}_s^i \doteq \bigcap_{\theta \in \mathcal{F}_i} \{\kappa \mid G_i(\kappa, \theta) \leq 0\} \tag{10}$$

$$\mathcal{G}_{\neg s}^i \doteq \bigcap_{\theta \in \mathcal{F}_i} \{\kappa \mid G_i(\kappa, \theta) \geq 0\} \tag{11}$$

as the sets of κ which are guaranteed to satisfy/not satisfy p_i.

To query if κ is guaranteed to satisfy/not satisfy p_i, we can check the feasibility of the following problem:

Problem 5 (Query containment of κ in/outside of $\mathcal{S}_i(\theta_i^p)$):

$$\begin{aligned} \text{find} \quad & \theta^p, \boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_{i,t}^{j,\neg k}, \boldsymbol{\nu}_t^{j,k}, \mathbf{s}_{i,t}^j, \mathbf{Q}_{i,t}^j, \mathbf{Z}^j, \ \forall i, j, t \\ \text{subject to} \quad & \{\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})\}_{j=1}^{N_s} \\ & G_i(\kappa, \theta_i^p) \geq 0 \ \textbf{OR} \ G_i(\kappa, \theta_i^p) \leq 0 \end{aligned}$$

If forcing κ to (not) satisfy p_i renders Prob. 5 infeasible, we can deduce that to be consistent with the KKT conditions, κ must (not) satisfy p_i. Similarly, continuous volumes of κ which must (not) satisfy p_i can be extracted by solving:

Problem 6 (Volume extraction):

$$\begin{aligned} \underset{\substack{\varepsilon, \kappa_{\text{near}}, \theta^p, \boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_{i,t}^{j,\neg k}, \\ \boldsymbol{\nu}_t^{j,k}, \mathbf{s}_{i,t}^j, \mathbf{Q}_{i,t}^j, \mathbf{Z}^j}}{\text{minimize}} \quad & \varepsilon \\ \text{subject to} \quad & \{\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})\}_{j=1}^{N_s} \\ & \|\kappa_{\text{near}} - \kappa_{\text{query}}\|_\infty \leq \varepsilon \\ & G_i(\kappa_{\text{near}}, \theta_i^p) > 0 \ \textbf{OR} \ G_i(\kappa_{\text{near}}, \theta_i^p) \leq 0 \end{aligned}$$

Prob. 6 searches for the largest box centered around κ_{query} contained in $\mathcal{G}_s^i/\mathcal{G}_{\neg s}^i$. An explicit approximation of $\mathcal{G}_s^i/\mathcal{G}_{\neg s}^i$ can then be obtained by solving Prob. 6 for many different κ_{query}.

V. Learning Temporal Logic Structure (θ^p, θ^s)

We will discuss how to frame the search over LTL structures θ^s (Sec. V-A), the learnability of θ^s based on demonstration optimality (Sec. V-B), and how we combine notions of discrete and continuous optimality to learn θ^s and θ^p (Sec. V-C).

A. Representing LTL structure

We adapt [30] to search for a directed acyclic graph (DAG), $\mathcal{D}$, that encodes the structure of a parametric LTL formula and is equivalent to its parse tree, with identical subtrees merged. Hence, each node still has at most two children, but can have multiple parents. This framework enables both a complete search over length-bounded LTL formulas and encoding of specific formula templates through constraints on $\mathcal{D}$ [30].

Each node in $\mathcal{D}$ is labeled with an AP or operator from (1) and has at most two children; binary operators like $\wedge$ and $\vee$ have two, unary operators like $\Diamond_{[t_1,t_2]}$ have one, and APs have none (see Fig. 2). Formally, a DAG with N_{DAG} nodes, $\mathcal{D} = (\mathbf{X}, \mathbf{L}, \mathbf{R})$, can be represented as: $\mathbf{X} \in \{0,1\}^{N_{\text{DAG}} \times N_{\mathbb{g}}}$, where $\mathbf{X}_{u,v} = 1$ if node u is labeled with element v of the grammar and 0 else, and $\mathbf{L}, \mathbf{R} \in \{0,1\}^{N_{\text{DAG}} \times N_{\text{DAG}}}$, where $\mathbf{L}_{u,v} = 1$ / $\mathbf{R}_{u,v} = 1$ if node v is the left/right child of node u and 0 else. The DAG is enforced to be well-formed (i.e. there is one root node, no isolated nodes, etc.) with further constraints; see [30] for more details. Since $\mathcal{D}$ defines a parametric LTL formula, we set $\theta^s = \mathcal{D}$. To ensure that demonstration j satisfies the LTL formula encoded by $\mathcal{D}$, we introduce a satisfaction matrix $\mathbf{S}^{\text{dem}}_j \in \{0,1\}^{N_{\text{DAG}} \times T_j}$, where $\mathbf{S}^{\text{dem}}_{j,(u,t)}$ encodes the truth value of the subformula for the subgraph with root node u at time t (i.e., $\mathbf{S}^{\text{dem}}_{j,(u,t)} = 1$ iff the suffix of ξ^{dem}_j starting at time t satisfies the subformula). This can be encoded with constraints:

$$|\mathbf{S}^{\text{dem}}_{j,(u,t)} - \Phi^t_{uv}| \leq M(1 - \mathbf{X}_{u,v}) \tag{12}$$

where Φ^t_{uv} is the truth value of the subformula for the subgraph rooted at u if labeled with v, evaluated on the suffix of ξ^{dem}_j starting at time t. The truth values are recursively generated, and the leaf nodes, each labeled with some AP i, have truth values set to $\mathbf{Z}^j_i(\theta^p_i)$. Next, we can enforce that the demonstrations satisfy the formula encoded in $\mathcal{D}$ by enforcing:

$$\mathbf{S}^{\text{dem}}_{j,(\text{root},1)} = 1, \;\; j = 1, \ldots, N_s \tag{13}$$

We will also use synthetically-generated invalid trajectories $\{\xi^{\neg s}\}_{j=1}^{N_{\neg s}}$ (Sec. V-C). To ensure $\{\xi^{\neg s}\}_{j=1}^{N_{\neg s}}$ do not satisfy the formula, we add more satisfaction matrices $\mathbf{S}^{\neg s}_j$ and enforce:

$$\mathbf{S}^{\neg s}_{j,(\text{root},1)} = 0, \;\; j = 1, \ldots, N_{\neg s}. \tag{14}$$

After discussing learnability, we will show how $\mathcal{D}$ can be integrated into the KKT-based learning framework in Sec. V-C.

B. A detour on learnability

When learning only the AP parameters θ^p (Sec. IV), we assumed that the demonstrator chooses any *feasible* assignment of $\mathbf{Z}$ consistent with the specification, then finds a locally-optimal trajectory for those fixed $\mathbf{Z}$. Feasibility is enough if the structure θ^s of $\varphi(\theta^s, \theta^p)$ is known: to recover θ^p, we just need to find some $\mathbf{Z}$ which is feasible with respect to the known θ^s (i.e. $\Phi(\mathbf{Z}^j, \theta^p, \theta^s) = 1$) and makes ξ^{dem}_j locally-optimal; that is, the demonstrator can choose an arbitrarily suboptimal high-level plan as long as its low-level plan is locally-optimal for the chosen high-level plan. However, if θ^s is also unknown, only using boolean feasibility is not enough to recover meaningful logical structure, as this makes any formula φ for which $\Phi(\mathbf{Z}^j, \theta^p, \theta^s) = 1$ consistent with the demonstration, including trivially feasible formulas always evaluating to $\top$. Consider the example in Fig. 3: θ^p_1, θ^p_2 are known and we are given two kinematic demonstrations minimizing path length under input constraints, formula $\varphi = (\neg p_2 \; \mathcal{U}_{[0,T_j-1]} \; p_1) \wedge \Diamond_{[0,T_j-1]} p_2$, and start/goal constraints. Assuming boolean feasibility, we cannot distinguish between formulas in φ_f, the set of formulas for which the demonstrations are feasible in the discrete variables and locally-optimal in the continuous variables.

On the other end of the spectrum, we can assume the demonstrator is *globally-optimal*. This invalidates many structures in φ_f, i.e. the blue trajectory should not visit both $\mathcal{S}_1$ and $\mathcal{S}_2$ if $\varphi = (\Diamond_{[0,T_j-1]} p_1) \vee (\Diamond_{[0,T_j-1]} p_2)$; we achieve a lower cost by only visiting one. Using global optimality, we can distinguish between all but the formulas with globally-optimal trajectories of equal cost (formulas in φ_g), i.e. we cannot learn the ordering constraint $(\neg p_2 \; \mathcal{U}_{[0,T_j-1]} \; p_1)$ from only the blue trajectory, as it coincides with the globally-optimal trajectory for $\varphi = (\Diamond_{[0,T_j-1]} p_1) \wedge (\Diamond_{[0,T_j-1]} p_2)$; we need the yellow trajectory to distinguish the two. We now define an optimality condition between feasibility and global optimality

Definition 1 (Spec-optimality): A demonstration ξ^{dem}_j is *μ-spec-optimal (μ-SO)*, where $\mu \in \mathbb{Z}_+$, if for every index set $\iota \doteq \{(i_1, t_1), ..., (i_\mu, t_\mu)\}$ in $\mathcal{I} \doteq \{\iota \mid i_m \in \{1, ..., N_{\text{AP}}\}, t_m \in \{1, ..., T_j\}, m = 1, ..., \mu\}$, at least one of the following holds:

- ξ^{dem}_j is locally-optimal after removing the constraints associated with p_{i_m} on $\kappa^j_{t_m}$, for all $(i_m, t_m) \in \iota$.
- For each index $(i_m, t_m) \in \iota$, the formula is not satisfied for a perturbed $\mathbf{Z}$, denoted $\hat{\mathbf{Z}}$, where $\hat{Z}_{i_m,t_m}(\theta^p_{i_m}) = \neg Z_{i_m,t_m}(\theta^p_{i_m})$, for all $m = 1, \ldots, \mu$, and $\hat{Z}_{i',t'}(\theta^p_{i'}) = Z_{i',t'}(\theta^p_{i'})$ for all $(i', t') \notin \iota$.
- ξ^{dem}_j is infeasible with respect to $\hat{\mathbf{Z}}$.

Spec-optimality enforces a level of logical optimality: if a state κ^j_t on demonstration ξ^{dem}_j lies inside/outside of AP i (i.e. $G_i(\kappa^j_t, \theta^p_i) \leq 0$ / ≥ 0), and the cost $c(\xi^{\text{dem}}_j)$ can be lowered if that AP constraint is relaxed, then the constraint must hold to satisfy the specification. Intuitively, this means that the demonstrator does not visit/avoid APs which will needlessly increase the cost and are not needed to complete the task. Further discussion regarding spec-optimality is presented in App. B. As globally-optimal demonstrations must also be spec-optimal (c.f. Lem. 1), we will use spec-optimality to vastly reduce the search space when searching for formulas which make the demonstrations globally-optimal (Sec. V-C).

C. Counterexample-guided framework

In this section, we will assume that the demonstrator returns a solution to Prob. 1 which is *boundedly-suboptimal with respect to the globally optimal solution*, in that $c(\xi^{\text{dem}}_j) \leq$

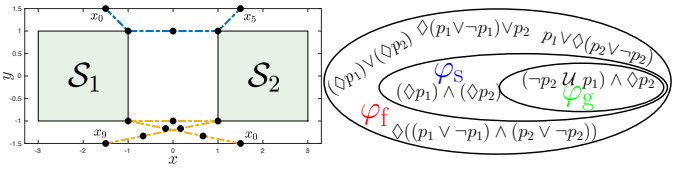

Fig. 3. **Left**: Two demonstrations which satisfy the LTL formula $\varphi = (\neg p_2 \, \mathcal{U}_{[0,T_j-1]} \, p_1) \wedge \Diamond_{[0,T_j-1]} p_2$ (first satisfy p_1, then satisfy p_2). **Right**: Some example formulas that are consistent with φ, for various levels of discrete optimality (φ_f: discrete feasibility, φ_s: spec-optimality, φ_g: discrete global optimality).

Algorithm 1: Falsification

Input: $\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}$, $\bar{\mathcal{S}}$, **Output**: $\hat{\theta}^s, \hat{\theta}^p$
$N_{\text{DAG}} \leftarrow 0$, $\{\xi^{\neg s}\} \leftarrow \{\}$
while $\neg$ *consistent* **do**
  $N_{\text{DAG}} \leftarrow N_{\text{DAG}} + 1$
  while *Problem 8 is feasible* **do**
    $\hat{\theta}^s$, $\hat{\theta}^p \leftarrow$ Problem 8$(\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}, \{\xi^{\neg s}\}, N_{\text{DAG}})$
    for $j = 1$ *to* N_s **do**
      $\xi_{xu}^j \leftarrow$ Problem 7(ξ_j^{dem})
      if $c(\xi_{xu}^j) < \frac{c(\xi_j^{\text{dem}})}{(1+\delta)}$ **then** $\{\xi^{\neg s}\} \leftarrow \{\xi^{\neg s}\} \cup \xi_{xu}$;
    if $\bigvee_{j=1}^{N_s}(c(\xi_{xu}^j) < \frac{c(\xi_j^{\text{dem}})}{(1+\delta)})$ **then** consistent $\leftarrow \top$;
    break;

$(1+\delta)c(\xi_j^*)$, for a known δ, where $c(\xi_j^*)$ is the cost of the optimal solution. This is reasonable as the demonstration should be feasible (completes the task), but may be suboptimal in terms of cost (i.e. path length, etc.), and δ can be estimated from repeated demonstrations. Under this assumption, any trajectory ξ_{xu} satisfying the known constraints $\bar{\eta}(\xi_{xu}) \in \bar{\mathcal{S}}$ at a cost lower than the suboptimality bound, i.e. $c(\xi_{xu}) \leq c(\xi_j^{\text{dem}})/(1+\delta)$, must violate $\varphi(\theta^s, \theta^p)$ [14, 15]. We can use this to reject candidate structures $\hat{\theta}^s$ and parameters $\hat{\theta}^p$. If we can find a counterexample trajectory that satisfies the candidate LTL formula $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ at a lower cost by solving Prob. 7,

Problem 7 (Counterexample search):

$$\begin{aligned} \text{find} \quad & \xi_{xu} \\ \text{subject to} \quad & \xi_{xu} \models \varphi(\hat{\theta}^s, \hat{\theta}^p) \\ & \bar{\eta}(\xi_{xu}) \in \bar{\mathcal{S}}(\xi_j^{\text{dem}}) \subseteq \mathcal{C} \\ & c(\xi_{xu}) < c(\xi_j^{\text{dem}})/(1+\delta) \end{aligned}$$

then $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ cannot be consistent with the demonstration. Thus, we can search for a consistent $\hat{\theta}^s$ and $\hat{\theta}^p$ by iteratively proposing candidate $\hat{\theta}^s$ / $\hat{\theta}^p$ by solving Prob. 8 (a modified version of Prob. 4, which we will discuss shortly) and searching for counterexamples that can prove the parameters are invalid/valid; this is summarized in Alg. 1. Heuristics on the falsification loop are discussed in App. C. We now discuss the core components of Alg. 1 (Probs. 7 and 8) in detail.

Counterexample generation: We propose different methods to solve Prob. 7 based on the dynamics. For piecewise affine systems, Prob. 7 can be solved directly as a MILP [40]. However, the LTL planning problem for general nonlinear systems is challenging [19, 29]. Probabilistically-complete sampling-based methods [19, 29] or falsification tools [2] can be applied, but can be slow on high-dimensional systems. For simplicity and speed, we solve Prob. 7 by finding a trajectory $\hat{\xi}_{xu} \models \varphi(\hat{\theta}^s, \hat{\theta}^p)$ and boolean assignment $\mathbf{Z}$ for a kinematic approximation of the dynamics via solving a MILP, then warm-start the nonlinear optimizer with $\hat{\xi}_{xu}$ and constrain it to be consistent with $\mathbf{Z}$, returning some ξ_{xu}. If $c(\xi_{xu}) < c(\xi_j^{\text{dem}})/(1+\delta)$, then we return, otherwise, we generate a new $\hat{\xi}_{xu}$. Whether this method returns a valid counterexample depends on if the nonlinear optimizer converges to a feasible solution; hence, this approach is not complete. However, we show that it works well in practice (see Sec. VIII).

Unifying parameter and structure search: When both θ^p and θ^s are unknown, they must be jointly learned due to their interdependence: learning the structure involves finding an unknown boolean function of θ^p, parameterized by θ^s, while learning the AP parameters θ^p requires knowing which APs were selected or negated, determined by θ^s. This can be done by combining the KKT (9) and DAG constraints (12)-(14) into a single MILP, which can then be integrated into Alg. 1:

Problem 8 (Learning θ^p, θ^s by global optimality, KKT):

$$\begin{aligned} \text{find} \quad & \mathcal{D}, \mathbf{S}_j^{\text{dem}}, \mathbf{S}_j^{\neg s}, \theta^p, \boldsymbol{\lambda}_t^{j,k}, \boldsymbol{\lambda}_{i,t}^{j,\neg k}, \boldsymbol{\nu}_t^{j,k}, \mathbf{s}_{i,t}^j, \mathbf{Q}_{i,t}^j, \mathbf{Z}^j, \\ & \qquad \forall i, j, t \\ \text{s.t.} \quad & \{\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})\}_{j=1}^{N_s} \\ & \text{topology constraints for } \mathcal{D} \\ & \text{Equations (12)} - \text{(13)}, \ j = 1, \ldots, N_s \\ & \text{Equation (14)}, \ j = 1, \ldots, N_{\neg s} \end{aligned}$$

In Prob. 8, since 1) the $\mathbf{Z}_i^j(\theta_i^p)$ at the leaf nodes of $\mathcal{D}$ are constrained via (7) to be consistent with θ^p and ξ_j^{dem} and 2) the formula defined by $\mathcal{D}$ is constrained to be satisfied for the $\mathbf{Z}$ via (12), the low-level demonstration ξ_j^{dem} must be feasible for the overall LTL formula defined by the DAG, i.e. $\varphi(\theta^s, \theta^p)$, where $\theta^s = \mathcal{D}$. $\text{KKT}_{\text{LTL}}(\xi_j^{\text{dem}})$ then chooses AP parameters θ^p to make ξ_j^{dem} locally-optimal for the continuous optimization induced by a fixed realization of boolean variables. Overall, Prob. 8 finds a pair of θ^p and θ^s which makes ξ_j^{dem} locally-optimal for a fixed $\mathbf{Z}^j$ which is *feasible* for $\varphi(\theta^s, \theta^p)$, i.e. $\Phi(\mathbf{Z}^j, \theta^p, \theta^s) = 1$, for all j. To also impose the spec-optimality conditions (Def. 1), we can add these constraints to Prob. 8:

$$\mathbf{S}_{j,(\text{root},1)}^{\text{dem},\hat{\mathbf{Z}}_n^j} \leq b_{nj}^1 \tag{15a}$$

$$\|\boldsymbol{\lambda}_{i_m,t_m}^{j,\neg k\top} \nabla_{x_t} \mathbf{g}_{i_m}^{\neg k}(\eta(x_t^j), \theta_{i_m}^p)\| \leq M(1 - b_{nj}^2), \quad m = 1, ..., \mu \tag{15b}$$

$$\mathbf{g}_{i_m}^{\neg k}(\eta(x_t^j), \theta_{i_m}^p) \geq -M(1 - \mathbf{e}_{nm}^j), \ m = 1, ..., \mu \tag{15c}$$

$$\mathbf{1}_{N_{\text{ineq}}^{i_m}}^\top \mathbf{e}_{nm}^j \geq \hat{Z}_{i_m t_m}^j(\theta_{i_m}^p) - b_{nj}^3, \ m = 1, ..., \mu \tag{15d}$$

$$\mathbf{g}_{i_m}^{\neg k}(\eta(x_t^j), \theta_{i_m}^p) \leq M(\hat{Z}_{i_m,t_m}^j + b_{nj}^3) \tag{15e}$$

$$b_{nj}^1 + b_{nj}^2 + b_{nj}^3 \leq 1, \quad \mathbf{b}_{nj} \in \{0,1\}^3, \quad \mathbf{e}_{nm}^j \in \{0,1\}^{N_{\text{ineq}}^{i_m}} \tag{15f}$$

for $n = 1, \ldots, |\mathcal{I}|$, where $\mathbf{S}_j^{\text{dem},\hat{Z}_n^j}$ is the satisfaction matrix for ξ_j^{dem} where the leaf nodes are perturbed to take the values

of $\hat{Z}_n^j$, where n indexes an $\iota \in \mathcal{I}$. (15a) models the case when the formula is not satisfied, (15b) models when ξ_j^{dem} remains locally-optimal upon relaxing the constraint (zero stationarity contribution), and (15c)-(15e) model the infeasible case.

Remark 1: If $\mu = 1$, the infeasibility constraints (15c)-(15e) can be ignored (since together with (15a), they are redundant), and we can modify (15f) to $b_{nj}^1 + b_{nj}^2 \leq 1$, $\mathbf{b}_{nj} \in \{0,1\}^2$.

Remark 2: It is only useful to enforce spec-optimality on index pairs $(i_1, t_1), \ldots, (i_\mu, t_\mu)$ where $G_{i_m}(\kappa_{t_m}^j, \theta_{i_m}^p) = 0$ for all $m = 1, ..., \mu$; otherwise the infeasibility case automatically holds. If θ^p is unknown, we won't know *a priori* when this holds, but if θ^p are (approximately) known, we can pre-process so that spec-optimality is only enforced for salient $\iota \in \mathcal{I}$.

Remark 3: Prob. 8 with spec-optimality constraints (15) can be used to directly search for a $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ which can be satisfied by visiting a set of APs in any order (i.e. surveillance-type tasks) without using the loop in Alg. 1, since (15) directly enforces that any AP (1-SO) or a set of APs (μ-SO) which were visited and which prevent the trajectory cost from being lowered must be visited for any candidate $\varphi(\hat{\theta}^s, \hat{\theta}^p)$.

VI. Learning Cost Function Parameters (θ^p, θ^s, θ^c)

If θ^c is unknown, it can be learned by modifying KKT_{LTL} to also consider θ^c in the stationarity condition: all terms like $\nabla_{\xi_{xu}} c(\xi_j^{\text{dem}})$ should be modified to $\nabla_{\xi_{xu}} c(\xi_j^{\text{dem}}, \theta^c)$. When $c(\cdot,\cdot)$ is affine in θ^c for fixed ξ_j^{dem}, the stationarity condition is representable with a MILP constraint. However, the falsification loop in Alg. 1 requires a fixed cost function in order to judge if a trajectory is a counterexample. Thus, one valid approach is to first solve Prob. 8, searching also for θ^c, then fixing θ^c, and running Alg. 1 for the fixed θ^c (see App. A). Note that this procedure either eventually returns an LTL formula consistent with the fixed θ^c, or Alg. 1 becomes infeasible, and a new θ^c must be generated and Alg. 1 rerun. While this procedure is guaranteed to eventually return a set of θ^c, θ^s, and θ^p which make each ξ_j^{dem} globally-optimal with respect to $c(\xi_{xu}, \theta^c)$ under $\varphi(\theta^s, \theta^p)$, it may require iterating through an infinite number of candidate θ^c and hence is not guaranteed to terminate in finite time (Cor. 3). Nevertheless, we note that for certain simple classes of formulas (Rem. 3), a consistent set of θ^c, θ^s, and θ^p can be recovered in one shot.

VII. Theoretical Analysis

In this section, we prove that our method is complete under some assumptions, without (Thm. 1) or with (Cor. 2) spec-optimality, and that we can compute guaranteed conservative estimates of $\mathcal{S}_i/\mathcal{A}_i$ (Thm. 2). Finally, we show that leveraging stronger optimality assumptions on the demonstrator shrinks the set of consistent formulas (Thm. 3). See App. C for proofs.

Assumption 1: Prob. 7 is solved with a complete planner.

Assumption 2: Each demonstration is locally-optimal (i.e. satisfies the KKT conditions) for fixed boolean variables.

Assumption 3: The true parameters θ^p, θ^s, and θ^c are in the hypothesis space of Prob. 8: $\theta^p \in \Theta_p$, $\theta^s \in \Theta_s$, $\theta^c \in \Theta_c$.

Theorem 1 (Completeness and consistency, unknown case): Under Assumptions 1-3, Alg. 1 is guaranteed to return a formula $\varphi(\theta^s, \theta^p)$ such that 1) $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ and 2) ξ_j^{dem} is globally-optimal under $\varphi(\theta^s, \theta^p)$, for all j, 3) if such a formula exists and is representable by the provided grammar.

Corollary 1 (Shortest formula): Let N^* be the minimal size DAG for which there exists (θ^p, θ^s) such that $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ for all j. Under Assumptions 1-3, Alg. 1 is guaranteed to return a DAG of length N^*.

Lemma 1: All globally-optimal trajectories are μ-SO.

Corollary 2 (Alg. 1 with spec-optimality): By modifying Alg. 1 so that Prob. 8 uses constraints (15), Alg. 1 still returns a consistent solution $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ if one exists, i.e. each ξ_j^{dem} is feasible and globally optimal for each $\varphi(\hat{\theta}^s, \hat{\theta}^p)$.

Corollary 3 (Completeness and consistency, unknown cost): Under Assumptions 1-3, Alg. 2 returns a formula $\varphi(\theta^s, \theta^p)$ such that 1) $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ and 2) ξ_j^{dem} is globally-optimal with respect to θ^c under the constraints of $\varphi(\theta^s, \theta^p)$, for all j, 3) if such a formula exists and is representable by the provided grammar.

Theorem 2 (Conservativeness for fixed template): Suppose that θ^s and θ^c are known, and θ^p is unknown. Then, extracting $\mathcal{G}_s^i$ and $\mathcal{G}_{\neg s}^i$, as defined in (10)-(11), from the feasible set of Prob. 4 projected onto Θ_i^p (denoted $\mathcal{F}_i$), returns $\mathcal{G}_s^i \subseteq \mathcal{S}_i$ and $\mathcal{G}_{\neg s}^i \subseteq \mathcal{A}_i$, for all $i \in \{1, \ldots, N_{\text{AP}}\}$.

Theorem 3 (Distinguishability): For the consistent formula sets defined in Sec. V-B, we have $\varphi_g \subseteq \varphi_{\tilde{\mu}\text{-SO}} \subseteq \varphi_{\hat{\mu}\text{-SO}} \subseteq \varphi_f$, for $\tilde{\mu} > \hat{\mu}$.

VIII. Experimental Results

We show that our algorithm outperforms a competing method, can learn shared task structure from demonstrations across environments, and can learn LTL formulas θ^p, θ^s and uncertain cost functions θ^c on high-dimensional problems. Please refer to the supplementary video for visualizations of the results: **https://youtu.be/cpUEcWCUMqc**.

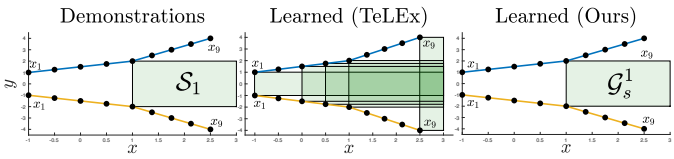

Fig. 4. Toy example for baseline comparison [22].

Baseline comparison: Likely the closest method to ours is [22], which learns a pSTL formula that is tightly satisfied by the demonstrations via solving a nonconvex problem to local optimality: $\arg\max_{\theta^p} \min_j \tau(\theta^p, \xi_j^{\text{dem}})$, where $\tau(\theta^p, \xi_j^{\text{dem}})$ measures how tightly ξ_j^{dem} fits the learned formula. We run the authors' code [21] on a toy problem (see Fig. 4), where the demonstrator has kinematic constraints, minimizes path length, and satisfies start/goal constraints and $\varphi = \Diamond_{[0,8]} p_1$, where $x \models p_1 \Leftrightarrow [I_{2\times 2}, -I_{2\times 2}]^\top x \leq [3, 2, -1, 2]^\top = [3, \theta_1^p]^\top$. We assume the structure θ^s is known, and we aim to learn θ^p to explain why the demonstrator deviated from an optimal straight-line path to the goal. Solving Prob. 6 returns $\mathcal{G}_s^1 = \mathcal{S}_1$ (Fig. 4, right). On the other hand, we run TeLEx multiple times, converging to different local optima, each corresponding

to a "tight" θ^p (Fig. 4, center): TeLEx cannot distinguish between multiple different "tight" θ^p, which makes sense, as the method tries to find *any* "tight" solution. This example suggests that if the demonstrations are goal-directed, a method that leverages their optimality is likely to better explain them.

Learning shared task structure: In this example, we show that our method can extract logical structure shared between demonstrations that complete the same high-level task, but in different environments (Fig. 5). A point robot must first go to the mug (p_1), then go to the coffee machine (p_2), and then go to goal (p_3) while avoiding obstacles (p_4, p_5). As the floor maps differ, θ^p also differ, and are assumed known. We add two relevant primitives to the grammar, sequence: $\varphi_1 \mathcal{Q} \varphi_2 \doteq \neg\varphi_2 \mathcal{U}_{[0,T_j-1]} \varphi_1$, enforcing that φ_2 cannot occur until after φ_1 has occurred for the first time, and avoid: $\mathcal{V}\varphi \doteq \Box_{[0,T_j-1]}\neg\varphi$, enforcing φ never holds over $[1, T_j]$. Then, the true formula is: $\varphi^* = \mathcal{V}p_4 \wedge \mathcal{V}p_5 \wedge (p_1 \mathcal{Q} p_2) \wedge (p_2 \mathcal{Q} p_3) \wedge \Diamond_{[0,T_j-1]} p_3$.

Suppose first that we are given the blue demonstration in Env. 2. Running Alg. 1 with 1-SO constraints (15) terminates in one iteration at $N_{\text{DAG}} = 14$ with $\varphi_0 = \mathcal{V}p_4 \wedge \mathcal{V}p_5 \wedge \Diamond_{[0,T_j-1]} p_2 \wedge \Diamond_{[0,T_j-1]} p_3 \wedge (p_1 \mathcal{Q} p_2)$: always avoid obstacles 1 and 2, eventually reach coffee and goal, and visit mug before coffee. This formula is insufficient to complete the true task (the ordering constraint between coffee and goal is not learned). This is because the optimal trajectories satisfying φ_0 and φ^* are the same cost, i.e. both φ_0 and φ^* are consistent with the demonstration and could have been returned, and $\varphi_0, \varphi^* \in \varphi_g$ (c.f. Sec. VII). Now, we also use the blue demonstration from Env. 1 (two examples total). Running Alg. 1 terminates in two iterations at $N_{\text{DAG}} = 14$ with the formulas $\varphi_1 = \mathcal{V}p_4 \wedge \mathcal{V}p_5 \wedge \Diamond_{[0,T_j-1]} p_1 \wedge \Diamond_{[0,T_j-1]} p_2 \wedge \Diamond_{[0,T_j-1]} p_3$ (which enforces that the mug, coffee, and goal must be eventually visited, but in any order, while avoiding obstacles) and $\varphi_2 = \varphi^*$. Since the demonstration in Env. 1 doubles back to the coffee before going to goal, increasing its cost over first going to goal and then to coffee, the ordering constraint between the two is learnable. We also plot the generated counterexample (Fig. 5, yellow), which achieves a lower cost, since φ_1 involves no ordering constraints. We use the learned formula to plan a path completing the task in a new environment (App. E). Overall, this example suggests we can use demonstrations in different environments to learn shared task structure and disambiguate between potential explanations.

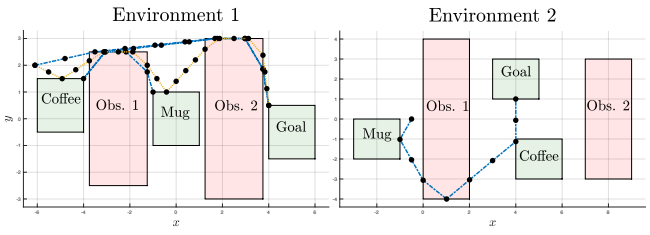

Fig. 5. Different environments (different θ^p) with shared task (same θ^s).

Multi-stage manipulation task: We consider the setup in Figs. 1, 6 of teaching a 7-DOF Kuka iiwa robot arm to prepare a drink: first move the end effector to the button on the faucet (p_1), then grasp the cup (p_2), then move the cup to the

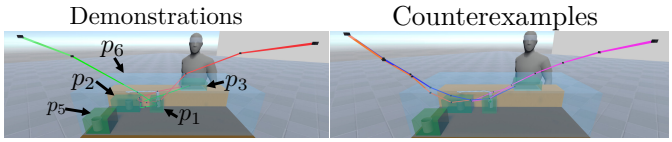

Fig. 6. Demonstrations and counterexamples for the manipulation task.

customer (p_3), all while avoiding obstacles. After grasping the cup, an end-effector pose constraint $(\alpha, \beta, \gamma) \in \mathcal{S}_4(\theta_4^p)$ (p_4) must be obeyed. We add two "distractor" APs: a different cup (p_5) and a region (p_6) where the robot can hand off the cup. We also modify the grammar to include the sequence operator $\mathcal{Q}$, (defined as before), and add an "after" operator $\varphi_1 \mathcal{T} \varphi_2 \doteq \Box_{[0,T_j-1]}(\varphi_2 \rightarrow \Box_{[0,T_j-1]}\varphi_1)$, that is, φ_1 must hold after and including the first timestep where φ_2 holds. The true formula is: $\varphi^* = (p_1 \mathcal{Q} p_2) \wedge (p_2 \mathcal{Q} p_3) \wedge \Diamond_{[0,T_j-1]} p_3 \wedge (p_4 \mathcal{T} p_2)$. We use a kinematic arm model: $j^i_{t+1} = j^i_t + u^i_t$, $i = 1, \ldots, 7$, where $\|u_t\|_2^2 \leq 1$ for all t. Two suboptimal human demonstrations ($\delta = 0.7$) optimizing $c(\xi_{xu}) = \sum_{t=1}^{T-1} \|j_{t+1} - j_t\|_2^2$ are recorded in virtual reality. We assume we have nominal estimates of the AP regions $\mathcal{S}_i(\theta^p_{i,\text{nom}})$ (i.e. from a vision system), and we want to learn the θ^s and θ^p of φ^*.

We run Alg. 1 with the 1-SO constraints (15), and encode the nominal θ^p_i by enforcing that $\Theta^p_i = \{\theta^p_i \mid \|\theta^p_i - \theta^p_{i,\text{nom}}\|_1 \leq 0.05\}$. At $N_{\text{DAG}} = 11$, the inner loop runs for 3 iterations (each taking 30 minutes on an i7-7700K processor), returning candidates $\varphi_1 = (p_1 \mathcal{Q} p_3) \wedge (p_2 \mathcal{Q} p_3) \wedge (\Diamond_{[0,T_j-1]} p_3) \wedge (p_4 \mathcal{T} p_3)$, $\varphi_2 = (p_1 \mathcal{Q} p_3) \wedge (p_2 \mathcal{Q} p_3) \wedge (\Diamond_{[0,T_j-1]} p_3) \wedge (p_4 \mathcal{T} p_2)$, and $\varphi_3 = \varphi^*$. φ_1 says that before going to the customer, the robot has to visit the button and cup in any order, and then must satisfy the pose constraint after visiting the cup. φ_2 has the meaning of φ^*, except the robot can go to the button or cup in any order. Note that φ_3 is a stronger formula than φ_2, and φ_2 than φ_1; this is a natural result of the falsification loop, which returns incomparable or stronger formulas with more iterations, as the counterexamples rule out weaker or equivalent formulas. Also note that the distractor APs don't feature in the learned formulas, even though both demonstrations pass through p_6. This happens for two reasons: we increase N_{DAG} incrementally and there was no room to include distractor objects in the formula (since spec-optimality may enforce that p_1-p_3 appear in the formula), and even if N_{DAG} were not minimal, p_6 would not be guaranteed to show up, since visiting p_6 does not increase the trajectory cost.

We plot the counterexamples in Fig. 6: blue/purple are from iteration 1; orange is from iteration 2. They save cost by violating the ordering and pose constraints: from the left start state, the robot can save cost if it visits the cup before the button (blue, orange trajectories), and loosening the pose constraint can reduce joint space cost (orange, purple trajectories). The right demonstration produces no counterexample in iteration 2, as it is optimal for this formula (changing the order does not lower the optimal cost). For the learned θ^p, $\theta^p_i = \theta^p_{i,\text{nom}}$ except for p_2, p_3, where the box shrinks slightly from the nominal; this is because by tightening the box, a Lagrange multiplier can be increased to reduce the KKT residual. We use the learned θ^p, θ^s to plan formula-satisfying trajectories from new start

states (see App. F). Overall, this example suggests that Alg. 1 can recover θ^p and θ^s on a high-dimensional problem and

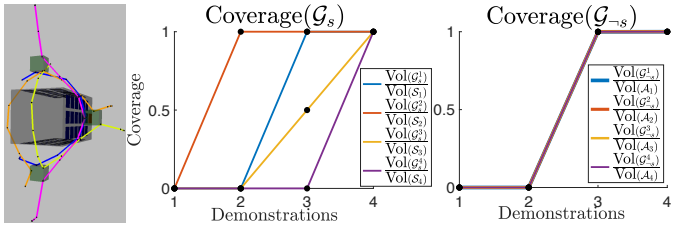

Fig. 7. Quadrotor surveillance demonstrations and learning curves.

Multi-stage quadrotor surveillance: We demonstrate that we can jointly learn θ^p, θ^s, and θ^c in one shot on a 12D nonlinear quadrotor system (see App. G). We are given four demonstrations of a quadrotor surveilling a building (Fig. 7): it needs to visit three regions of interest (Fig. 7, green) while not colliding with the building. All visitation constraints can be learned directly with 1-SO (see Rem. 3) and collision-avoidance can also be learned with 1-SO, with enough demonstrations. The true formula is $\varphi^* = \Diamond_{[0,T_j-1]} p_1 \wedge \Diamond_{[0,T_j-1]} p_2 \wedge \Diamond_{[0,T_j-1]} p_3 \wedge \Box_{[0,T_j-1]} \neg p_4$, where p_1-p_3 represent the regions of interest and p_4 is the building. We aim to learn θ^p_i for the parameterization $\mathcal{S}_i(\theta^p_i) = \{[I_{3\times 3}, -I_{3\times 3}]^\top [x,y,z]^\top \leq \theta^p_i\}$, assuming $\theta^p_{4,6} = 0$ (the building is not hovering). The demonstrations minimize $c(\xi_{xu}, \theta^c) = \sum_{r\in R} \sum_{t=1}^{T-1} \gamma_r (r_{t+1} - r_t)^2$, where $R = \{x, y, z, \dot{\alpha}, \dot{\beta}, \dot{\gamma}\}$ and $\gamma_r = 1$, i.e. equal penalties to path length and angular acceleration. We assume $\gamma_r \in [0.1, 1]$ and is unknown: we want to learn the cost weights for each state.

Solving Prob. 8 with 1-SO conditions (at $N_{\text{DAG}} = 12$) takes 44 minutes and recovers θ^p, θ^s, and θ^c in one shot. To evaluate the learned θ^p, we show in Fig. 7 that the coverage of the $\mathcal{G}^i_s$ and $\mathcal{G}^i_{\neg_s}$ for each p_i (computed by fixing the learned θ^s and running Prob. 6) monotonically increases with more data. In terms of recovered θ^s, with one demonstration, we return $\varphi_1 = \Diamond_{[0,T_j-1]} p_2 \wedge \Diamond_{[0,T_j-1]} p_3 \wedge \Diamond_{[0,T_j-1]} p_4 \wedge \Box_{[0,T_j-1]} \neg p_1$. This highlights the fact that since we are not provided labels, there is an inherent ambiguity of how to label the regions of interest (i.e. p_i, $i = 1, \ldots, 3$ can be associated with any of the green boxes in Fig. 7 and be consistent). Also, one of the regions of interest in φ gets labeled as the obstacle (i.e. p_1 and p_4 are swapped), since one demonstration is not enough to disambiguate which of the four p_i should touch the ground. Note that this ambiguity can be eliminated if labels are provided (see App. B) or if more demonstrations are provided: for two and more demonstrations, we learn $\varphi_i = \varphi^*$, $i = 2, \ldots, 4$. When using all four demonstrations, we recover the cost parameters θ^c and structure θ^s exactly, i.e. $\varphi(\hat{\theta}^s, \hat{\theta}^p) = \varphi^*$, and fixing the learned θ^s and running Prob. 6 returns $\mathcal{G}^i_s = \mathcal{S}_i$ and $\mathcal{G}^i_{\neg_s} = \mathcal{A}_i$, for all i. The learned θ^c, θ^s, and θ^p are used to plan trajectories that efficiently complete the task for different initial and goal states (see App. G). Overall, this example suggests that our method can jointly recover a consistent set of θ^p, θ^s, and θ^c for high-dimensional systems.

IX. Conclusion

We present an method that learns LTL formulas with unknown atomic propositions and logical structure from only positive demonstrations, assuming the demonstrator is optimizing an uncertain cost function. We use both implicit (KKT) and explicit (algorithmically generated lower-cost trajectories) optimality conditions to reduce the hypothesis space of LTL specifications consistent with the demonstrations. In future work, we aim to robustify our method to mislabeled demonstrations, explicitly consider demonstration suboptimality arising from risk, and reduce our method's computation time.

Acknowledgments

The authors thank Daniel Neider for insightful discussions. This research was supported in part by an NDSEG fellowship, NSF grants IIS-1750489 and ECCS-1553873, and ONR grants N00014-17-1-2050 and N00014-18-1-2501.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004. doi: 10.1145/1015330.1015430.

[2] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 254–257, 2011.

[3] Brandon Araki, Kiran Vodrahalli, Thomas Leech, Cristian Ioan Vasile, Mark Donahue, and Daniela Rus. Learning to plan with logical automata. In *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

[4] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 2009.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[6] Alexey Bakhirkin, Thomas Ferrère, and Oded Maler. Efficient parametric identification for STL. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2018, Porto, Portugal, April 11-13, 2018*, pages 177–186, 2018. doi: 10.1145/3178126.3178132. URL https://doi.org/10.1145/3178126.3178132.

[7] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

[8] Armin Biere, Keijo Heljanko, Tommi A. Junttila, Timo Latvala, and Viktor Schuppan. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2(5), 2006.

[9] Giuseppe Bombara, Cristian Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016*, pages 1–10, 2016.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[11] Sara Bufo, Ezio Bartocci, Guido Sanguinetti, Massimo Borelli, Umberto Lucangelo, and Luca Bortolussi. Temporal logic based monitoring of assisted ventilation in intensive care patients. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, pages 391–403, 2014.

[12] Sylvain Calinon and Aude Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *International Conference on Intelligent Robots and Systems (IROS)*, 2008. doi: 10.1109/IROS.2008.4650593.

[13] Alberto Camacho and Sheila A. McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*, pages 621–630, 2019.

[14] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018. URL arxiv.org/abs/1812.07084.

[15] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning parametric constraints in high dimensions from demonstrations. *3rd Conference on Robot Learning (CoRL)*, 2019. URL arxiv.org/abs/1910.03477.

[16] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning constraints from locally-optimal demonstrations under cost function uncertainty. *In Robotics and Automation Letters (RA-L)*, 2020. URL arxiv.org/abs/2001.09336.

[17] Stéphane Demri and Philippe Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.*, 174(1):84–103, 2002.

[18] Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *International Journal of Robotics Research (IJRR)*, 36(13-14):1474–1488, 2017. doi: 10.1177/0278364917745980.

[19] Jie Fu, Ivan Papusha, and Ufuk Topcu. Sampling-based approximate optimal control under temporal logic constraints. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, April 18-20, 2017*, pages 227–235, 2017.

[20] Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology, 7th International Conference, CMSB 2009, Bologna, Italy, August 31-September 1, 2009. Proceedings*, pages 218–234, 2009.

[21] Susmit Jha. susmitjha/telex. URL https://github.com/susmitjha/TeLEX.

[22] Susmit Jha, Ashish Tiwari, Sanjit A. Seshia, Tuhin Sahai, and Natarajan Shankar. Telex: learning signal temporal logic from positive examples using tightness metric. *Formal Methods in System Design*, 54(3):364–387, 2019.

[23] Miles Johnson, Navid Aghasadeghi, and Timothy Bretl. Inverse optimal control for deterministic continuous-time

nonlinear systems. In *IEEE Conference on Decision and Control (CDC)*, 2013.

[24] Arezou Keshavarz, Yang Wang, and Stephen P. Boyd. Imputing a convex objective function. In *IEEE International Symposium on Intelligent Control (ISIC)*, pages 613–619. IEEE, 2011.

[25] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. Temporal logic inference for classification and prediction from data. In *17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC'14, Berlin, Germany, April 15-17, 2014*, pages 273–282, 2014.

[26] Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control*, 62(3):1210–1222, 2017.

[27] Sanjay Krishnan, Animesh Garg, Richard Liaw, Brijen Thananjeyan, Lauren Miller, Florian T. Pokorny, and Ken Goldberg. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *International Journal of Robotics Research (IJRR)*, 38(2-3), 2019.

[28] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation for parametric STL. In *2019 IEEE Intelligent Vehicles Symposium, IV 2019, Paris, France, June 9-12, 2019*, pages 185–192, 2019. doi: 10.1109/IVS.2019.8814167. URL https://doi.org/10.1109/IVS.2019.8814167.

[29] Lening Li and Jie Fu. Sampling-based approximate optimal temporal logic planning. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 1328–1335, 2017.

[30] Daniel Neider and Ivan Gavran. Learning linear temporal properties. In *2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018*, pages 1–10, 2018.

[31] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 663–670, San Francisco, CA, USA, 2000. ISBN 1-55860-707-2.

[32] A. L. Pais, Keisuke Umezawa, Yoshihiko Nakamura, and A. Billard. Learning robot skills through motion segmentation and constraints extraction. ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2013.

[33] Ivan Papusha, Min Wen, and Ufuk Topcu. Inverse optimal control with regular language specifications. In *2018 Annual American Control Conference, ACC 2018, Milwaukee, WI, USA, June 27-29, 2018*, pages 770–777, 2018.

[34] Pravesh Ranchod, Benjamin Rosman, and George Dimitri Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 471–477, 2015.

[35] Nathan D. Ratliff, J. Andrew Bagnell, and Martin Zinkevich. Maximum margin planning. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 729–736, 2006.

[36] Francesco Sabatino. Quadrotor control: modeling, nonlinearcontrol design, and simulation, 2015.

[37] Ankit Shah, Pritish Kamath, Julie A. Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS) 2018*, pages 3808–3817, 2018.

[38] Prashant Vaidyanathan, Rachael Ivison, Giuseppe Bombara, Nicholas A. DeLateur, Ron Weiss, Douglas Densmore, and Calin Belta. Grid-based temporal logic inference. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, pages 5354–5359, 2017.

[39] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho, and Sanjit A. Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 5372–5382, 2018.

[40] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 5319–5325, 2014.

[41] Zhe Xu, Alexander J. Nettekoven, A. Agung Julius, and Ufuk Topcu. Graph temporal logic inference for classification and identification. In *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*, pages 4761–4768. IEEE, 2019.

[42] Weichao Zhou and Wenchao Li. Safety-aware apprenticeship learning. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, pages 662–680, 2018.

APPENDIX

In this supplementary material, we present a detailed description of the LTL semantics that we use (Appendix A), further explanation and motivation for spec-optimality, a discrete optimality condition that we leverage (Appendix B), an expanded version of our theoretical analysis (Appendix C), including proofs and additional comments, minor extensions and expanded details on our method (Appendix D), and some additional experimental details (Appendices E-G).

APPENDIX A
LTL SEMANTICS

We denote the satisfaction of a formula $\varphi(\theta^s, \theta^p)$ on a finite-duration trajectory ξ_{xu} of total duration T, evaluated at time $t \in \{1, 2, \dots, T\}$, as $(\xi_{xu}, t) \models \varphi$. Then, the formula satisfaction is defined recursively in (16). We emphasize that since we consider discrete-time trajectories, a time interval $[t_1, t_2]$ is evaluated only on integer time instants $\{t_1, t_1 + 1, \dots, t_2\}$; this is made concrete in (16). We also provide the following intuitive description of the LTL operators:

- The "or" operator $\varphi_1 \vee \varphi_2$ denotes a disjunction between formulas φ_1 and φ_2
- The "and" operator $\varphi_1 \wedge \varphi_2$ denotes a conjunction between formulas φ_1 and φ_2
- The "bounded-time always" operator $\square_{[t_1,t_2]}\varphi$ denotes that a formula φ always holds over the interval $[t_1, t_2]$
- The "bounded-time until" operator $\varphi_1 \, \mathcal{U}_{[t_1,t_2]} \, \varphi_2$ denotes that a formula φ_2 eventually holds during the interval $[t_1, t_2]$, and for all timesteps prior to that, φ_1 must hold.
- The "bounded-time eventually" operator $\Diamond_{[t_1,t_2]}\varphi$ denotes that a formula φ eventually has to hold during the interval $[t_1, t_2]$.

APPENDIX B
DETAILS ON SPEC-OPTIMALITY

For ease of reading, we have copied the definition of spec-optimality from the main body.

Definition B.1 (Spec-optimality): A demonstration ξ_j^{dem} is *μ-spec-optimal (μ-SO)*, where $\mu \in \mathbb{Z}_+$, if for every index set $\iota \doteq \{(i_1, t_1), ..., (i_\mu, t_\mu)\}$ in $\mathcal{I} \doteq \{\iota \mid i_m \in \{1, ..., N_{\text{AP}}\}, t_m \in \{1, ..., T_j\}, m = 1, ..., \mu\}$, at least one of the following holds:

- ξ_j^{dem} is locally-optimal after removing the constraints associated with p_{i_m} on $\kappa_{t_m}^j$, for all $(i_m, t_m) \in \iota$.
- For each index $(i_m, t_m) \in \iota$, the formula is not satisfied for a perturbed $\mathbf{Z}$, denoted $\hat{\mathbf{Z}}$, where $\hat{Z}_{i_m,t_m}(\theta^p_{i_m}) = \neg Z_{i_m,t_m}(\theta^p_{i_m})$, for all $m = 1, \dots, \mu$, and $\hat{Z}_{i',t'}(\theta^p_{i'}) = Z_{i',t'}(\theta^p_{i'})$ for all $(i', t') \notin \iota$.
- ξ_j^{dem} is infeasible with respect to $\hat{\mathbf{Z}}$.

Recall from the main body that spec-optimality aims to enforce a level of logical optimality. More specifically, spec-optimality is a measure of logical optimality, evaluated locally around a demonstration. To see this, note that the conditions in Def. B.1 are essentially checking how the local optimality of a demonstration changes as a result of local perturbations to the assignments of the discrete variables $\mathbf{Z}$. The three conditions in Def. B.1 capture the three possibilities upon perturbing $\mathbf{Z}$: the demonstration could become infeasible if $\mathbf{Z}$ is perturbed (this is what the third condition checks), the demonstration could remain feasible but local optimality may not change (this is what the first condition checks), or the demonstration could remain feasible and no longer be locally-optimal (this is what the second condition checks). By enforcing that a demonstration is spec-optimal with respect to the formula being satisfied, we enforce that this last possibility (feasible but not locally-optimal) never occurs. We would want to enforce this, for instance, if the demonstration is assumed to be globally-optimal for the true LTL formula, because there should be no alternative assignment of $\mathbf{Z}$ which admits a direction in which the demonstration cost can be improved. In terms of learning the LTL formula, demonstration spec-optimality is a useful condition because it prunes trivially-satisfiable formulas from the search space, spec-optimality necessarily holds for a demonstration to be globally-optimal (Lem. C.1), and it can be compactly enforced within a MILP, making it an efficient surrogate for enforcing global optimality. Generally, without spec-optimality, the falsification loop in Alg. 1 will need to eliminate more formulas on the way to finding a formula which makes the demonstrations globally-optimal. As a final note, we can interpret μ as a tuning knob for shifting the computation between the falsification loop and Prob. 8; imposing a larger μ can potentially rule out more formulas at the cost of adding additional constraints and decision variables to Problem 8.

To show how these conditions can be used on a concrete example, consider again the problem visualized in Fig. 3. In this problem, θ_1^p, θ_2^p are known and we are given two kinematic demonstrations minimizing path length under input

$$
\begin{aligned}
(\xi_{xu}, t) \models p_i \quad &\Leftrightarrow \quad \mathbf{g}_i(\eta_i(x_t), \theta_i^p) \leq \mathbf{0} \\
(\xi_{xu}, t) \models \neg p_i \quad &\Leftrightarrow \quad \neg((\xi_{xu}, t) \models p_i) \\
(\xi_{xu}, t) \models \varphi_1 \vee \varphi_2 \quad &\Leftrightarrow \quad (\xi_{xu}, t) \models \varphi_1 \vee (\xi_{xu}, t) \models \varphi_2 \\
(\xi_{xu}, t) \models \varphi_1 \wedge \varphi_2 \quad &\Leftrightarrow \quad (\xi_{xu}, t) \models \varphi_1 \wedge (\xi_{xu}, t) \models \varphi_2 \\
(\xi_{xu}, t) \models \square_{[t_1,t_2]}\varphi \quad &\Leftrightarrow \quad (t + t_1 \leq T) \wedge (\forall \tilde{t} \in [t + t_1, \min(t + t_2, T)], (\xi_{xu}, \tilde{t}) \models \varphi) \\
(\xi_{xu}, t) \models \varphi_1 \mathcal{U}_{[t_1,t_2]} \varphi_2 \quad &\Leftrightarrow \quad (t + t_1 \leq T) \wedge (\exists \tilde{t} \in [t + t_1, \min(t + t_2, T)] \text{ s.t. } (\xi_{xu}, \tilde{t}) \models \varphi_2) \wedge \\
&\qquad\qquad (\forall \check{t} \in [t, \tilde{t} - 1], (\xi_{xu}, \check{t}) \models \varphi_1) \\
(\xi_{xu}, t) \models \Diamond_{[t_1,t_2]}\varphi \quad &\Leftrightarrow \quad (t + t_1 \leq T) \wedge (\exists \tilde{t} \in [t + t_1, \min(t + t_2, T)] \text{ s.t. } (\xi_{xu}, \tilde{t}) \models \varphi)
\end{aligned}
\tag{16}
$$

constraints, formula $\varphi = (\neg p_2 \, \mathcal{U}_{[0,T_j-1]} \, p_1) \wedge \Diamond_{[0,T_j-1]} p_2$, and start/goal constraints. We also introduce a different candidate formula $\hat{\varphi} = \Diamond_{[0,T_j-1]} p_1 \vee \Diamond_{[0,T_j-1]} p_2$. For this example, consider only the blue demonstration. If the demonstration is globally-optimal, then $\hat{\varphi}$ is inconsistent, because to improve the trajectory cost, the demonstrator would choose to visit only one of $\mathcal{S}_1$ or $\mathcal{S}_2$, not both.

We will show how spec-optimality can be used to distinguish between φ and $\hat{\varphi}$; specifically, we show the demonstration is 1-SO with respect to φ but not for $\hat{\varphi}$. For both φ and $\hat{\varphi}$, $\mathcal{I} = \{(1,1), \ldots, (1,5), (2,1), \ldots, (2,5)\}$. Let's consider φ first. For $\iota \in \{(1,1), (2,1), (2,2), (1,3), (2,3), (1,4), (1,5), (2,5)\}$, the third condition in Def. B.1 will hold, since at these timesteps, the demonstration is not on the boundary of the paired AP. For $\iota \in \{(1,2), (2,4)\}$, the second condition in Def. B.1 will hold, since perturbing $\mathbf{Z}$ at either of these timesteps (from $Z_{1,2}(\theta_1^p) = 1$ to 0 or from $Z_{2,4}(\theta_2^p) = 1$ to 0) will cause φ to be not satisfied. Thus, the demonstration is spec-optimal with respect to φ. On the other hand, for $\hat{\varphi}$, again for $\iota \in \{(1,1), (2,1), (2,2), (1,3), (2,3), (1,4), (1,5), (2,5)\}$, the third condition in Def. B.1 will hold. However, none of the three conditions will hold for $\iota \in \{(1,2), (2,4)\}$, since the demonstration will not be locally-optimal upon relaxing the constraints for either p_1 or p_2, and since $\hat{\varphi}$ only enforces that either one of $\mathcal{S}_1$ or $\mathcal{S}_2$ are visited, $\hat{\varphi}$ is still satisfied if either $Z_{1,2}(\theta_1^p)$ or $Z_{2,4}(\theta_2^p)$ is flipped to 0. Hence, the demonstration is not spec-optimal with respect to $\hat{\varphi}$.

APPENDIX C
THEORETICAL ANALYSIS (EXPANDED)

In this section, we prove that our method is complete under some assumptions, without (Thm. C.1) or with (Cor. C.2) spec-optimality, and that we can compute guaranteed conservative estimates of $\mathcal{S}_i$/$\mathcal{A}_i$ (Thm. C.2). Finally, we show that the stronger the assumptions on the demonstrator, the smaller the set of consistent formulas (the less ill-posed the problem) (Thm. C.3). We copy the theorem statements and assumptions from the main body for ease of reading.

A. Correctness and conservativeness

Assumption C.1: Prob. 7 is solved with a complete planner.

Assumption C.2: Each demonstration is locally-optimal (i.e. satisfies the KKT conditions) for fixed boolean variables.

Assumption C.3: The true parameters θ^p, θ^s, and θ^c are in the hypothesis space of Prob. 8: $\theta^p \in \Theta^p$, $\theta^s \in \Theta^s$, $\theta^c \in \Theta^c$.

We will use these assumptions to show that when the cost function parameters θ^p are known, our falsification loop in Alg. 1 is guaranteed to return a consistent formula; that is, it makes the demonstrations globally-optimal.

Theorem C.1 (Completeness & consistency, unknown θ^s, θ^p): Under Assumptions C.1-C.3, Alg. 1 is guaranteed to return a formula $\varphi(\theta^s, \theta^p)$ such that 1) $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ and 2) ξ_j^{dem} is globally-optimal under $\varphi(\theta^s, \theta^p)$, for all j, 3) if such a formula exists and is representable by the provided grammar.

Proof: To see the first point - that Alg. 1 returns $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ such that $\xi_j^{\text{dem}} \models \varphi(\hat{\theta}^s, \hat{\theta}^p)$ for all j, note that in Prob. 8, the constraints (12)-(14) on the satisfaction matrices $\mathbf{S}_j^{\text{dem}}$ encode that each demonstration is feasible for the choice of θ^p and θ^s; hence, the output of Prob. 8 will return a feasible $\varphi(\hat{\theta}^s, \hat{\theta}^p)$. As Alg. 1 will eventually return some $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ which is an output of Prob. 8, the $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ that is ultimately returned is feasible.

Next, to see the second point - that the ultimately returned $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ makes each ξ_j^{dem} globally-optimal, note that at some iteration of the inner loop, if Prob. 7 is feasible and its solution algorithm is complete (Assumption C.1), it will return a trajectory which is lower-cost than the demonstration and satisfies $\varphi(\hat{\theta}^s, \hat{\theta}^p)$. Note that Prob. 7 will always be feasible, since Prob. 8 returns θ^p, θ^s for which the demonstration is feasible, and the feasible set of Prob. 7 contains the demonstration. The falsification loop will continue until Prob. 7 cannot produce a trajectory of strictly lower cost for each demonstration; this is equivalent to ensuring that each demonstration is globally optimal for the $\varphi(\hat{\theta}^s, \hat{\theta}^p)$.

To see the last point, we note that if there exists a formula $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ which satisfies the demonstrations, it is among the feasible set of possible outputs of Alg. 1; that is, the representation of LTL formulas, $\mathcal{D}$, is complete (c.f. Lemma 1 in [30]). ∎

We will further show that the formula returned by Alg. 1 is the shortest formula which is consistent with the demonstrations; this is due to N_{DAG} only being incremented upon infeasibility of a smaller N_{DAG} to explain the demonstrations.

Corollary C.1 (Shortest formula): Let N^* be the minimal size DAG for which there exists (θ^p, θ^s) such that $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ for all j. Under Assumptions C.1-C.3, Alg. 1 is guaranteed to return a DAG of length N^*.

Proof: The result follows since Algorithm 1 increases N_{DAG} incrementally (in the outer loop) until some $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ is returned which makes all of the demonstrations feasible and globally-optimal, and each inner iteration of Algorithm 1 is guaranteed to find a consistent $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ if one exists (c.f. Theorem C.1). ∎

As leveraging a notion of discrete optimality is crucial to reduce the search space of LTL formulas, we show that all globally-optimal demonstrations must also be μ-spec-optimal for the true specification, for any positive integer μ.

Lemma C.1: All globally-optimal trajectories are μ-SO.

Proof: We show that it is not possible for a demonstration ξ_j^{dem} to be globally-optimal while failing to satisfy (a), (b), and (c). If the constraints corresponding to p_{i_m} at $\kappa_{t_m}^j$ are relaxed, for some $\{(i_m, t_m)\}_{m=1}^{\mu}$, then ξ_j^{dem} can either remain locally-optimal (which means (a) is satisfied, and happens if all the constraints are inactive or redundant) or become not locally-optimal. If ξ_j^{dem} becomes not locally-optimal for the relaxed problem (i.e. (a) is not satisfied), then at least one of the original constraints is active, implying $\bigvee_{m=1}^{\mu} \left(G_{i_m}(\kappa_{t_m}^j) = 0\right)$. In this case, one of the following holds: either (1) each $\kappa_{t_m}^j$ lies on its constraint boundary: $\bigwedge_{m=1}^{\mu} \left(G_{i_m}(\kappa_{t_m}^j) = 0\right)$, or (2) at least one κ_{t_m} does not lie on its constraint boundary. If (2) holds, then ξ_j^{dem} must be infeasible for $\hat{\mathbf{Z}}$, so (c) must be

satisfied. If (1) holds, then ξ_j^{dem} is both feasible for $\hat{\mathbf{Z}}$ and not locally-optimal with respect to the relaxed constraints. Then, there exists some trajectory $\hat{\xi}_{xu}$ such that $c(\hat{\xi}_{xu}) < c(\xi_j^{\text{dem}})$, and for at least one m in $1, \dots, \mu$, $G_{i_m}(\hat{\kappa}_{t_m}^j) > 0$, where $\hat{\kappa}_{t_m}^j$ is the constraint state at time t_m on $\hat{\xi}_{xu}$. $\hat{\xi}_{xu}$ cannot be feasible with respect to the true specification, since it makes ξ_j^{dem} not globally-optimal, so in this case (b) must hold. ∎

Using the previous lemma, we can show that modifying Alg. 1 to additionally impose the spec-optimality conditions in Prob. 8 still enjoys the completeness properties discussed in Theorem C.1, while also in general reducing the number of falsification iterations needed as a result of the reduced search space.

Corollary C.2 (Alg. 1 with spec-optimality): By modifying Alg. 1 so that Prob. 8 uses constraints (15), Alg. 1 still returns a consistent solution $\varphi(\hat{\theta}^s, \hat{\theta}^p)$ if one exists, i.e. each ξ_j^{dem} is feasible and globally optimal for each $\varphi(\hat{\theta}^s, \hat{\theta}^p)$.

Proof: The result follows from completeness of Alg. 1 (c.f. Theorem C.1) and Lemma C.1: adding (15a)-(15c) enforces that ξ_j^{dem} are spec-optimal, and via Lemma C.1, ξ_j^{dem}, which is a globally-optimal demonstration, must also be spec-optimal. Hence, imposing constraints (15a)-(15c) is consistent with the demonstration. ∎

Next, we show how the consistency properties extend to the case of unknown cost function, if Alg. 2 returns a solution, which it is not guaranteed to do in finite time.

Corollary C.3 (Consistency, unknown θ^c): Under Assumptions C.1-C.3, if Alg. 2 terminates in finite time, it returns a formula $\varphi(\theta^s, \theta^p)$ such that 1) $\xi_j^{\text{dem}} \models \varphi(\theta^s, \theta^p)$ and 2) ξ_j^{dem} is globally-optimal with respect to θ^c under the constraints of $\varphi(\theta^s, \theta^p)$, for all j, 3) if such a formula exists and is representable by the provided grammar.

Proof: Note that Alg. 2 is simply Alg. 1 with an outer loop where potential cost parameters θ^c are chosen. From Theorem C.1, we know that under Assumptions C.1-C.2, for the true cost parameter θ^c, Alg. 1 is guaranteed to return θ^p and θ^s which make the demonstrations globally-optimal under θ^c. From Assumption C.3 and the fact that the true parameters θ^p, θ^s, and θ^c will make the demonstrations globally-optimal, we know there exists at least one consistent set of parameters (the true parameters). Then, Alg. 2 will eventually find a consistent solution (possibly the true parameters), as it iteratively runs Alg. 1 for all consistent θ^c. ∎

Finally, we show that for fixed LTL structure and cost function, querying and volume extraction (Problems 5 and 6) are guaranteed to return conservative estimates of the true $\mathcal{S}_i$ or $\mathcal{A}_i$.

Theorem C.2 (Conservativeness for unknown θ^p): Suppose that θ^s and θ^c are known, and θ^p is unknown. Then, extracting $\mathcal{G}_s^i$ and $\mathcal{G}_{\neg s}^i$, as defined in (10)-(11), from the feasible set of Prob. 4 projected onto Θ_i^p (denoted $\mathcal{F}_i$), returns $\mathcal{G}_s^i \subseteq \mathcal{S}_i$ and $\mathcal{G}_{\neg s}^i \subseteq \mathcal{A}_i$, for all $i \in \{1, \dots, N_{\text{AP}}\}$.

Proof: We first prove that $\mathcal{G}_{\neg s}^i \subseteq \mathcal{A}_i$. Suppose that there exists $\kappa \in \mathcal{G}_{\neg s}^i$ such that $\kappa \notin \mathcal{A}_i$. Then by definition, for all $\theta_i^p \in \mathcal{F}_i$, $G_i(\kappa, \theta_i^p) \geq 0$. However, we know that all locally-optimal demonstrations satisfy the KKT conditions with respect to the true parameter $\theta_i^{p,*}$; hence, $\theta_i^{p,*} \in \mathcal{F}$. Then, $x \in \mathcal{A}(\theta_i^{p,*})$. Contradiction. Similar logic holds for proving that $\mathcal{G}_s^i \subseteq \mathcal{S}_i$. Suppose that there exists $x \in \mathcal{G}_s^i$ such that $x \notin \mathcal{S}_i$. Then by definition, for all $\theta_i^p \in \mathcal{F}_i$, $G_i(\kappa, \theta_i^p) \leq 0$. However, we know that all locally-optimal demonstrations satisfy the KKT conditions with respect to the true parameter $\theta_i^{p,*}$; hence, $\theta_i^{p,*} \in \mathcal{F}_i$. Then, $\kappa \in \mathcal{S}_i(\theta_i^{p,*})$. Contradiction. ∎

B. Learnability

The goal of this theorem is to show that the stronger the assumptions on the demonstrator, the smaller the set of consistent formulas (the less ill-posed the problem).

Theorem C.3 (Distinguishability): For the consistent formula sets defined in Sec. V-B, we have $\varphi_g \subseteq \varphi_{\tilde{\mu}\text{-SO}} \subseteq \varphi_{\hat{\mu}\text{-SO}} \subseteq \varphi_f$, for $\tilde{\mu} > \hat{\mu}$.

Proof: $\varphi_g \subseteq \varphi_{\tilde{\mu}\text{-SO}}$, since per Lemma C.1, all globally-optimal trajectories are $\tilde{\mu}$-SO. Thus, restricting Prob. 8 to enforce global optimality requires more constraints than restricting Prob. 8 to enforce $\tilde{\mu}$-SO. With more constraints, the feasible set of consistent formulas cannot be larger for global optimality. Similarly, as enforcing $\tilde{\mu}$-SO requires more constraints than enforcing $\hat{\mu}$-SO, the feasible set of consistent formulas cannot be larger for $\tilde{\mu}$-SO than for $\hat{\mu}$-SO. $\varphi_{\mu\text{-SO}} \subseteq \varphi_f$, since enforcing μ-SO also enforces feasibility. Thus, restricting Prob. 8 to enforce μ-SO requires more constraints than the standard Prob. 8. With more constraints, the feasible set of consistent formulas cannot be larger for μ-SO. ∎

APPENDIX D
METHOD EXTENSIONS AND EXPANDED DETAILS

A. Unknown cost algorithm

In Algorithm 2, we formally write the analogue of the falsification approach in Alg. 1 when the cost function parameters θ^c are unknown. The approach is the same as Algorithm 1, apart from an additional outer while loop, where candidate θ^c are selected. Upon the failure of a θ^c to yield a consistent θ^p and θ^s, the θ^c is added into a set of cost parameters for Problem 8 to avoid, Θ_{av}^c. The avoidance condition can be implemented with integer constraints, i.e. $|\theta_i^c - \hat{\theta}_i^c| \geq \varepsilon_{\text{av}} - (1 - z_{\text{av}}^i)$, $\sum_i z_{\text{av}}^i \geq 1$, for $i = 1, \dots, |\theta_c|$. See Sec. VI for more discussion.

B. Encoding prior knowledge

Known labels: We have assumed that the demonstrations only include state/control trajectories and not the AP labels; this can lead to ambiguity as to which $\mathcal{S}$ should be assigned to which proposition p_i. For example, consider the example in Fig. 3 (left), where the aim is to recover $\varphi(\theta^p) = \Diamond\mathcal{S}_1(\theta_1^p) \vee \Diamond\mathcal{S}_2(\theta_2^p)$. The KKT conditions will imply that the demonstrator had to visit two boxes and their locations, but not if the left box should be labeled $\mathcal{S}_1$ or $\mathcal{S}_2$. However, in some settings it may be reasonable that the labels for each AP are provided, i.e. for an AP which requires a robot arm to grasp an object, we might have sensor data determining if the object has been grasped.

Algorithm 2: Falsification, unknown cost function

Input: $\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}$, $\bar{\mathcal{S}}$, **Output**: $\hat{\theta}^s, \hat{\theta}^p, \hat{\theta}^c$
$N_{\text{DAG}} \leftarrow 0$, $\{\xi^{\neg s}\} \leftarrow \{\}$, $\Theta_{\text{av}}^c \leftarrow \{\}$
while *true* **do**
  $\hat{\theta}^s, \hat{\theta}^p, \hat{\theta}^c \leftarrow$ Problem $8'(\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}, \{\xi^{\neg s}\}, N_{\text{DAG}}, \Theta_{\text{av}}^c)$
  while $\neg$ *consistent* **do**
    $N_{\text{DAG}} \leftarrow N_{\text{DAG}} + 1$
    while *Problem 8 is feasible* **do**
      $\hat{\theta}^s, \hat{\theta}^p \leftarrow$ Problem $8(\{\xi_j^{\text{dem}}\}_{j=1}^{N_s}, \{\xi^{\neg s}\}, N_{\text{DAG}}, \hat{\theta}^c)$
      for $j = 1$ *to* N_s **do**
        $\xi_{xu}^j \leftarrow$ Problem $7(\xi_j^{\text{dem}})$
        if $c(\xi_{xu}^j, \hat{\theta}^c) < c(\xi_j^{dem}, \hat{\theta}^c)/(1+\delta)$ **then**
          $\{\xi^{\neg s}\} \leftarrow \{\xi^{\neg s}\} \cup \xi_{xu}$
      if $\bigvee_{j=1}^{N_s}(c(\xi_{xu}^j, \hat{\theta}^c) < c(\xi_j^{dem}, \hat{\theta}^c)/(1+\delta))$ **then**
        consistent $\leftarrow \top$; break
  if *consistent* **then** return;
  else $\Theta_{\text{av}}^c \leftarrow \Theta_{\text{av}}^c \cup \hat{\theta}^c$; break;

In this case, we can incorporate this by simply constraining $\mathbf{Z}_i^j(\theta_i^p)$ to be the labels; this then removes the ambiguity mentioned earlier.

Prior knowledge on θ^p: In some settings, we may have a rough idea of θ^p, i.e. as noisy bounding boxes from a vision system. We might then want to avoid deviating from these nominal parameters, denoted θ_{nom}^p, or restrict θ^p to some region around θ_{nom}^p, denoted $\Theta_{i,\text{nom}}$, subject to the KKT conditions holding. This can be done by adding $\sum_{j=1}^{N_{\text{AP}}} \|\theta_i^p - \theta_{i,\text{nom}}^p\|_1$ as an objective or $\theta_{i,\text{nom}}^p \in \Theta_{i,\text{nom}}$ as a constraint to Prob. 4.

C. Variants on the falsification loop

Depending on the desired application, it may be useful to impose an ordering in which candidate structures θ^s are returned in line 4 of Alg. 1. For example, the user may want to return the most restrictive formulas first (i.e. formulas with the smallest language), since more restrictive formulas are less likely to admit counterexamples (and hence the falsification should terminate in fewer iterations). On the other hand, the user may want to return the least restrictive formulas first, generating many invalid formulas in order to explicitly know what formulas do not satisfy the demonstrator's wishes.

However, imposing an entailment-based ordering on the returned formulas is computationally challenging, as in general this will involve pairwise LTL entailment checks over a large set of possible LTL formulas, and each check is in PSPACE [17]. Despite this, we can heuristically approximate this by assigning weights to each node type in the DAG based on their logical "strength", such that each DAG with the same set of nodes has an overall weight $w = \sum_{u=1}^{N_{\text{DAG}}} \sum_{v=1}^{N_{\text{g}}} w_{u,v} \mathbf{X}_{u,v}$. For example, $\vee$ should be assigned a lower weight than $\wedge$, since $\vee$s can never restrict language size, while $\wedge$ can never grow it. Then, stronger/weaker formulas can be returned first by adding constraint $w \geq w_{\text{thresh}}/w \leq w_{\text{thresh}}$, where w_{thresh} is reduced/increased until a consistent formula is found.

Note that multiple consistent formula structures can be also generated by adding a constraint for Prob. 8 to not return the same formula structure and continuing the falsification loop after the first consistent formula is found.

Appendix E
Additional Experimental Details (Environment Transfer)

We use the learned θ^s to plan a trajectory which completes the high-level task (first going to the mug, then the coffee machine, then the goal, while always avoiding obstacles) in a novel environment map (with different AP parameters θ^p). Please refer to our accompanying video at **https://youtu.be/cpUEcWCUMqc** for animations of the planned trajectories.

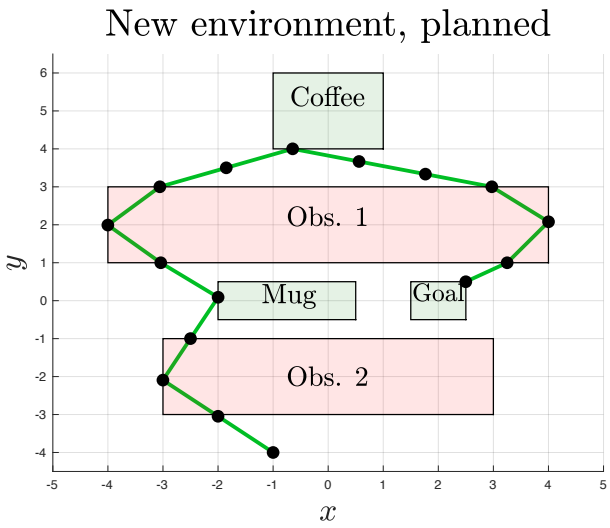

Fig. 8. Environment-transfer planned trajectory.

Appendix F
Additional Experimental Details (Manipulation)

We use the learned θ^p and θ^s to plan trajectories which complete the task from new initial conditions in the environment (Fig. 9). Please refer to our video at **https://youtu.be/cpUEcWCUMqc** for animations of the planned trajectories.

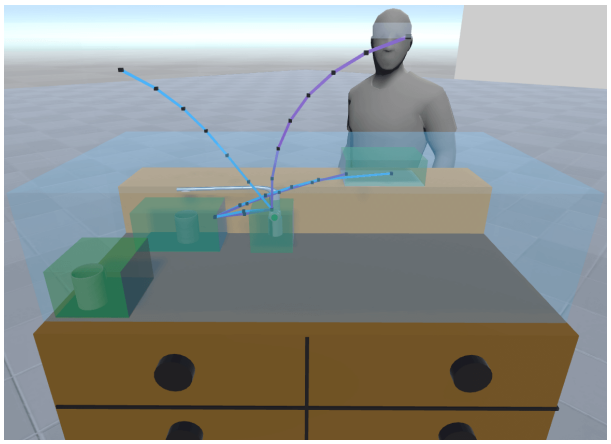

Fig. 9. Arm planned trajectories.

APPENDIX G
ADDITIONAL EXPERIMENTAL DETAILS (QUADROTOR)

The system dynamics for the quadrotor [36] are:

$$\begin{bmatrix} \dot{\chi} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \\ \ddot{\chi} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\alpha} \\ \ddot{\beta} \\ \ddot{\gamma} \end{bmatrix} = \begin{bmatrix} \dot{\chi} \\ \dot{y} \\ \dot{z} \\ \dot{\beta}\frac{\sin(\gamma)}{\cos(\beta)} + \dot{\gamma}\frac{\cos(\gamma)}{\cos(\beta)} \\ \beta\cos(\gamma) - \dot{\gamma}\sin(\gamma) \\ \dot{\alpha} + \dot{\beta}\sin(\gamma)\tan(\beta) + \dot{\gamma}\cos(\gamma)\tan(\beta) \\ -\frac{1}{m}[\sin(\gamma)\sin(\alpha) + \cos(\gamma)\cos(\alpha)\sin(\beta)]u_1 \\ -\frac{1}{m}[\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha)\sin(\beta)]u_1 \\ g - \frac{1}{m}[\cos(\gamma)\cos(\beta)]u_1 \\ \frac{I_y - I_z}{I_x}\dot{\beta}\dot{\gamma} + \frac{1}{I_x}u_2 \\ \frac{I_z - I_x}{I_y}\dot{\alpha}\dot{\gamma} + \frac{1}{I_y}u_3 \\ \frac{I_x - I_y}{I_z}\dot{\alpha}\dot{\beta} + \frac{1}{I_z}u_4 \end{bmatrix}, \tag{17}$$

with control constraints $\|u_t\|_2 \leq 10$. We time-discretize the dynamics by performing forward Euler integration with discretization time $\delta t = 1.2$ seconds. The 12D state is $x = [\chi, y, z, \alpha, \beta, \gamma, \dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma}]^\top$, and the relevant constants are $g = -9.81\text{m/s}^2$, $m = 1\text{kg}$, $I_x = 0.5\text{kg} \cdot \text{m}^2$, $I_y = 0.1\text{kg} \cdot \text{m}^2$, and $I_z = 0.3\text{kg} \cdot \text{m}^2$.

With the four demonstrations provided (see Sec. VIII), we learn θ^c, θ^p, and θ^s, and obtain a representation of $\mathcal{G}^i_s$ and $\mathcal{G}^i_{\neg s}$, for all AP p_i. Using θ^c, θ^s, and $\mathcal{G}^i_s$, we plan trajectories from new initial states to new goal states in the environment which are guaranteed to satisfy the true LTL formula; these trajectories are presented in Fig. 10. Please refer to our accompanying video at **https://youtu.be/cpUEcWCUMqc** for animations of the planned trajectories.

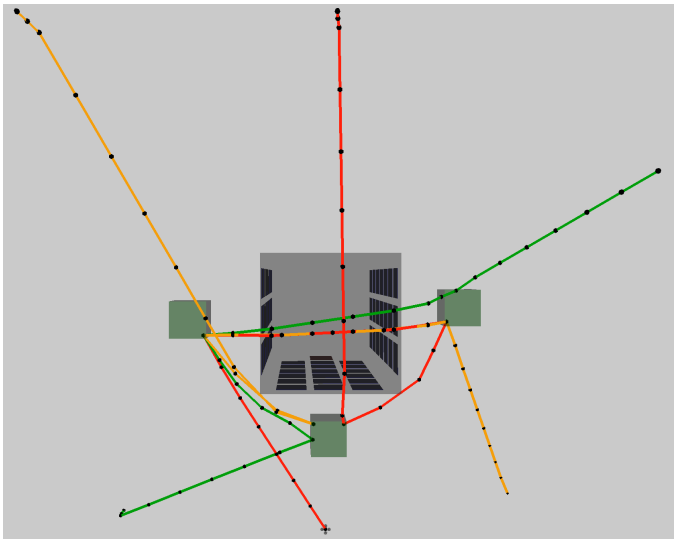

Fig. 10. Quadrotor planned trajectories.