2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Learning from Demonstrations under Stochastic Temporal Logic Constraints

Panagiotis Kyriakis, Jyotirmoy V. Deshmukh, Paul Bogdan

***Abstract*— We address the problem of learning from demonstrations when the learner must satisfy safety and/or performance requirements expressed as Stochastic Temporal Logic (StTL) specifications. We extend the maximum causal entropy inverse reinforcement learning framework to account for StTL constraints and show how to encode them via a minimal set of mixed-integer linear constraints. Our method is based on a *cut-and-generate* algorithm that iterates between two phases: in the *cut* phase, we use cutting hyperplanes to approximate the feasible region of the non-linear constraint that encodes atomic predicates and in the *generate* phase, we propagate these hyperplanes through the schematics to generate constraints for arbitrary formulas. Our algorithmic contributions are validated in different environments and specifications.**

I. Introduction

Learning from Demonstrations (*LfD*) is an emerging paradigm to design control policies [1]. In this paradigm, learning agents acquire new skills not by programming but by imitating a human expert. LfD (also known as *imitation learning*) is a vibrant research area and several classes of methods have been developed such as behavior cloning [2] and direct policy learning [3]. Inverse reinforcement learning (IRL) enables a learning agent to infer the reward function from the given demonstrations. An important difficulty is that the optimization problem is under-defined as there may exist more than one reward function that explain a given set of demonstrations [4]. To address this issue, several approaches such as feature expectation matching [5], Bayesian IRL [6] and the Maximum Causal Entropy (MCE) IRL [7] have been developed.

In a real-world setting, we often have to learn from imperfect or sub-optimal demonstrations, which may not be consistent with desired safety and performance specifications. To encode such specifications, several formal methods, such as Computational Tree Logic [8], Linear Temporal Logic [9], Signal Temporal Logic [10] have been developed and used for expressing specifications in a variety of control and learning tasks [11], [12], [13], [14], [15]. In the realm of stochastic dynamical systems (e.g., Markov Decision Processes), Probabilistic Computational Tree Logic (PCTL) has been used to express safety specifications in the context of learning from demonstrations [16]. Recently, Signal Temporal Logic (STL) was extended to three probabilistic variants, Probabilistic STL (PrSTL) [17], Chance Constrained Temporal Logic (C2TL) [18], Stochastic STL (StSTL) [19] and Stochastic Temporal Logic [20], [21]. Even though STL has been used in the context of learning from demonstrations [22], [12], its probabilistic extensions are yet to be explored.

In this paper, we address the problem of learning from demonstrations subject to Stochastic Temporal Logic (StTL) specifications. Due to its popularity, we choose to extend the maximum causal entropy inverse reinforcement learning framework and we propose a problem formulation whereby the StTL specification is imposed as a constraint in the optimization problem. We show how to encode the StTL formula via a minimal set of mixed-integer linear constraints. We generate these constraints using cutting hyperplanes as an approximation of the feasible regions of atomic predicates and then we recursively propagate these hyperplanes through the formula schematics. The benefits of our approach are two-fold: first, our encoding produces a minimal set of integer constraints, and second, by using cutting hyperplanes, we avoid having to propagate non-linear constraints.

II. Preliminaries

Markov Decision Processes. We consider the framework of infinite-time, discounted Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \delta, \mathcal{D}_{\text{init}}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the (finite) state and (finite) action space, δ is the transition kernel, $r(s,a) \equiv \mathcal{R}$ is the reward, $\mathcal{D}_{\text{init}}$ is an initial distribution and $\gamma \in \mathbb{R}$ is a discount factor. A stationary control policy π is a function that assigns a probability distribution over actions for all states. An MDP $\mathcal{M}$ paired with a policy π induces a Markov Chain $S_t^\pi,\ t = 0, 1, \dots$ over the same state space. The performance of a given control policy is measured via the expected discounted reward defined as $R(\pi) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t))$, where the expectation is taken over trajectories sampled from the Markov chain S_t^π induced by π. An optimal policy π^* is a policy that maximizes the return, i.e., $\pi^* \in \arg\max_{\pi \in \Pi} R(\pi)$. A finite set of trajectories $\mathcal{D} = \{(s_0^i, a_0^i), (s_1^i, a_1^i), (s_2^i, a_2^i), \dots\}_{i=1,2,\dots}$ obtained by executing π in $\mathcal{M}$ are called *demonstrations*.

Stochastic Temporal Logic. We consider preferences that are encoded via *Stochastic Temporal Logic (StTL)* formulas. Formally, StTL formulas are defined over atomic predicates represented by chance constraints of the aforementioned form: $\varphi = \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \mathbf{U}_I \psi$, where I is an interval.

The authors gratefully acknowledge the support by the National Science Foundation under the Career Award CPS/CNS-1453860, the NSF award under Grant Numbers CCF-1837131, MCB-1936775, CNS-1932620, and CMMI 1936624 and the DARPA Young Faculty Award and DARPA Director's Fellowship Award, under Grant Number N66001-17-1-4044, and a Northrop Grumman grant. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied by the Defense Advanced Research Projects Agency, the Department of Defense or the National Science Foundation. This work was also supported by the National Science Foundation under the CAREER Award SHF-2048094, FMitF award CCF-1837131, CPS award CNS-1932620, Toyota RD, and Northrop-Grumman Aerospace Systems.

III. StTL-Constrained Maximum Causal Entropy

In this section, we present the agent model for the StTL-constrained inverse reinforcement learning problem. Let $\mathcal{M} \setminus \mathcal{R}$ be the reduced MDP (i.e., the MDP without the reward function), φ an StTL formula that encodes safety and/or performance specifications and $\mathcal{D}$ a set of demonstrations. The agent seeks to recover a policy π that "best" imitates the given demonstrations and satisfies the specification. One way to quantify the "quality of imitation" is to study the expected value of a *reward feature vector* $\phi : \mathcal{S} \times \mathcal{A} \to [0,1]^{d_\phi}$. For a policy π and a discount factor γ, the *feature expectation vector* is defined as $f_r(\pi) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t))$. Given a set of demonstrations $\mathcal{D} = \{(s_0^i, a_0^i), (s_1^i, a_1^i), (s_2^i, a_2^i), \dots\}_{i=1,2,\dots}$, the empirical counterpart of the feature expectation vector is given by $\hat{f}_r(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D},t} \gamma^t \phi(s_t^i, a_t^i)$.

The problem of inferring the reward and/or optimal policy is ill-posed because each policy can be optimal for many reward functions. The principle of Maximum Casual Entropy (MCE) [4], [7] resolves the ambiguity by maximizing the entropy of the distribution over trajectories subject to a *feature expectation matching* constraint, i.e., $f_r(\pi) = \hat{f}_r(\mathcal{D})$. Motivated by the MCE framework, we introduce the following problem:

$$\max_{\pi} \quad H^\beta(\pi) = \mathbb{E}\left[-\sum_{t=0}^{\infty} \beta^t \log \pi(a_t|s_t) \right] \tag{1a}$$

$$f_r(\pi) = \hat{f}_r(\mathcal{D}), \tag{1b}$$

$$(S^\pi, 0) \models \varphi, \tag{1c}$$

where $H^\beta(\pi)$ is the *discounted casual entropy*, $\beta \in (0,1)$ is a discount factor and S_t^π denotes the Markov chain induced by $\mathcal{M}$ and π. Constraints that ensure that $\pi(a_t|s_t)$ is a valid distribution are omitted for simplicity. While the original MCE framework [23] assumes that the reward is a linear function $w^T f(s,a)$ for some feature vector $f(s,a)$ and optimizes the weight w, we choose the variant introduced in [24], where the authors directly optimize the policy. We choose this variant as it is more convenient for imposing the StTL specification.

Solving Problem 1 is far from trivial because the StTL specification (1c) is essentially a logic constraint. In what follows, we show how to encode the constraint (1c) via a minimal set of integer constraints and in the next section how to ensure that these constraints are linear. To this end, we parametrize the policy via a vector $\theta \in \mathbb{R}^{d_\theta}$, i.e., $\pi(a|s) = \pi_\theta(a|s)$. Let $p_t(s)$ be the state distribution of the Markov Chain S_t^π at time t. We have

$$p_t(s) = \sum_{s',a} p_{t-1}(s') \delta(s|s',a) \pi_\theta(a|s'), \tag{2}$$

for $t = 1, 2, \dots$ and $p_0(s) = \mathcal{D}_{\text{init}}$. We start by encoding StTL predicates via a set of linear constraints and then recursively propagate these constraints through the StTL schematics to generate constraints for any formula.

StTL Predicates. A predicate $\mu = P(S_t^\pi \notin F) \leq \epsilon$ over S_t^π is satisfied if and only if it holds that

$$\sum_{s \notin F} p_t(s) ds = \sum_{s \notin F, s', a} p_{t-1}(s') \delta(s|s',a) \pi_\theta(a|s') \leq \epsilon \tag{3}$$

for all times $t = 1, 2, \dots$. We define the following quantity

$$q_c(s', a) = \sum_{s \notin F} \delta(s|s', a). \tag{4}$$

By substituting Eq. (4) to Eq. (3), we obtain the following inequality

$$c_t^\mu(\theta) = \sum_{s',a} p_{t-1}(s') q_c(s',a) \pi_\theta(a|s') - \epsilon \leq 0. \tag{5}$$

where the superscript μ denotes the dependence of the constraint on the predicate. Each predicate appearing in the specification φ gives rise to such a constraint. We show how to approximate this constraint of Eq. (5) via cutting hyperplanes in the next section. For now, we assume that the predicate μ is encoded via the following set defined by a set of linear constraints

$$g_t^\mu(\theta) = \left\{ \theta \in \mathbb{R}^{d_\theta} : \theta^T d_{t,i}^\mu \leq 0, \ i = 1, 2, \dots \right\}, \tag{6}$$

where $d_{t,i}^\mu \in \mathbb{R}^{d_\theta}$ is to be determined.

Negation. To encode the negation of an atomic predicate $\neg\mu = P_t(S_t^\pi \notin F) \geq \epsilon$ we use similar reasoning as above and obtain a set of linear constraints on θ similar to Eq. (6).

Conjunction. Let $\varphi_i, \ i = 1, 2, \dots, m$ be a set of formulas and consider the formula $\varphi = \bigwedge_{i=1}^m \varphi_i$ defined by their conjunction. Let $g_t^i(\theta), i = 1, 2, \dots m$ be the sets that encode the formula φ_i at time t. Since conjunction requires simultaneous satisfaction of all predicates, formula φ is encoded by the intersection of the constraints corresponding to each predicate, i.e., by the following set

$$g_t^\wedge(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : \theta \in \bigcap_{i=1}^m g_t^i(\theta)\}. \tag{7}$$

Disjunction. Let $\varphi_i, \ i = 1, 2, \dots, m$ be a set of formulas and consider the formula $\varphi = \bigvee_{i=1}^m \varphi_i$ defined by their disjunction. Let $g_t^i(\theta), i = 1, 2, \dots m$ be the sets that encode the formula φ_i at time t and $b_t^i \in \{0,1\}$ be a set of binary variables. Then, the disjunction operator is encoded as follows

$$g_t^\vee(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : \theta \in b_t^i g_t^i(\theta), \text{ and } \sum_{i=1}^m b_t^i \geq 1\}. \tag{8}$$

The second constraint essentially ensures that at least one of the predicates is satisfied. Note that, in contrast to conjunction, the disjunction forces to introduce binary variables.

Always and Finally. The temporal operators $\mathbf{G}_{[a,b]}\varphi$ and $\mathbf{F}_{[a,b]}\varphi$ can be expressed as temporal conjunction and disjunction, respectively, over all time instances t in the interval

$I = [a, b]$. Therefore, they can be encoded by the following constraints

$$g_t^{\mathbf{G}\varphi}(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : \theta \in \bigcap_{t=a}^{b} g_t^{\varphi}(\theta)\}, \tag{9}$$

$$g_t^{\mathbf{F}\varphi}(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : \theta \in b_t g_t^{\varphi}(\theta), \text{ and } \sum_{t=a}^{b} b_t \geq 1\}, \tag{10}$$

where $g_t^{\varphi}(\theta) \leq 0$ is the set that encodes the formula φ at time t and $b_t \in \{0, 1\}$.

Until. Let φ, ψ be two StTL formulas. The *bounded until* can be expressed in terms of the *unbounded until* as $\varphi \mathbf{U}_{[a,b]} \psi = \mathbf{G}_{[0,a]}\varphi \wedge \mathbf{F}_{[a,b]}\psi \wedge \mathbf{F}_{[a,a]}\big(\varphi \mathbf{U} \psi\big)$. Therefore, it suffices to encode the *unbounded until*. To this end, we introduce the following set

$$g_t^{z}(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : g_t^{\varphi}(\theta) \leq 0 \text{ and } g_{t+1}^{\varphi \mathbf{U} \psi}(\theta) \leq 0\} \tag{11}$$

and encode the unbounded until as follows

$$g_t^{\varphi \mathbf{U} \psi}(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : \tag{12}$$

$$b^{\psi} g_t^{\psi}(\theta) \leq 0,\ b^{z} g_t^{z}(\theta) \leq 0,\ b^{\psi} + b^{z} \geq 1\}, \tag{13}$$

for $t = 1, \ldots, T-1$ and $g_T^{\varphi \mathbf{U} \psi}(\theta) = \{\theta \in \mathbb{R}^{d_\theta} : g_T^{\psi}(\theta) \leq 0\}$.

Our method given in Algorithm 1 leverages Eq. (6-13) to encode an arbitrary formula by recursively generating a set of mixed-integer linear constraints on the parameters θ of the control policy. We effectively use three types of constraints: intersections of linear constraints (conjunction), unions of linear constraints (disjunction) and linear binary constraints that control which of the aforementioned linear constraints are active when taking their union. Our approach differs from the encoding presented in [24] because we encode predicates via linear constraints on the parameters of the control policy, whereas the later one enforces a constraint on an introduced binary variable. The benefit of our approach is that no binary variables are necessary to encode negations, conjunctions and the globally operator (i.e., conjunction is encoded as the intersection of constraints). Our approach requires *exactly* $N_d + (N_u + 3)T$ binary variables, where N_d is the number of disjunction operators, and N_u the number of until operators. This is substantially better than the $\mathcal{O}(T|P|)$ binary variables required by [24], where $|P|$ is the cardinality of the predicate set. Our algorithm essentially reduces all operations to N_c conjunctions, N_d disjunctions and introduces integer variables only for the later ones.

Let $g_i^c(\theta)$ and $g_{jk}^d(\theta)$ be the (linear) constraints recursively generated by Algorithm 1. The former ones correspond to conjunctions while the later ones to disjunctions. We reformulate Problem 1 as follows

$$\max_{\theta, b_{jk}} \quad \min_{\lambda \geq 0} \quad \mathcal{L}(\lambda; \pi_\theta, \mathcal{D}) \tag{14a}$$

$$\theta \in g_i^c(\theta),\ i = 1, \ldots, n_c \tag{14b}$$

$$\theta \in \bigcup_{k=1}^{n_b} b_{jk} g_{jk}^d(\theta),\ j = 1, \ldots, n_d \tag{14c}$$

$$\sum_{k=1}^{n_b} b_{jk} \geq 1,\ j = 1, \ldots, n_d \tag{14d}$$

Algorithm 1: Recursive Constraint Generation

Inputs: Formula φ of maximum horizon T, ordered set P of predicates in φ, set $\{C^\mu\}_{\mu \in P}$ of linear constraints

procedure `RecConGen` $(\varphi, \{C^\mu\}_{\mu \in P})$**:**
  switch φ **do**
    case μ_t ***or*** $\neg \mu_t$ **do**
      return C^μ
    end
    case $\bigwedge_{i=1}^{m} \varphi_i$ **do**
      $g_t^i(\theta) \leftarrow$ `RecConGen` $(\varphi_i, \{C^\mu\}_{\mu \in P})$
      $g_t^{\wedge}(\theta) \leftarrow \{\theta \in \mathbb{R}^{d_\theta} : \theta \in \bigcap_{i=1}^{m} g_t^i(\theta)\}$
      return $g_t^{\wedge}(\theta),\ t = 1, \ldots, T$
    end
    case $\bigvee_{i=1}^{m} \varphi_i$ **do**
      $g_t^i(\theta) \leftarrow$ `RecConGen` $(\varphi_i, \{C^\mu\}_{\mu \in P})$
      $g_t^{\vee}(\theta) \leftarrow \{\theta \in \mathbb{R}^{d_\theta} : \theta \in b_t^i g_t^i(\theta),$
       $i = 1, 2, \ldots, m \text{ and } \sum_i b_t^i \geq 1\}$
      return $g_t^{\vee}(\theta),\ t = 1, \ldots, T$
    end
    case $\varphi \mathbf{U}_{[a,b]} \psi$ **do**
      $g_t^{\varphi}(\theta) \leftarrow$ `RecConGen` $(\varphi, \{C^\mu\}_{\mu \in P})$
      $g_t^{\psi}(\theta) \leftarrow$ `RecConGen` $(\psi, \{C^\mu\}_{\mu \in P})$
      $g_t^{\varphi \mathbf{U} \psi}(\theta) \leftarrow$ Eq. (13)
      return $g_t^{\varphi \mathbf{U} \psi}(\theta),\ t = 1, \ldots, T$
    end
  end

where $\mathcal{L}(\lambda; \pi, \mathcal{D}) = H^\beta(\pi) - \lambda^T(f_r(\pi) - \hat{f}_r(\mathcal{D}))$ is the *Lagrangian*, $b_{jk} \in \{0, 1\}$. Observe that we use Lagrangian duality to relax the feature expectation matching constraint (1b). This form is convenient because the resulting problem has only mixed-integer linear constraints and can be solved using standardized solvers. We note that, even though our framework is not restrictive to finite horizon formulas, we impose an upper bound of T to the horizon of φ. The number of constraints is proportional to T and, if the horizon is infinite, we have infinite number of linear constraints, which requires more sophisticated techniques to solve [25]. The issue is resolved by upper-bounding the time horizon. Finally, it is also worth noting that for the special case of a formula φ that consists of conjunctions and global operators, our method does not introduce any binary variables and Problem 14 has only linear constraints.

IV. Cut and Generate Method

In the previous section, we assumed that the constraint given by Eq. (5) is encoded via a set of linear constraints and used this approximation to recursively generate constraints for φ. In this section, we show how to leverage the recursive constraint generation given in Algorithm 1 to generate linear constraints for any formula. The fundamental idea to iterate between using cutting hyperplanes to approximate the feasible region of Eq. (5) and generating constraints for the entire formula using the obtained hyperplanes. We

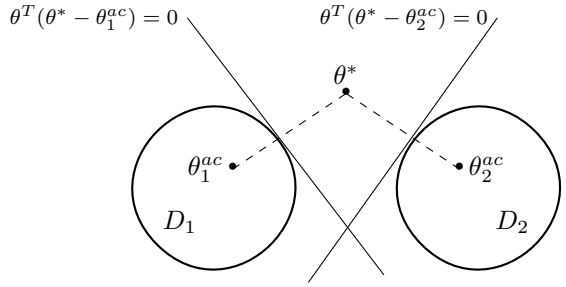

Fig. 1: Hyperplane approximation for the formula $\varphi = \mu_1 \vee \mu_2$. The feasible region of φ is $D_1 \cup D_2$ and is approximated by $\{\theta \in \mathbb{R}^2 : \theta^T(\theta^* - \theta_1^{ac}) \leq 0 \ \vee \ \theta^T(\theta^* - \theta_2^{ac}) \leq 0\}$, i.e., the union of the two hyperplanes. The union is non-convex and a binary variable is introduced that controls which constraint is active.

call this approach cut-and-generate (Algorithm 2). We start by initializing the set of constraints C to the empty set (Line 1) a set C^μ that keeps track of the linear constraints generated for each predicate μ (Line 2). Then, we solve the unconstrained Problem 14a (i.e., the Lagrange dual with the feature expectation matching constraint relaxed) to obtain an initial solution θ^* (Line 3). Next, we check if θ^* satisfies the specification φ. This is achieved by evaluating Eq. (5) at θ^* for each predicate in φ and propagating the resulting truth values through the StTL schematics.

If θ^* does not satisfy φ a new, more constrained problem is constructed. To this end, we use θ^* and a point that belongs to the feasible space of Eq. (5) in order to construct a cutting hyperplane as a linear approximation of that feasible space (see Fig. 1 for an illustration). That hyperplane introduces a new linear constraint to the problem and eliminates a half-plane from the search space. Intuitively, that point should be centered; points close to the boundary eliminate "less" space and may require more iterations. In the convex optimization literature, several centering methods have been proposed [26]. Due to its simplicity, we choose the analytic center, which is found by solving the following unconstrained minimization

$$\theta_t^{ac} = \underset{\theta \in \mathbb{R}^{d_\theta}}{\arg\min} \log c_t^\mu(\theta), \tag{15}$$

Observe that the analytic center corresponds to the minimization of the logarithmic barrier function. To find the minimum, we first calculate the gradient of Eq. (5):

$$\nabla c_t^\mu(\theta) = \sum_{s',a} q_c(s',a)\Big(p_{t-1}(s')\nabla\pi_\theta(a|s') + \pi_\theta(a|s')\nabla p_{t-1}(s')\Big), \tag{16}$$

where $q_c(s',a)$ is given by Eq. (4). The gradient of $p_t(s)$ can be found by differentiating Eq. (2) and is given by the following recursive equation:

$$\nabla p_t(s) = \sum_{s',a} \delta(s|s',a)\Big(p_{t-1}(s')\nabla\pi_\theta(a|s') + \pi_\theta(a|s')\nabla p_{t-1}(s')\Big). \tag{17}$$

Algorithm 2: Cut and Generate Algorithm

Inputs : MDP $\mathcal{M} \setminus \mathcal{R}$, demonstrations $\mathcal{D}$, parameterized policy $\pi_\theta(a|s)$, formula φ of maximum horizon T, ordered set P of atomic predicates in φ

Output: Optimal policy π_{θ^*}

1 $C \leftarrow \{\}$
2 $C^\mu \leftarrow \{\}$, for all predicates $\mu \in P$
3 $\theta^* \leftarrow \arg\max_\theta \min_{\lambda \geq 0} \mathcal{L}(\lambda; \pi_\theta, \mathcal{D})$
4 **while** θ^* *is not feasible* **do**
5 **forall** *predicates* μ *in* P **do**
6 $\theta_t^{ac} \leftarrow \arg\min \log c_t^\mu(\theta)$
7 $g_t^\mu(\theta) \leftarrow \{\theta \in \mathbb{R}^{d_\theta} : \theta^T(\theta^* - \theta_t^{ac}) \leq 0\}$
8 $C^\mu \leftarrow \{\theta \in \mathbb{R}^{d_\theta} : \theta \in g_t^\mu(\theta) \wedge \theta \in g(\theta), \ \forall g(\theta) \in C^\mu\}$
9 **end**
10 $C \leftarrow C \cup$ `RecConGen` $(\varphi, \{C^\mu\}_{\mu \in P})$
11 $\Theta \leftarrow \{\theta \in \mathbb{R}^{d_\theta} : g(\theta), \forall g(\theta) \in C\}$
12 $\theta^* \leftarrow \arg\max_{\theta \in \Theta} \min_{\lambda \geq 0} \mathcal{L}(\lambda; \pi_\theta, \mathcal{D})$
13 **end**

Even though Eq. (15) is not convex for practical policy parameterizations (e.g., neural networks), we leverage the gradient given by Eq. (16) to find a local minimum via standard gradient methods. Also, note that we need to find the analytic center only once for each predicate (i.e., we can pre-solve Eq. (15) for speed). Following that, we use the vector $\theta^* - \theta_t^{ac}$ to construct a cutting hyperplane and introduce this constraint to the problem (Lines 6-8, the "cut" step). Then, we use RECCONGEN algorithm to recursively generate constraints for the entire formula (Line 10, the "generate" step) and add those to the problem (Line 11). Observe that the constraints that define the set Θ (Line 11) contain integer variables, which are created by the RECCONGEN algorithm. Finally, we solve the newly constructed problem (Line 12) and iterate. The main benefit of our approach is that we approximate the feasible region of predicates (Eq. (5)) via linear constraints and then propagate these linear constraints through the formula. This allows us to approximate the feasible region of the problem via mixed-integer linear constraints. Note that if we used directly the non-linear constraint given by Eq. (5) that would result in mixed-integer non-linear constraints, which is far more difficult to solve.

V. EXPERIMENTAL EVALUATION

Single-Goal GridWorld (SGGW). Consider an agent moving in an $N \times N$ GridWorld enviroment. The agent starts deterministically from the bottom-left corner ($s = (0,0)$) and receives a reward equal to 1 at the terminal state (upper-right corner, $s = (N-1, N-1)$). When the agent takes an action, it transits to the correct next state with probability $1 - p_{slip}$ and with probability p_{slip} it "slips", i.e., it arrives at one of the remaining 3 states with equal probability. We assume that there exists a "hole" state and an "attractor" state, denoted

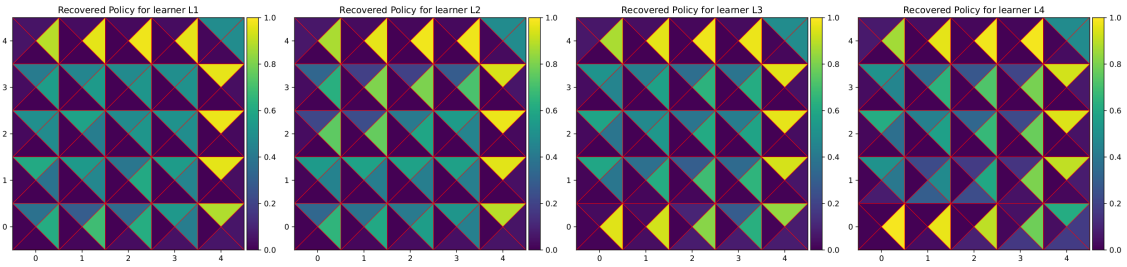

Fig. 2: Recovered optimal stochastic policies for L1-L4. The agent starts from $(0, 0)$ and receives a reward equal to 1 when it reaches $(4, 4)$. The color-map of each triangle in the corresponding rectangle denotes the probability of taking the underlying action.

with H and A, respectively. We consider the following StTL specifications:

$$\varphi_1 = \mathbf{G}_{[0,T]}\big(P(d(s_t, H) \leq c_1) \leq \epsilon_1\big) \tag{18}$$

$$\varphi_2 = \mathbf{F}_{[0,T]}\big(P(d(s_t, A) \leq c_2) \geq 1 - \epsilon_2\big) \tag{19}$$

where $d(\cdot, \cdot)$ denotes the distance function on the grid. The hole is placed on the upper-left corner ($H = (0, N-1)$) and the attractor on the bottom-right ($A = (N-1, 0)$). We choose a softmax policy and the remaining parameters are as follows: $N = 5, T = 10, \epsilon_1 = \epsilon_2 = 0.1,\ c_1 = 1, c_2 = 0$. We consider 4 different learner models: a) an unconstrained learner (L1). This is essentially the standart MCE inverse reinforcement learning framework (i.e., Problem 1 without the StTL constraint) and acts as a baseline, b) a learner constrained by φ_1 (L2): This learner must always stay away from the hole state with the given probability threshold, c) a learner constrained by φ_2 (L3): This learner must eventually get sufficiently close to the attractor state and d) a learner constrained by $\varphi_1 \wedge \varphi_2$ (L4): This learner is constrained by the conjunction of φ_1 and φ_2, i.e., it must stay away from the hole and approach the attractor. The results are shown in Fig. 2. Observe that L2 tends to "push" the agent away from the upper-left corner; L3 tends to "attract" the agent to the bottom-right corner; and L4 combines both.

Multi-Goal GridWorld (MGGW). This enviroment is similar to the previous one, albeit the transitions are deterministic, there exist obstacles and multiple goal states. We consider the following formulas:

1) Obstacle avoidance: $\varphi_3 = \mathbf{G}_{[0,T]}\big(P(d(s_t, \texttt{obs}) \leq c_3) \leq \epsilon_3\big)$, where $d(s_t, \texttt{obs})$ is the distance of the agent from the closest obstacle.
2) Multi-Goal reach: The agent is required to reach either one of goal states, i.e., $\varphi_4 = \mathbf{F}_{[0,T]}\big(P(d(s_t, \texttt{g1}) = c_4) \geq 1 - \epsilon_4\big) \vee \mathbf{F}_{[0,T]}\big(P(d(s_t, \texttt{g2}) = c_5) \geq 1 - \epsilon_5\big)$.

The imposed specification is $\varphi_3 \wedge \varphi_4$. To generate expert trajectories, we find the optimal policy using value iteration and then run it in the enviroment. The parameters are as follows: $N = 8, T = 16, c_3 = 1, c_4 = c_5 = 0, \epsilon_3 = \epsilon_4 = \epsilon_5 = 0.05$. The results are shown in Fig. 3.

Robustness. To quantify the degree of satisfaction of each of the aforementioned formulas over the policies learned by L1-L4, we use the metric of StTL robustness. For a more formal treatment of StTL robustness, the reader may consult [21]. Informally, high positive robustness indicates "better"

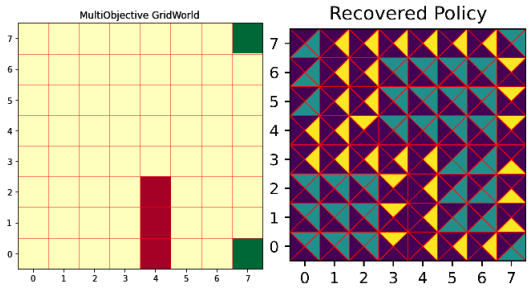

Fig. 3: The multi-goal gridworld enviroment along with the policy recovered under specification $\varphi_3 \wedge \varphi_4$.

satisfaction, negative robustness denotes that the formula is not satisfied and robustness value equal to zero denotes marginal satisfaction. For the multi-goal gridworld case, we consider 4 different learners, similarly to the SGGW. The results are shown in Tables I and II. Observe that each formula attains the higher robustness over the policy which is trained for that formula, i.e., φ_1 attains higher robustness for learner L1, which is trained under φ_1. Notice that the cross terms (e.g., φ_1 under L3) attain slightly positive robustness values. This is explainable due to the nature of the specifications and the enviroment. For instance, if the agent avoids the hole, then it is more inclined to be closer to the attractor. Finally, observe that L1 attains negative robustness for all formulas, which indicates that the standard MCE does not suffice to satisfy the specifications.

Scalability and Time Complexity. To demonstrate the applicability of our method in problems with larger state spaces, we perform a scalability analysis. In more detail, we define the metric *overhead* as the run-time of Algorithm 2 subtracted by the run-time of the MCE algorithm (line 12). The reason for defining that metric is that our method essentially builds on top of the MCE algorithm, which we use as a black box and have no influence over its run-time. Therefore, this metric essentially captures the number of successive refinements in the hyperplane approximation. We compute the overhead for both SGGW and MGGW for different grid sizes. In the MGGW case, the size of the obstacle increases with the size of the grid. We consider formulas $\varphi_1 \wedge \varphi_2$ and $\varphi_3 \wedge \varphi_4$, respectively, and the time horizon is set to $2N$, where N is the size of the grid. The result shown in Fig. 4 indicates a linear dependence of the time overhead on the size of the grid for both enviroments, which is tolerable overhead given the problem complexity.

TABLE I: Single-Goal GridWorld

	L1 (MCE)	L2	L3	L4
φ_1	-0.44	1.98	0.10	0.15
φ_2	-0.25	0.07	1.44	0.28
$\varphi_1 \wedge \varphi_2$	-0.23	0.12	0.25	1.07

TABLE II: Multi-Goal GridWorld

	L1 (MCE)	L2	L3	L4
φ_3	-0.14	1.24	0.18	0.25
φ_4	-0.35	0.15	1.58	0.13
$\varphi_3 \wedge \varphi_4$	-0.25	0.17	0.24	1.21

Robustness of the underlying StTL formulas with respect to policies learned by learners L1-L4. Higher robustness is better and negative implies that the formula is not satisfied.

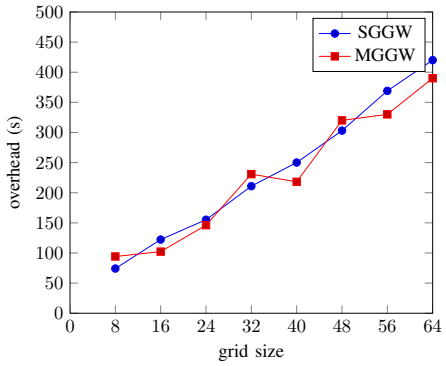

Fig. 4: Empirical complexity estimation

VI. Summary

We address the problem of learning from demonstrations when the agent must satisfy a set of specifications expressed as Stochastic Temporal Logic (StTL) formulas. Building on the maximum causal entropy inverse reinforcement learning framework, our method uses cutting hyperplanes to approximate the feasible region of StTL predicates and propagates these hyperplanes through the StTL schematics to generate constraints for arbitrary formulas. This results in a set of mixed-integer linear constraints that encode the satisfaction of the specification. We validated the practical usability of our approach in different enviroments and specifications.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[2] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, p. 4950–4957, AAAI Press, 2018.

[3] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *CoRR*, vol. abs/1011.0686, 2010.

[4] B. D. Ziebart, *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, USA, 2010.

[5] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*, pp. 663–670, Morgan Kaufmann, 2000.

[6] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, (San Francisco, CA, USA), p. 2586–2591, Morgan Kaufmann Publishers Inc., 2007.

[7] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, p. 1433–1438, AAAI Press, 2008.

[8] J. V. Leeuwen, Warwick, A. R. Meyer, and M. Nival, *Handbook of Theoretical Computer Science: Algorithms and Complexity*. Cambridge, MA, USA: MIT Press, 1990.

[9] C. Fritz, "Constructing büchi automata from linear temporal logic using simulation relations for alternating büchi automata," in *CIAA*, 2003.

[10] A. Donzé, "On signal temporal logic," in *Runtime Verification* (A. Legay and S. Bensalem, eds.), (Berlin, Heidelberg), pp. 382–383, Springer Berlin Heidelberg, 2013.

[11] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, (New York, NY, USA), p. 239–248, Association for Computing Machinery, 2015.

[12] D. Kasenberg and M. Scheutz, "Interpretable apprenticeship learning with temporal logic specifications," *CoRR*, vol. abs/1710.10532, 2017.

[13] F. Memarian, Z. Xu, B. Wu, M. Wen, and U. Topcu, "Active task-inference-guided deep inverse reinforcement learning," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1932–1938, 2020.

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

[15] M. Wen, I. Papusha, and U. Topcu, "Learning from demonstrations with high-level side information," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3055–3061, 2017.

[16] W. Zhou and W. Li, "Safety-aware apprenticeship learning," 2018.

[17] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Robotics Science and Systems*, 2016.

[18] S. Jha, V. Raman, D. Sadigh, and S. A. Seshia, "Safe autonomy under perception uncertainty using chance-constrained temporal logic," *J. Autom. Reason.*, vol. 60, pp. 43–62, Jan. 2018.

[19] J. Li, P. Nuzzo, A. Sangiovanni-Vincentelli, Y. Xi, and D. Li, "Stochastic contracts for cyber-physical system design under probabilistic requirements," in *ACM/IEEE Int. Conf. on Formal Methods and Models for System Design*, 2017.

[20] J. V. Deshmukh, P. Kyriakis, and P. Bogdan, "Stochastic temporal logic abstractions: Challenges and opportunities," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 3–16, Springer, 2018.

[21] P. Kyriakis, J. V. Deshmukh, and P. Bogdan, "Specification mining and robust design under uncertainty: A stochastic temporal logic approach," *ACM Trans. Embed. Comput. Syst.*, vol. 18, Oct. 2019.

[22] A. G. Puranic, J. V. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic," *CoRR*, vol. abs/2102.07730, 2021.

[23] M. Bloem and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," in *53rd IEEE Conference on Decision and Control*, pp. 4911–4916, 2014.

[24] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, pp. 81–87, 2014.

[25] G. STILL, "Optimization problems with infinitely many constraints," *Buletinul ştiinţific al Universitatii Baia Mare, Seria B, Fascicola matematică-informatică*, vol. 18, no. 2, pp. 343–354, 2002.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.