Towards Specification Learning from Demonstrations

Ankit Shah
CSAIL
MIT
Cambridge, MA 02139
Email: ajshah@mit.edu

Julie Shah
CSAIL
MIT
Cambridge, MIT
Email: julie_a_shah@csail.mit.edu

***Abstract*—When observing task demonstrations, human apprentices are able to identify whether a given task is executed correctly long before they gain expertise in actually performing that task. Prior research into learning from demonstrations (LfD) has failed to capture this notion of the acceptability of an execution; meanwhile, temporal logics provide a flexible language for expressing task specifications. Inspired by this, we present a probabilistic model for inferring task specification as a temporal logic formula. We incorporate methods from probabilistic programming to define our priors, along with a domain-independent likelihood function to enable sampling-based inference. We demonstrate the efficacy of our model for inferring specifications with over 90% similarity between the inferred specification and the ground truth, both within a synthetic domain and a real-world table setting task.**

I. Introduction

Imagine showing a friend how to play your favorite quest-based video game. A mission within such a game might be composed of multiple sub-quests that must be completed in order to complete that level. In this scenario, it is likely that your friend would comprehend what needs to be done in order to complete the mission well before he or she was actually able to play the game effectively. While learning from demonstrations, human apprentices can identify whether a task is executed correctly well before gaining expertise in that task. Most current approaches to learning from demonstration frame this problem as one of learning a reward function or policy within a Markov decision process setting; however, user specification of acceptable behaviors through reward functions and policies remains an open problem [3]. Temporal logics have been used in prior research as a language for expressing desirable system behaviors, and can improve the interpretability of specifications if expressed as compositions of simpler templates (akin to those described by Dwyer et al. [5]). In this work, we propose a probabilistic model for inferring the temporal structure of a task as a linear temporal logic (LTL) specification.

A specification inferred from demonstrations is valuable in conjunction with synthesis algorithms for verifiable controllers ([18] and [25]), as a reward signal during reinforcement learning ([20], [21]), and as a system model for execution monitoring. In our work, we frame specification learning as a Bayesian inference problem.

The flexibility of LTL for specifying behaviors also represents a key challenge with regard to inference due to a large hypothesis space. We define prior and likelihood distributions over a smaller but relevant part of the LTL formulas, using templates based on work by Dwyer et al [5]. Ideas from universal probabilistic programming languages formalized by Freer et al [8] and Goodman et al [9], [10] are key to our modeling approach. Indeed, probabilistic programming languages enabled Ellis et al [7], [6] to perform inference over complex, recursively defined hypothesis spaces of graphics programs and pronunciation rules. We demonstrate the capability of our model to achieve greater than 90% similarity between the ground truth specification and the inferred specification, both within a synthetic domain and a real-world task of setting a dinner table.

II. Related Work

The surveys by Argall et al. [2] and Chernova et al. [4] provide a comprehensive review of techniques built on these works as applied to robotics. One common approach in prior research frames learning from demonstration as an inverse reinforcement learning (IRL) problem. Ng et al [22] and Abbeel et al [1] first formalized the problem of inverse reinforcement learning as one of optimization in order to identify the reward function that best explains observed demonstrations. Ziebart et al [30] introduced algorithms to compute optimal policy for imitation using the maximum entropy criterion. Konidaris et al [17] and Niekum et al [23] framed IRL in a semi-Markov setting, allowing for an implicit representation of the temporal structure of the task. However, according to Arnold et al [3], one drawback of inverse reinforcement learning is that it is non-trivial to extract task specifications from a learned reward function or policy. Our method bridges this gap by directly learning the specifications for acceptable execution of the given task.

A second common approach involves learning objectives of the task. Hayes and Scassellati [11] Scassellati learn the temporal structure of the actions in the task as a hierarchical task network. Toris et al. [28] adopted an unsupervised clustering approach to model the final positions of the domain objects using crowd-sourced goal configurations. However, these approaches do not generalize to all types of task

specification. Using temporal logics, our approach is capable of encoding both these notions along with a larger range of temporal behaviors expressible in LTL.

Temporal logics, introduced by Pnueli [24], are an expressive grammar used to describe the desirable temporal properties of task execution. Temporal logics have previously been used as a language for goal definitions in reinforcement learning algorithms ([20], [21]), reactive controller synthesis ([18], [25]), and domain independent planning [14].

Kasenberg and Scheutz [13] explored mining globally persistent specifications from optimal traces of a finite state Markov decision process (MDP). Jin et al [12] proposed algorithms for mining temporal specifications similar to rise time and setting time for closed-loop control systems. Works by Kong et al [15], [16] and Yoo and Belta [29], are most closely related to our own, as our work incorporates only the observed state variable (and not the actions of the demonstrators) as input to the model. Kong et al [15], [16] and Yoo and Belta [29] mined PSTL specifications for given demonstrations while simultaneously inferring signal propositions akin to our own user-defined atomic propositions by conducting breadth first search over a directed acyclic graph (DAG) formed by candidate formulas. Our prior specifications allow for better connectivity between different formulas, while using MCMC-based approximate inference allows for fixed runtimes. We adopt a fully Bayesian approach to model the inference problem, enabling our model to maintain a posterior distribution over candidate formulas. This distribution provides a measure of confidence when predicting the acceptability of a new demonstration that the aforementioned approaches do not.

III. Linear Temporal Logic

Linear temporal logic (LTL), introduced by Pneuli [24], provides an expressive grammar for describing temporal behaviors. A LTL formula is composed of atomic propositions (discrete time sequences of Boolean literals) and both logical and temporal operators, and is interpreted over traces $[\boldsymbol{\alpha}]$ of the set of propositions $\boldsymbol{\alpha}$. The notation $[\boldsymbol{\alpha}], t \models \varphi$ indicates that φ holds at time t. The trace $[\boldsymbol{\alpha}]$ satisfies φ (denoted as $[\boldsymbol{\alpha}] \models \varphi$) iff $[\boldsymbol{\alpha}], 0 \models \varphi$. The minimal syntax of LTL can be described as follows:

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi_1 \mid \varphi_1 \mathbf{U} \varphi_2 \tag{1}$$

p is an atomic proposition; φ_1 and φ_2 are valid LTL formulas. The operator $\mathbf{X}$ is read as 'next' and $\mathbf{X}\varphi_1$ evaluates as true at time t if φ_1 evaluates to true at $t+1$. The operator $\mathbf{U}$ is read as 'until' and the formula $\varphi_1 \mathbf{U} \varphi_2$ evaluates as true at a time t_1 if φ_2 evaluates as true at some time $t_2 > t_1$ and φ_1 evaluates as true for all time steps t such that $t_1 \leq t \leq t_2$. In addition to the minimal syntax, we also use the additional first order logic operators $\wedge$ (and) and $\mapsto$ (implies), as well as other higher-order temporal operators, $\mathbf{F}$ (eventually) and $\mathbf{G}$ (globally). $\mathbf{F}\varphi_1$ evaluates to true at t_1 if φ_1 evaluates as true for some $t \geq t_1$. $\mathbf{G}\varphi_1$ evaluates to true at t_1 if φ_1 evaluates as true for all $t \geq t_1$.

IV. Bayesian Specification Inference

A large number of tasks comprised of multiple subtasks can be represented by a combination of three temporal behaviors among those defined by Dwyer et al [5] namely, global satisfaction of a proposition, eventual completion of a subtask, and temporal ordering between subtasks. With φ_{global}, $\varphi_{eventual}$, and φ_{order} representing LTL formulas for these behaviors, the task specification is written as follows:

$$\varphi = \varphi_{global} \wedge \varphi_{eventual} \wedge \varphi_{order} \tag{2}$$

We represent the task demonstrations as an observed sequence of state variables, $\boldsymbol{x}$. Let $\boldsymbol{\alpha} \in \{0,1\}^n$ represent a vector of a finite dimension formed by Boolean propositions. $\boldsymbol{\alpha} = f(\boldsymbol{x})$ (i.e., the propositions) are a function of the state variables of the system at a given time instant. The output of specification learning is a formula, $\varphi \in \boldsymbol{\varphi}$, that best explains the demonstrations, where $\boldsymbol{\varphi}$ is the set of all formulas satisfying the template described in Equation 2.

A. Formula Template

Global satisfaction: Let $\mathbf{T}$ be the set of candidate propositions to be globally satisfied, and let $\boldsymbol{\tau} \subseteq \boldsymbol{T}$ be the actual subset of propositions globally satisfied. The LTL formula that specifies this behavior is written as follows:

$$\varphi_{global} = \left(\bigwedge_{\tau \in \boldsymbol{\tau}} (\mathbf{G}(\tau)) \right) \tag{3}$$

Such formulas are useful for specifying that some constraints must always be met for example, a robot avoiding collisions during motion, or an aircraft avoiding no-fly zones.

Eventual completion: Let $\boldsymbol{\Omega}$ be the set of all candidate subtasks, and let $\boldsymbol{W_1} \subseteq \boldsymbol{\Omega}$ be the set of subtasks that must be completed if the conditions represented by $\pi_w; w \in \boldsymbol{W_1}$ are met. ω_w are propositions representing the completion of a subtask. The LTL formula that specifies this behavior is written as follows:

$$\varphi_{eventual} = \left(\bigwedge_{w \in \mathbf{W_1}} (\pi_w \rightarrow \mathbf{F}\omega_w) \right) \tag{4}$$

Temporal ordering: Every set of feasible ordering constraints over a set of subtasks is mapped to a directed acyclic graph (DAG) over nodes representing these subtasks. Each edge in the DAG corresponds to a binary precedence constraint. Let $\boldsymbol{W_2}$ be the set of binary temporal orders defined by $\boldsymbol{W_2} = \{(w_1, w_2) : w_1 \in \boldsymbol{V}, w_2 \in \text{Descendants}(w_1)\}$, where $\boldsymbol{V}$ is the set of all nodes in the task graph. Thus, the ordering constraints include an enumeration of not just the edges in the task-graph, but all descendants of a given node. For subtasks w_1 and w_2, the ordering constraint is written as follows:

$$\varphi_{order} = \left(\bigwedge_{(w_1, w_2) \in \mathbf{W_2}} (\pi_{w_1} \rightarrow (\neg\omega_{w_2} \mathbf{U} \omega_{w_1})) \right) \tag{5}$$

This formula states that if conditions for the execution of w_1 i.e. π_{w_1} are satisfied, w_2 must not be completed until w_1 has been completed.

For the purposes of this paper, we assume that all required propositions $\boldsymbol{\alpha} = [\boldsymbol{\tau}, \boldsymbol{\pi}, \boldsymbol{\omega}]^T$ and labeling functions $f(\boldsymbol{x})$ are known, along with the sets $\boldsymbol{T}$ and $\boldsymbol{\Omega}$, and the mapping of the condition propositions π_w to their subtasks. Under these assumptions, the problem of inferring the correct formula for a task is equivalent to identifying the correct subsets $\boldsymbol{\tau}$, $\boldsymbol{W_1}$, and $\boldsymbol{W_2}$, which explain the observed demonstrations well.

B. Specification learning as Bayesian Inference

The Bayes theorem is fundamental to the problem of inference, and is stated as follows:

$$P(h \mid \mathbf{D}) = \frac{P(h)P(\mathbf{D} \mid h)}{\sum_{h \in \mathbf{H}} P(h)P(\mathbf{D} \mid h)} \tag{6}$$

$P(h)$ is the prior distribution over the hypothesis space, and $P(\mathbf{D} \mid h)$ is the likelihood of observing the data given a hypothesis. Our hypothesis space is defined by $\boldsymbol{H} = \boldsymbol{\varphi}$, where $\boldsymbol{\varphi}$ is the set of all formulas that can be generated by the production rule defined by the template in Equation 2. The observed data comprises the set of demonstrations provided to the system by expert demonstrators (note that we assume all these demonstrations are acceptable). $\boldsymbol{D}$ represents a set of sequences of the propositions, defined by $\boldsymbol{D} = \{[\boldsymbol{\alpha}]\}$.

*1) **Prior specification**:* While sampling candidate formulas as per the template depicted in Equation 2, we treat the sub-formulas in Equations 3, 4, and 5 as independent to each other. As generating the actual formula, given the selected subsets, is deterministic, sampling φ_{global} and $\varphi_{eventual}$ is equivalent to selecting a subset of a given finite universal set. Given a set A, we define `SampleSubset`(A,p) as the process of applying a Bernoulli trial with success probability p to each element of A and returning the subset of elements for which the trial was successful. Thus, sampling φ_{global} and $\varphi_{eventual}$ is accomplished by performing `SampleSubset`$(\boldsymbol{T}, p_G)$ and `SampleSubset`$(\boldsymbol{\Omega}, p_E)$. Sampling φ_{order} is equivalent to sampling a DAG, with the nodes of the graph representing subtasks. Based on domain knowledge, appropriately constraining the DAG topologies would result in better inference with fewer demonstrations. Here, we present two possible methods of sampling a DAG, with different restrictions on the graph topology.

Algorithm 1 SampleSetsOfLinearChains

```
1: function SAMPLESETSOFLINEARCHAIN(Ω, p_part)
2:     i ← 1; C_i ← []
3:     P ← random permutation(Ω)
4:     for a ∈ P do
5:         C_i.append(a)
6:         k ← Bernoulli(p_part)
7:         if k = 1 then
8:             i = i + 1; C_i ← []
9:     return C_j ∀ j
```

Linear chains: A linear chain is a DAG such that all subtasks must occur in a single, unique sequence out of all permutations. Sampling a linear chain is equivalent to selecting a permutation from a uniform distribution and is achieved via the following probabilistic program: for a set of size n, sample $n-1$ elements from that set without replacement, with uniform probability.

TABLE I: Prior definitions and hyperparameters.

Prior	φ_{Order}	Hyperparameters
Prior 1	`RandomPermutation`	p_G, p_E
Prior 2	`SampleSetsOfLinearChains`	p_G, p_E, p_{part}

Sets of linear chains: This graph topology includes graphs formed by a set of disjoint sub-graphs, each of which is either a linear chain or a solitary node. The execution of subtasks within a particular linear chain must be completed in the specified order; however, no temporal constraints exist between the chains. Algorithm 1 depicts a probabilistic program for constructing these sets of chains. In line 2, the first active linear chain is initialized as an empty sequence. In line 3, a random permutation of the nodes is produced. For each element $a \in \boldsymbol{P}$, line 5 adds the element to the last active chain. Lines 6 and 8 ensure that after each element, either a new active chain is initiated (with probability p_{part}) or the old active chain continues (with probability $1 - p_{part}$).

Two prior distributions based on the three probabilistic programs are described in Table I. In all the priors, φ_{global} and $\varphi_{eventual}$ are sampled using `SampleSubset`$(\boldsymbol{T}, p_G)$ and `SampleSubset`$(\boldsymbol{\Omega}, p_E)$, respectively.

*2) **Likelihood function**:* The likelihood distribution, $P(\{[\boldsymbol{\alpha}]\} \mid \varphi)$, is the probability of observing the trajectories within the data set given the candidate specification. It is reasonable to assume that the demonstrations are independent of each other; thus, the total likelihood can be factored as $P(\{[\boldsymbol{\alpha}]\} \mid \varphi) = \prod_{\{[\boldsymbol{\alpha}]\}} P([\boldsymbol{\alpha}] \mid \varphi)$.

The probability of observing a given trajectory demonstration is dependent upon the underlying dynamics of the domain and the characteristics of the agents producing the demonstrations. In the absence of this knowledge, our aim is to develop an informative, domain-independent proxy for the true likelihood function based only on the properties of the candidate formula; we call this the 'Complexity-based' (CB) likelihood function. Our approach is founded upon the classical interpretation of probability championed by Laplace [19], which involves computing probabilities in terms of a set of equally likely outcomes. Let there be N_{conj} conjunctive clauses in φ; there are then $2^{N_{conj}}$ possible outcomes in terms of the truth values of the conjunctive clauses. In the absence of any additional information, we assign equal probabilities to each of the potential outcomes. Then, according to the classical interpretation of probability, for candidate formula φ_1, defined by subsets $\boldsymbol{\tau_1}, \boldsymbol{W_{1_1}}$ and $\boldsymbol{W_{2_1}1}$; and φ_2, defined by subsets $\boldsymbol{\tau_2}, \boldsymbol{W_{1_2}}$, and $\boldsymbol{W_{2_2}}$, the likelihood odds ratio is defined as

follows:

$$\frac{P([\boldsymbol{\alpha}] \mid \varphi_1)}{P([\boldsymbol{\alpha}] \mid \varphi_2)} = \begin{cases} \frac{2^{N_{conj_1}}}{2^{N_{conj_2}}} = \frac{2^{|\boldsymbol{\tau_1}|+|\boldsymbol{W_{1_1}}|+|\boldsymbol{W_{2_1}}|}}{2^{|\boldsymbol{\tau_2}|+|\boldsymbol{W_{1_2}}|+|\boldsymbol{W_{2_2}}|}}, & [\boldsymbol{\alpha}] \models \varphi_2 \\ \frac{2^{N_{conj_1}}}{\epsilon} = \frac{2^{|\boldsymbol{\tau_1}|+|\boldsymbol{W_{1_1}}|+|\boldsymbol{W_{2_1}}|}}{\epsilon}, & [\boldsymbol{\alpha}] \not\models \varphi_2 \end{cases} \tag{7}$$

Here, a finite probability proportional to ϵ is assigned to a demonstration that does not satisfy the given candidate formula. With this likelihood distribution, a more restrictive formula with a low prior probability can gain favor over a simpler formula with higher prior probability given a large number of observations that would satisfy it. However, if the candidate formula is not the true specification, a larger set of demonstrations is more likely to include non-satisfying examples, thereby substantially decreasing the posterior probability of the candidate formula. The design of this likelihood function is inspired by the size principle described by Tenenbaum [27].

A second choice for a likelihood function, inspired by Shepard [26], is defined as the SIM model by Tenenbaum [27]. We call this the 'Complexity-independent' (CI) likelihood function, and it is defined as follows:

$$P([\boldsymbol{\alpha}] \mid \varphi) = \begin{cases} 1-\epsilon, & \text{if } [\boldsymbol{\alpha}] \models \varphi \\ \epsilon, & \text{Otherwise} \end{cases} \tag{8}$$

*3) **Inference**:* We implemented our probabilistic model in webppl [10], a Turing-complete probabilistic programming language. The hyperparameters, including those defined in Table I and ϵ, were set as follows: $p_E, p_G = 0.8$; $p_{part} = 0.3$; $N_{new} = 5$; $\epsilon = 4 \times log(2) \times (|\boldsymbol{T}| + |\boldsymbol{\Omega}| + 0.5|\boldsymbol{\Omega}|(|\boldsymbol{\Omega}| - 1))$. These values were held constant for all evaluation scenarios. The equation for ϵ was defined such that evidence of a single non-satisfying demonstration would negate the contribution of four satisfying demonstrations to the posterior probability. The posterior distribution of candidate formulas is constructed using webppl's Markov chain Monte Carlo (MCMC) sampling algorithm from 10,000 samples, with 100 samples used as burn-in. The posterior distribution is stored as a categorical distribution, with each possibility representing a unique formula. The maximum a posteriori (MAP) candidate represents the best estimate for the specification as per the model. The inference was run on a desktop with an Intel i7-7700 processor.

V. Evaluations

We evaluate our model on a synthetic domain where it is easy to generate demonstrations as per a known ground truth specification. This allows us to vary the ground truth specification and evaluate our model for different scenarios.

If the ground truth formula is defined using subsets $\boldsymbol{\tau}^*, \boldsymbol{W_1^*}$ and $\boldsymbol{W_2^*}$ as per Equations 3, 4, 5, and a candidate formula φ is defined by subsets $\boldsymbol{\tau}, \boldsymbol{W_1}$ and $\boldsymbol{W_2}$. We define the degree of similarity using the Jaccard index as follows:

$$L(\varphi) = \frac{|\{\boldsymbol{\tau}^* \cup \boldsymbol{W_1^*} \cup \boldsymbol{W_2^*}\} \cap \{\boldsymbol{\tau} \cup \boldsymbol{W_1} \cup \boldsymbol{W_2}\}|}{|\{\boldsymbol{\tau}^* \cup \boldsymbol{W_1^*} \cup \boldsymbol{W_2^*}\} \cup \{\boldsymbol{\tau} \cup \boldsymbol{W_1} \cup \boldsymbol{W_2}\}|} \tag{9}$$

The maximum possible value of $L(\varphi)$ is one when both formulas are equivalent. A key benefit of our approach is that we compute a posterior distribution over candidate formulas, thus we report the expected value of $\mathbb{E}(L(\varphi))$, as a measure of the deviation inferred distribution from the ground truth. We also report the max value of $L(\varphi)$ amongst the top-5 maximum aposteriori candidates. Finally we also classify the inferred orders in $\boldsymbol{W_2}$ as correct if they are included in the ground truth, incorrect if they reverse any constraint in the ground truth and extra otherwise (Extra orders overconstrain the problem but do not induce incorrect behaviors).

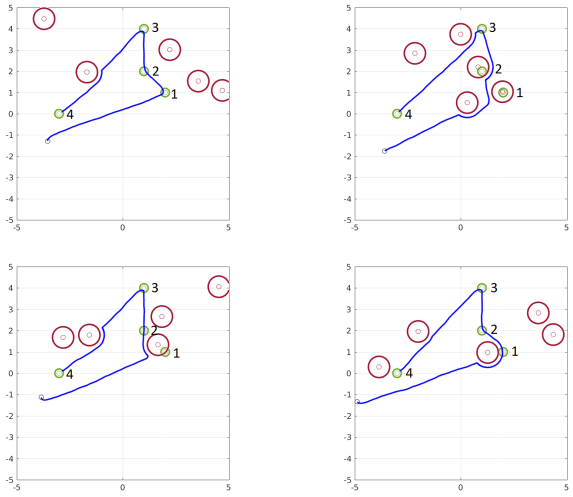
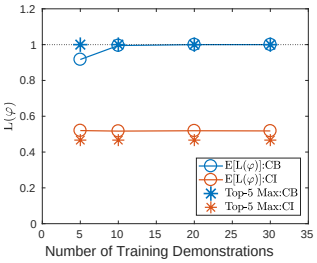


Fig. 1: Example trajectories from Scenario 1. The green circles denote the POIs and the red circles denote the threat avoidance zones.

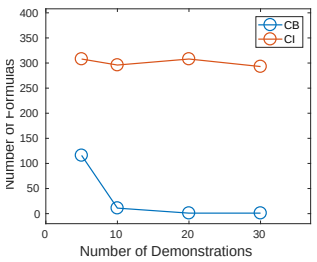
A. Synthetic Domain

In our synthetic domain, an agent navigates within a two-dimensional space that includes points of interest (POIs) to visit and threats to avoid. A predicate, ω_i, is associated with each POI and evaluates as true if the agent is within a tolerance region of the given POI. Each threat has a predicate, τ_i, associated with it, which evaluates as true if the agent enters an avoidance region for that threat. Finally, propositions π_i are associated with the accessibility of i^{th} POI, and evaluate as true if the given POI is not within the avoidance region of any threat. The agent is programmed to visit the accessible POIs and avoid threats, as per the ground truth specification.

In scenario 1 we generated example trajectories where the agent visits four POIs in a specific order $[1, 2, 3, 4]$. During each demonstration, five threat locations were sampled from a uniform distribution in the task space. Figure 1 depicts some of the demonstrated trajectories. The posterior distribution was computed by using prior 1 and both CB (Equation 7) and CI (Equation 8) likelihood functions for different training set sizes. The expected value and the maximum value amongst top-5 formula candidates of $L(\varphi)$ is depicted in Figure 2a. We observed that the CB likelihood function performs better than CI likelihood function at inferring the complete specification. Using the CI likelihood resulted in higher posterior probability

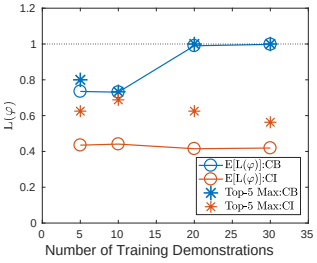


(a) Scenario 1 prior 1 $L(\varphi)$

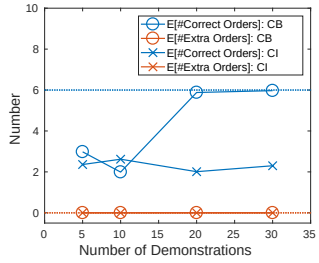


(b) Scenario 1 unique formulas

Fig. 2: Figure 2a depicts the $L(\varphi)$ values for Scenario 1, with the dotted line representing its maximum possible value. Figure 2b depicts the number of unique formulas in the posterior distribution.



(a) Scenario 1 prior 2 $L(\varphi)$



(b) Scenario 1 prior 2 orders

Fig. 3: Figure 3a depicts the $L(\varphi)$ values for Scenario 1 using prior 2, with the dotted line representing its maximum possible value. Figure 3b depicts the correct and extra orders inferred by the model. The dotted lines represent the ideal value.

assigned to formulas with high prior probability that are satisfied by all demonstrations. These tended to be simple non-informative formulas. Using the CB likelihood function assigned higher probability mass to more complex formulas that explain the demonstrations correctly. Figure 2b depicts the number of unique formulas in the posterior distributions. The CB likelihood function resulted in posteriors being more peaky, with fewer unique formulas as the training set size increases; this effect was not observed with the CI likelihood function. We also computed the posterior distribution using Prior 2 with both likelihood functions.

In Scenario 2, we incorporated five POIs where $[1, 3, 5]$ must be visited in that order and $\{2, 4\}$ can be visited in any order. Figure 4 The posterior distribution was computed using priors 2 as the ground truth specification does not lie in the support of prior 1. The expected value and the maximum value amongst top-5 formula candidates of $L(\varphi)$ is depicted in Figure 5a. With a large enough number of training examples, the CB likelihood function was able to infer the complete formula; with 10 or more training demonstrations, it returned the ground truth specification among the top 5 candidates with regards to the posterior distribution. On the other hand, the CI likelihood function failed to infer the complete specification. Figure 5b depicts the correct and extra orders inferred in Scenario 2. With a smaller training set size, both likelihood functions returned only a partial set of the correct orders, however, with more training demonstrations, the CB likelihood correctly inferred all three binary ordering constraints.

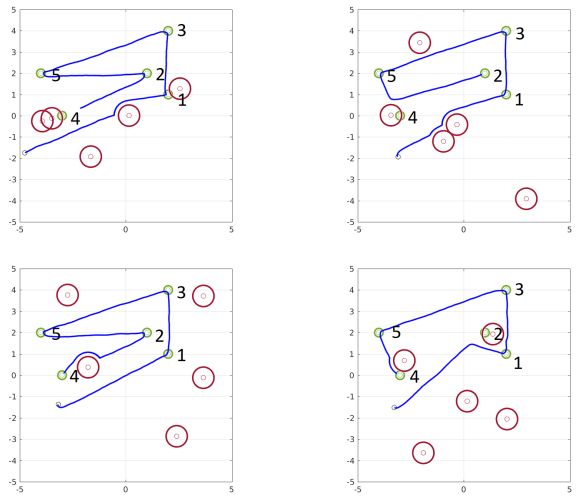
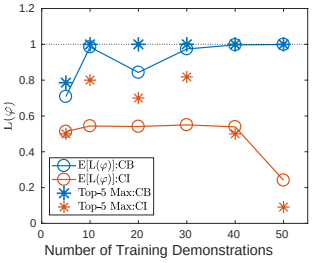
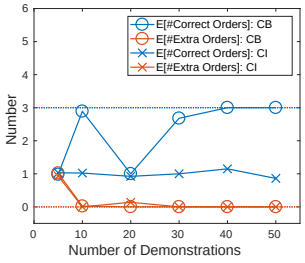


Fig. 4: Example trajectories from Scenario 2. Example trajectories from Scenario 1. The green circles denote the POIs and the red circles denote the threat avoidance zones.



(a) Scenario 2 prior 2 $L(\varphi)$



(b) Scenario 2 prior 2 orders

Fig. 5: Figure 5a depicts the $L(\varphi)$ values for Scenario 2 using prior 2. Figure 5a depicts the correct and extra orderings inferred by the model. The dotted lines represent the ideal value.

In Scenario 3, we incorporated five POIs $\{1, 2, 3, 4, 5\}$. Each of the POIs must be visited if accessible, however there were no constraints on the order in which they were visited. Figure 6 depicts the some of the example demonstrations. Again, the posterior distribution was computed using prior 2. The expected value and the maximum value among the top 5 formula candidates of $L(\varphi)$ is depicted in Figure 7. We observed that while the CB likelihood function inferred the correct formula with only 10 demonstrations, the CI likelihood function is unable to infer a formula with more than 10% similarity to the ground truth specification. As there are no ordering constraints among the POIs, the number of correct orders is zero. Figure 7b depicts the extra orders inferred by the model with both likelihood functions. It was observed that

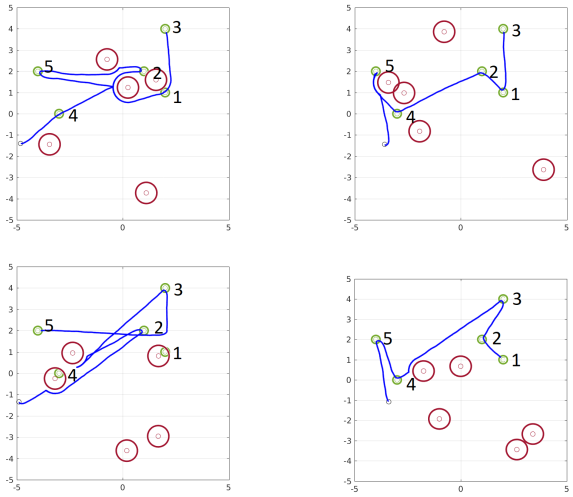
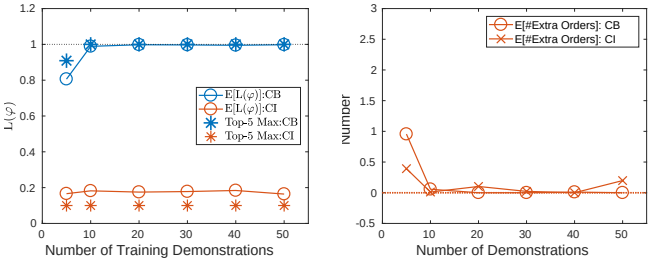


Fig. 6: Example trajectories from Scenario 3. Example trajectories from Scenario 1. The green circles denote the POIs and the red circles denote the threat avoidance zones.

while the CB likelihood converged to zero extra orders with a larger training set, the CI likelihood resulted in no change with respect to training set size.



(a) Scenario 2 $L(\varphi)$ (b) Scenario 2 orders

Fig. 7: Figure 7a depicts the $L(\varphi)$ values for Scenario 3 using prior 2. Figure 7b depicts the extra orderings inferred by the model. The dotted lines represent the ideal value.

In all the scenarios, it was observed that using the CB likelihood resulted in a better match between the inferred distribution over candidate formulas and the ground truth specification as compared to using the CI likelihood. Further, with a large enough training set, the CB likelihood inferred the ground truth specification for all the scenarios.

The runtime for MCMC inference is a function of the number of samples generated, the number of demonstrations in the training set, and the length of demonstrations. Scenarios 1 and 2 required an average runtime of 10 and 90 minutes for training set sizes of 5 and 50, respectively.

VI. Conclusion and Future Work

In conclusion we presented a probabilistic model to infer task specifications in terms of three behaviors encoded as LTL templates. For the ordering constraints we proposed two prior distributions that allow for efficient sampling of candidate formulas. We also proposed a likelihood function for demonstrations which depends only on the number of conjunctive clauses in the formula and is transferable across domains as it requires no information about the domain itself. Finally, we also demonstrated the effectiveness of our model first in a synthetic domain with multiple types of temporal constraints.

We believe that our approach of learning specifications in a temporal logic template opens up several avenues for future research. The prior distributions, we present here, restrict the topology of the DAG formed by precedence constraints, whereas real-world tasks are known to have more complex task constraints. For example in assembly tasks, there are multiple sub-assemblies which can be modeled as a linear chain, but an overall assembly can only be started once the sub-assemblies are completed. Development of prior distributions that support such DAG topologies would be one potential area for future research.

When people people infer task specifications, they simultaneously infer the type of DAG topology along with the particular instantiation. For example if they suspect that they are observing a task with a linear chain of precedence constraints, they would only try to infer the particular order, however, given only a few non-satisfying demonstrations, they might start favoring a more complex topology. Such inference behaviors can be modeled by defining a hierarchical probabilistic model. Further the hyper-priors for such a model can be tuned by observing the differences between its predictions of the task specification and a human supervisors prediction after observing the same set of demonstrations.

The specifications learned in LTL can also be combined with planning techniques developed by Kress-Gazit et al. [18], Raman et al. [25] or Littman et al. [21] to develop a LfD system that allows for an interpretable representation of its objectives and is guaranteed to satisfy the inferred objectives.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009.

[3] Thomas Arnold, Daniel Kasenberg, and Matthias Scheutz. Value alignment or misalignment–what will keep systems accountable. In *3rd International Workshop on AI, Ethics, and Society*, 2017.

[4] Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[5] Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international*

conference on Software engineering, pages 411–420. ACM, 1999.

[6] Kevin Ellis, Armando Solar-Lezama, and Josh Tenenbaum. Unsupervised learning by program synthesis. In *Advances in neural information processing systems*, pages 973–981, 2015.

[7] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B Tenenbaum. Learning to infer graphics programs from hand-drawn images. *arXiv preprint arXiv:1707.09627*, 2017.

[8] Cameron E. Freer, Daniel M. Roy, and Joshua B. Tenenbaum. Towards common-sense reasoning via conditional simulation: legacies of turing in artificial intelligence. In Rod Downey, editor, *Turing's Legacy*, volume 42 of *Lecture Notes in Logic*, pages 195–252. Cambridge University Press, 2014. ISBN 9781107338579. URL http://dblp.uni-trier.de/db/books/collections/D2014.html#FreerRT14.

[9] Noah Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*, 2012.

[10] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2018-4-9.

[11] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[12] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V Deshmukh, and Sanjit A Seshia. Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(11): 1704–1717, 2015.

[13] Daniel Kasenberg and Matthias Scheutz. Interpretable apprenticship learning with temporal logic specifications. *arXiv preprint arXiv:1710.10532*, 2017.

[14] Joseph Kim, Christopher J Banks, and Julie A Shah. Collaborative planning with encoding of users' high-level strategies. In *AAAI*, pages 955–962, 2017.

[15] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282. ACM, 2014.

[16] 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.

[17] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[18] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6): 1370–1381, 2009.

[19] Pierre Simon Laplace and Pierre Simon. A philosophical essay on probabilities, translated from the 6th french edition by frederick wilson truscott and frederick lincoln emory, 1951.

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

[21] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

[22] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. URL http://dl.acm.org/citation.cfm?id=645529.657801.

[23] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[24] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

[25] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M Murray, and Sanjit A Seshia. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 239–248. ACM, 2015.

[26] RN Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987. ISSN 0036-8075. doi: 10.1126/science.3629243. URL http://science.sciencemag.org/content/237/4820/1317.

[27] Joshua B Tenenbaum. Rules and similarity in concept learning. In *Advances in neural information processing systems*, pages 59–65, 2000.

[28] Russell Toris, David Kent, and Sonia Chernova. Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4504–4510. IEEE, 2015.

[29] Chanyeol Yoo and Calin Belta. Rich time series classification using temporal logic. In *Robotics: Science and Systems*, 2017.

[30] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.