


Supervised Bayesian specification inference from demonstrations

Ankit Shah^{1,*} , Pritish Kamath², Shen Li³, Patrick Craven⁴, Kevin Landers⁵, Kevin Oden⁴ and Julie Shah³

The International Journal of
Robotics Research
2023, Vol. 42(14) 1245–1264
© The Author(s) 2023
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649231204659
journals.sagepub.com/home/ijr



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 a task's execution; meanwhile, temporal logics provide a flexible language for expressing task specifications. Inspired by this, we present Bayesian specification inference, 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 observed between the inferred specification and the ground truth—both within a synthetic domain and during a real-world table setting task.

Keywords

Specification inference, learning from demonstrations, probabilistic models

1. 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 finish 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 long before gaining expertise in that task. In: the context of learning from demonstrations for robotic tasks, a system that can evaluate the acceptability of an execution before learning to execute a task can lead to more-focused exploration of execution strategies. Further, a system that can express its specifications would be more transparent with regard to its objectives than a system that simply imitates the demonstrator. Such characteristics are highly desirable in applications such as manufacturing or disaster response, where the cost of a mistake can be especially high. Finally, a robotic system with a correct understanding of the acceptability of executions can explore more-creative execution traces not present in the demonstrated set.

Most current approaches to learning from demonstration frame this problem as one of learning a reward function or policy within the setting of a Markov decision process; however, user specification of acceptable behaviors through reward functions and policies remains an open problem

Arnold et al. (2017). 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., 1999). In: this work, we propose a probabilistic model for inferring a task's temporal structure as a linear temporal logic (LTL) specification.

A specification inferred from demonstrations is valuable in conjunction with synthesis algorithms for verifiable controllers (Kress-Gazit et al., 2009 and Raman et al., 2015), as a reward signal during reinforcement learning (Li et al., 2017 and Littman et al., 2017), 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

¹Department of Computer Science, Brown University, Providence, RI, USA

²Google, Work carried out at CSAIL, MIT, Cambridge, MA, USA

³CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

⁴Lockheed Martin Corporation, Orlando, FL, USA

⁵Work carried out at Lockheed Martin Corporation, Orlando, FL, USA

*Previously at CSAIL, MIT, Cambridge, MA, USA

Corresponding author:

Ankit Shah, Department of Computer Science, Brown University,
115 Waterman Street, Providence, RI 02912, USA.

Email: ankit_j_shah@brown.edu

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. \(1999\)](#). Ideas from universal probabilistic programming languages formalized by [Freer et al. \(2014\)](#) and [Goodman et al. \(2008\)](#); [Goodman and Stuhlmüller \(2014\)](#) are key to our modeling approach; indeed, probabilistic programming languages enabled [Ellis et al. \(2018b, 2015\)](#) to perform inference over complex, recursively defined hypothesis spaces of graphics programs and pronunciation rules.

We evaluate our model’s performance within three domains. First, we test our model on a two dimensional synthetic domain and a real-world task involving setting a dinner table. For both these domains, the ground-truth specifications are known, and we report the capability of our model to achieve greater than 90% similarity between the inferred and ground-truth specifications. We also demonstrate the capability of our model to infer mission objective specifications for evaluating large-force combat flying exercises involving multiple friendly and hostile aircraft. The LFE domain is particularly challenging and unique in robotics research. It involves multiple independent decision-making participants, some of which act cooperatively and some adversarially. We demonstrate that our model makes predictions that are well-aligned with those of an expert acting as the commander for an example LFE mission. Further, we demonstrate that our method of using template compositions allows for an interpretable decision boundary for the classifier inferred by our model.

Bayesian specification inference was first introduced in work by [Shah et al. \(2018\)](#); in this paper, we extend the probabilistic model to be capable of learning both inductively (from positive examples only) and from both positive and negative examples. We also extend the evaluation presented by [Shah et al. \(2018\)](#) to include the large-force exercise domain, a multi-agent domain with independent agents acting both cooperatively and as adversaries.

2. Related work

[Argall et al. \(2009\)](#), [Chernova and Thomaz \(2014\)](#), and [Ravichandar et al. \(2020\)](#) provided a comprehensive survey of prior research into robot learning from demonstration (LfD) as applied to robotics. This body of work can broadly be organized into three major categories: imitation of task demonstrations, inference of task specifications, and learning models of interactions with the environment.

When LfD is framed through the lens of imitation of expert trajectories, the objective is to minimize a distance metric between the demonstrated trajectory and the trajectory performed by the learner. Prior research has utilized techniques such as dynamic motion primitives ([Schaal, 2006](#)), generalized cylinders ([Ahmadzadeh et al., 2017](#)), or an inference-based approach to learning and motion planning ([Rana et al., 2018](#)). A recent method proposed by [Billard et al. \(2022\)](#) and [Figuroa and Billard \(2018\)](#) leveraged dynamical systems theory and non-parametric priors

to learn the sequence of stable controllers while guaranteeing performance and safe interaction with the robot. Imitation-based approaches are best suited for learning motion-level task trajectories.

The next class of methods lies at the intersection of policy imitation and specification inference: inverse reinforcement learning (IRL). In these approaches, the demonstrations are recorded as state-action tuples. IRL algorithms are then designed to align the learned policy with the demonstrator’s policy in any given state. The task specification is implicitly encoded as a reward function representing the task. [Ng and Russell \(2000\)](#) and [Abbeel and Ng \(2004\)](#) first introduced IRL and through an optimization-based framing. [Ziebart et al. \(2008\)](#) developed algorithms for computing the stochastic policy with the maximum entropy while matching the state-visitation statistics. [Hadfield-Menell et al. \(2017\)](#) proposed a Bayesian approach to reward inference in a Markov setting. In contrast, [Chen et al. \(2020\)](#) and [Brown et al. \(2019, 2020\)](#) utilized ranking information to infer a reward function that enables a learner to outperform the demonstrator.

These works frame IRL in a setting where the underlying decision process is a Markov decision process (MDP). [Konidaris et al. \(2012\)](#), [Niekum et al. \(2015\)](#), and [Ranchod et al. \(2015\)](#) frame the IRL problem in a semi-Markov setting. Further [Unhelkar and Shah \(2019\)](#) proposed agent Markov models (AMM), a hierarchical approach that models the demonstrator’s policy as piecewise Markov with discrete control modes inferred using a non-parametric prior. These approaches utilize reward functions or policies to represent the task specification implicitly. However, [Arnold et al. \(2017\)](#) challenged the notion of using implicit modalities of task specification due to the difficulty in aligning optimal behavior with the intended behavior. However, choosing a suitable formalism for task specifications remains an open problem.

Some works directly infer the task specification as an intermediary representation for the learner. The learner does not learn “how” to perform the task but infers a binary classifier that indicates whether a given task execution is acceptable or not. Different specification formalisms such as object poses ([Toris et al. \(2015\)](#)), hierarchical task networks ([Breazeal et al., 2004](#)), and relative object poses ([Pérez-D’Arpino and Shah, 2017](#), [Mueller et al., 2018](#); [Luebbbers et al., 2020](#)). While these approaches separate the notion of task from the method of performing the task, these specification formalisms are limited in their ability to express non-Markov task specifications. This paper uses linear temporal logic (LTL) as the task specification language.

An analog problem to LfD in the software engineering domain is that of specification mining from execution traces of a program. Due to the well-studied relationship between LTL and automata ([Vardi, 1996](#); [Gerth et al., 1995](#)), LTL has been widely used as the specification language of choice for symbolic planning and specification mining. [Jha and Seshia \(2017\)](#) provided a theoretical framework for inductive inference of specifications from acceptable and unacceptable

execution traces. Further [Gabel and Su \(2008; 2010\)](#) and [Chivilikhin et al. \(2015\)](#) proposed template-based algorithms for specification mining. [Lemieux et al. \(2015\)](#) proposed Texada, a specification mining tool for inferring specifications corresponding to a given input template. These methods were based on exact satisfaction semantics (the candidate specifications must be satisfied by all execution traces) and are intended to recover all valid candidate specifications without quantifying uncertainty. Our proposed approach adopts Bayesian concept learning to allow for noisy demonstration data and estimates the model’s uncertainty about the true specification.

Within the context of stochastic sequential decision-making problems, [Kasenberg and Scheutz \(2017\)](#) and [Xu et al. \(2019\)](#) proposed specification inference models conditioned on observations of both state and actions of the trajectories provided by the demonstrators. Similarly, within symbolic planning domains, prior research focused on identifying the minimal LTL formulas that explain the difference between sets of plan traces ([Camacho and McIlraith, 2019](#); [Kim et al., 2017](#)). Finally, [Chou et al. \(2022\)](#) proposed an optimization-based method that infers a compact specification as a temporal logic formula with parametric propositions from demonstrations. Our proposed approach also focuses on inferring specifications from just state observations. It does not rely upon both positive and negative examples being available but will benefit from the availability of negative examples. Further, as it adopts a Bayesian approach, it infers a posterior belief distribution over the true specification rather than generating a single formula as the output. This allows the model to appropriately express its uncertainty over the true specification, particularly when only limited data is available.

[Kong et al. \(2014, 2017\)](#) proposed algorithms for mining signal temporal logic (STL) specifications based on a parametric grammar. Specifically, TempLogIn ([Kong et al., 2017](#)) provides the closest baseline to our approach. However, a key difference is that while TempLogIn performs an exhaustive breadth-first search over candidate formulas, our sampling-based inference approach can bias this search over regions of the formula hypothesis space in more fruitful directions. Moreover, TempLogIn requires both positive and negative examples, while our approach learns purely inductively. [Vazquez-Chanlatte et al. \(2018\)](#) proposed a maximum likelihood estimation framework for specification inference. We demonstrate that our proposed likelihood estimator is identical to their estimator for a near-perfect demonstrator. Finally, [Sobti et al. \(2023\)](#) proposed an extension of our inductive learning approach ([Shah et al., 2018](#)) to incorporate Bayesian experiment design, where the demonstration environment is varied to elicit discriminative and diverse demonstrations from the expert. Here, we focus on domains where specification inference is run strictly after data collection and has no influence on the data generation process.

In this paper, we identified two key challenges in inferring temporal logic specifications from demonstrations.

First, multiple candidate LTL formulas are satisfied by all observed demonstrations. Longer and more demonstrations alleviate the problem of ambiguity, but larger datasets are not always guaranteed in real-world scenarios. To address this challenge, our approach was inspired by Bayesian concept learning ([Tenenbaum, 1999](#)), where the learner can encode ambiguity over multiple candidate specifications as a belief distribution, thus not being restricted to inferring a single LTL formula. The second key challenge we addressed was the intractability of the hypothesis space of all possible LTL formulas. We constrained the hypothesis space by adopting a template-based approach using a subset of templates identified by [Dwyer et al., 1999](#). We combined this with probabilistic programming languages ([Freer et al., 2014](#); [Goodman et al., 2008](#)) to perform Bayesian inference over complex structured hypothesis spaces. Leveraging probabilistic programming languages is key to our approach. [Freer et al. \(2014\)](#) and [Goodman et al. \(2008\)](#) formalized the idea of a universal probabilistic programming language. These have led to a suite of Turing-complete probabilistic programming languages such as Church ([Goodman et al., 2008](#)), webppl ([Goodman and Stuhlmüller, 2014](#)), and Gen ([Cusumano-Towner et al., 2019](#)). Probabilistic programming languages have enabled adopting a Bayesian approach to grammatical inference ([De la Higuera, 2010](#)) by allowing the composition of modular inference algorithms over a wide variety of distributions. In particular, [Ellis et al. \(2015; 2018a, 2018b\)](#) demonstrated the success of these approaches in inferring graphics program and language grammar rules from very few examples. [Silver et al. \(2020\)](#) demonstrated the utility of Bayesian grammatical inference in learning robotic manipulation policies by chaining primitive skills.

3. Bayesian specification inference

Our goal is to explicitly identify task specifications that model temporal properties; therefore, we selected linear temporal logic (LTL) ([Pnueli, 1977](#)) as the task specification language. We leveraged the compositional nature of formal languages such as LTL, and adopted a template-based approach that constructs complex specifications by composing a known library of templates. We begin with a brief introduction to the syntax, definitions, and semantics of LTL. Next, we identify a relevant fragment of LTL based on temporal properties identified by [Dwyer et al. \(1999\)](#), and [Menghi et al. \(2019\)](#) to compose our hypothesis space. Next, we formally define the problem of Bayesian specification inference and describe our technical approach.

3.1. Linear temporal logic

Linear temporal logic (LTL), introduced by [Pnueli \(1977\)](#), 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 $[\alpha]$ of the set of propositions, α . The notation $[\alpha], t \models \varphi$ indicates that φ holds at time t . The trace $[\alpha]$ satisfies φ (denoted as $[\alpha] \models \varphi$) iff $[\alpha], 0 \models \varphi$. The minimal syntax of LTL can be described as follows:

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

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 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 $t \geq t_1$. While we define the syntax of LTL here, we provide concrete examples of using LTL to describe desirable temporal behaviors as we define the hypothesis space for specification inference.

3.2. Defining the hypothesis space

We identified three temporal properties, namely, global satisfaction of a constraint, eventual completion of subtask, and temporal ordering among the subtasks as the most relevant temporal behaviors for task specifications. With φ_{global} , $\varphi_{eventual}$, and φ_{order} representing LTL formulas for these behaviors, the task specification is written as follows:

$$\varphi = \varphi_{global} \wedge \varphi_{eventual} \wedge \varphi_{order} \quad (2)$$

Note that as the formula is constructed using conjunctions of the subformulas, each subformula must be satisfied to ensure the completion of the overall task. Next, we define the LTL templates for each of the subformulas that could be a part of the task specification.

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

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

Note that for each proposition, $\tau \in \mathbf{T}$, the temporal property of always satisfying τ is represented by $\mathbf{G}(\tau)$ in LTL. Simultaneously satisfying all propositions, ($\forall \tau \in \mathbf{T}$), we compose each individual formula using the conjunction operator, \wedge , which results in the final formula for φ_{global} , as depicted in equation (3). Such formulas are useful for specifying that some constraints must always be met—for example, a robot must avoid collisions while in motion, or an aircraft must avoid no-fly zones.

3.2.2. Eventual completion. Let Ω be the set of all candidate subtasks, and let $\mathbf{W}_1 \subseteq \Omega$ be the set of subtasks that must be completed if the conditions represented by π_w ; $w \in \mathbf{W}_1$ are met. ω_w are propositions representing the completion of a subtask. The LTL formula that specifies this behavior is written as follows:

$$\varphi_{eventual} = \left(\bigwedge_{w \in \mathbf{W}_1} (\pi_w \rightarrow \mathbf{F}\omega_w) \right) \quad (4)$$

Here, the \rightarrow operator signifies that the eventual completion of a subtask $w \in \mathbf{W}_1$ is only necessary if the implicant proposition π_w is true in the initial state. Such templates are useful in specifying the completion of certain subgoals conditional on initial task conditions that the decision-making agent does not control. For example, a dessert spoon must be placed on the table (represented by the proposition ω_w) if the dinner menu includes a dessert course (represented by the proposition π_w).

3.2.3. Temporal ordering. Every set of feasible ordering constraints over a set of subtasks is mapped to a DAG over nodes representing these subtasks. Each edge in the DAG corresponds to a binary precedence constraint. Let \mathbf{W}_2 be the set of binary temporal orders defined by $\mathbf{W}_2 = \{(w_1, w_2) : w_1 \in \mathbf{V}, w_2 \in \text{Descendants}(w_1)\}$, where \mathbf{V} is the set of all nodes within 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:

$$\varphi_{order} = \left(\bigwedge_{(w_1, w_2) \in \mathbf{W}_2} (\pi_{w_1} \rightarrow (\neg\omega_{w_2} \mathbf{U}\omega_{w_1})) \right) \quad (5)$$

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. Conditional temporal ordering is useful not only to enforce ordering constraints among the subgoals but also to toggle the enforcement when the subgoals are not accessible. For example, in multi-aircraft large-force missions, each aircraft type has individual tactical objectives. As described in our large force exercise (LFE) domain, suppression of enemy air defense (SEAD) aircraft are responsible for neutralizing enemy surface-to-air missile (SAM) systems capable of attacking friendly aircraft. It is usually desirable for the subgoal to represent enemy SAMs being inoperable before friendly aircraft enter enemy airspace, and the ordering template can represent this property; however, achieving the SEAD subgoal is only possible if SEAD aircraft are available at the start of the mission, therefore highlighting the importance of conditional ordering.

For the purposes of this paper, we assume that all required propositions $\alpha = [\tau, \pi, \omega]^T$ and labeling functions $f(x)$ are known, along with the sets \mathbf{T} and Ω and the mapping of

the condition propositions π_w to their subtasks. Given these assumptions, the problem of inferring the correct formula for a task is equivalent to identifying the correct subsets τ , W_1 , and W_2 , that explain the observed demonstrations well.

3.3. Specification learning as Bayesian inference

We represent the task demonstrations as an observed sequence of state variables, $[\mathbf{x}]$. Let $\alpha \in \{0,1\}^n$ represent a vector of finite dimension formed by n Boolean propositions. The propositions are related to the state variables through a labeling function, $\alpha = f(\mathbf{x})$, which is known a priori.

The inference model is provided a label, y , to indicate whether an execution is acceptable or not, along with the actual demonstrations. Thus, the training set $\mathbf{D} = \{([\alpha]_i, y_i); i \in \{1, 2, \dots, n_{demo}\}\}$ consists of n_{demo} demonstrations along with the label. The output, again, is a probability distribution $P(\varphi|\mathbf{D})$. The Bayes theorem is fundamental to the problem of inference, and is stated as follows:

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

$P(h)$ is the prior distribution over the hypothesis space, and $P(\mathbf{D}|h)$ is the likelihood of observing the data given a hypothesis. Our hypothesis space is defined by $\mathbf{H} = \varphi$, where φ 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). \mathbf{D} is the training dataset.

3.3.1. 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 $\text{SampleSubset}(A, p)$ as the process of applying a Bernoulli trial with a success probability of 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 $\text{SampleSubset}(\mathbf{T}, p_G)$ and $\text{SampleSubset}(\mathbf{\Omega}, p_E)$. 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 three possible methods of sampling a DAG, with different restrictions on the graph topology.

Algorithm 1 SampleSetsOfLinearChains

```

1: function SAMPLESETSOFLINEARCHAIN( $\Omega, p_{part}$ )
2:    $i \leftarrow 1; C_i \leftarrow []$ 
3:    $P \leftarrow$  random permutation( $\Omega$ )
4:   for  $a \in P$  do
5:      $C_i.append(a)$ 
6:      $k \leftarrow$  Bernoulli( $p_{part}$ )
7:     if  $k = 1$  then
8:        $i = i + 1; C_i \leftarrow []$ 
9:   return  $C_j \forall j$ 

```

3.3.1.1. Linear chains. A linear chain is a DAG such that all subtasks must occur within 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.

3.3.1.2. 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 $a \in P$, 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 a probability of p_{part}) or the old active chain continues (with a probability of $1 - p_{part}$).

3.3.1.3. Forest of sub-tasks. This graph topology includes forests (i.e., sets of disjoint trees). A given node has no temporal constraints with respect to its siblings, but must precede all its descendants. Algorithm 2 depicts a probabilistic program for sampling a forest. Line 2 creates P , a random permutation of the subtasks. Line 3 initializes an empty forest. In order to support a recursive sampling algorithm, the data structure representing forests is defined as an array of trees, \mathcal{F} . The i^{th} tree has two attributes: a root node, $\mathcal{F}[i].root$, and a ‘‘descendant forest,’’ $\mathcal{F}[i].descendant$, in which the root node of each tree is a child of the root node defined as the first attribute. The length of the forest, $\mathcal{F}.length$, is the number of trees included in that forest. The size of a tree, $\mathcal{F}[i].size$, is the number of nodes within the tree (i.e., the root node and all of its descendants). For each subtask in the random permutation P , line 5 inserts the given subtask into the forest as per the recursive function InsertIntoForest defined in lines 7 through 13. In line 8, an integer i is sampled from a categorical distribution, with $\{1, 2, \dots, \mathcal{F}.length + 1\}$ as the possible outcomes. The probability of each outcome is proportional to the size of the trees in the forest, while the probability of $\mathcal{F}.length + 1$ being the outcome is proportional to N_{new} , a user-defined parameter. This sampling process is similar in spirit to the Chinese restaurant process (Aldous (1983)). If the outcome of the draw is $\mathcal{F}.length + 1$, then a new tree with root node a

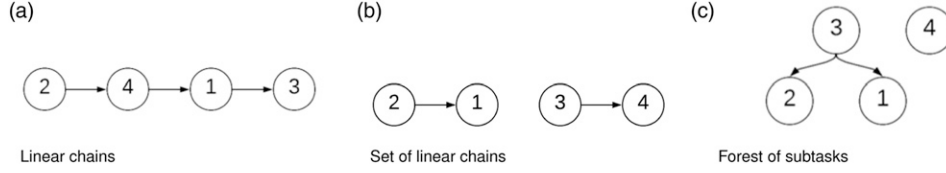


Figure 1. Illustrative examples of directed acyclic graphs of precedence constraints over four sub-tasks as sampled by our priors.

is created in line 10; otherwise, `InsertIntoForest` is called recursively to add a to the forest $\mathcal{F}[i].\text{descendants}$, as per line 12.

Algorithm 2 `SampleForestofSubtasks`

```

1: function SAMPLEFORESTOFSUBTASKS( $\Omega, N_{new}$ )
2:    $\mathcal{P} \leftarrow$  random permutation( $\Omega$ )
3:    $\mathcal{F} \leftarrow []$ 
4:   for  $a \in \mathcal{P}$  do
5:      $\mathcal{F} = \text{InsertIntoForest}(\mathcal{F}, a)$ 
6:   return  $\mathcal{F}$ 
7: function INSERTINTOFOREST( $\mathcal{F}, a$ )
8:    $i \leftarrow$  Categorical( $[\mathcal{F}[1].\text{size}, \mathcal{F}[2].\text{size}, \dots, \mathcal{F}[\mathcal{F}.\text{length}].\text{size}, N_{new}]$ )
9:   if  $i = \mathcal{F}.\text{length} + 1$  then
10:    Create new tree  $\mathcal{F}[\mathcal{F}.\text{length} + 1].\text{root} = a$ 
11:   else
12:     $\mathcal{F}[i].\text{descendants} = \text{InsertIntoForest}(\mathcal{F}[i].\text{descendants}, a)$ 
13:   return  $\mathcal{F}$ 

```

Figure 1 depicts illustrative examples of the directed acyclic precedence graphs sampled using the priors. Three prior distributions based on the four probabilistic programs are described in Table 1. In all the priors, ϕ_{global} and $\phi_{eventual}$ are sampled using `SampleSubset(\mathcal{T}, p_G)` and `SampleSubset(Ω, p_E)`, respectively.

3.3.2. Likelihood function. The likelihood distribution, $P(\{[\alpha]_i\} | \phi, \{y_i\})$, is the probability of observing the trajectories within the dataset 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 follows:

$$P(\{[\alpha]_i\} | \phi, \{y_i\}) = \prod_{i=1}^{n_{demo}} P(\phi) P([\alpha]_i | \phi, y_i) \quad (7)$$

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 and Dale, 1951, 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 τ_1, \mathbf{W}_{1_1} , and $\mathbf{W}_{2_1 \mathbf{1}}$) and ϕ_2 (defined by

Table 1. Prior definitions and hyperparameters.

Prior	ϕ_{Order}	Hyperparameters
Prior 1	RandomPermutation(Ω)	p_G, p_E
Prior 2	SampleSetsOfLinearChains(Ω, p_{part})	p_G, p_E, p_{part}
Prior 3	SampleForestofSubTasks(Ω, N_{new})	p_G, p_E, N_{new}

subsets τ_2, \mathbf{W}_{1_2} , and \mathbf{W}_{2_2}) the likelihood odds ratio if $y_i = 1$ is defined as follows:

$$\frac{P([\alpha]_i | \phi_1)}{P([\alpha] | \phi_2)} = \begin{cases} \frac{2^{N_{conj_1}}}{2^{N_{conj_2}}} = \frac{2^{|\tau_1| + |\mathbf{W}_{1_1}| + |\mathbf{W}_{2_1}|}}{2^{|\tau_2| + |\mathbf{W}_{1_2}| + |\mathbf{W}_{2_2}|}} & , [\alpha] \models \phi_2 \\ \frac{2^{N_{conj_1}}}{\epsilon} = \frac{2^{|\tau_1| + |\mathbf{W}_{1_1}| + |\mathbf{W}_{2_1}|}}{\epsilon} & , [\alpha] \not\models \phi_2 \end{cases} \quad (8)$$

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 (2000).

A second choice for a likelihood function, inspired by Shepard (1987), is defined as the SIM model by Tenenbaum (2000); we call this the “complexity-independent” (CI) likelihood function, and it is defined as follows:

$$P([\alpha] | \phi) = \begin{cases} 1 - \epsilon, & \text{if } [\alpha] \models \phi \\ \epsilon, & \text{Otherwise} \end{cases} \quad (9)$$

We must define likelihood functions for both acceptable and unacceptable demonstrations. Note that the likelihood function defined by equation (8) produces a relatively larger likelihood value if the candidate formula correctly classifies the demonstration, and a very small likelihood value if it does not. Following the classical probability argument as before, with $2^{N_{conj}}$ conjunctive clauses in a candidate formula, there are $2^{N_{conj}}$ possible evaluations of each of the individual clauses that would result in the given demonstration not satisfying the candidate formula. Thus, the likelihood function for $y_i = 0$ is defined as follows:

$$\frac{P([\alpha]_i | \varphi_1)}{P([\alpha]_i | \varphi_2)} = \begin{cases} \frac{2^{N_{conj_1}} (2^{N_{conj_2}} - 1)}{2^{N_{conj_2}} (2^{N_{conj_1}} - 1)} & , [\alpha] \not\models \varphi_2 \\ \frac{2^{N_{conj_1}}}{(2^{N_{conj_1}} - 1)\epsilon} & , [\alpha] \models \varphi_2 \end{cases} \quad (10)$$

An equivalent SIM likelihood function for examples with $y_i = 0$ is defined as follows:

$$P([\alpha] | \varphi) = \begin{cases} 1 - \epsilon, & \text{if } [\alpha] \not\models \varphi \\ \epsilon, & \text{Otherwise} \end{cases} \quad (11)$$

Note that for larger values of N_{conj_1} and N_{conj_2} and page a negative label $y_i = 1$, the difference between the CI and the CB likelihood function is very small.

Vazquez-Chanlatte et al. (2018) proposed a maximum likelihood estimation framework for specifications expressed informal logics. Their approach for computing the maximum entropy estimate results in the likelihood function being proportional to the ratio of the observed satisfaction rate of a candidate specification to the satisfaction rate resulting from randomized behavior. Our approach to approximating the likelihood function corresponds to their result if the observed satisfaction rate is assumed to be close to unity, and the random satisfaction rate is computed assuming that each of the clauses is satisfied as a random outcome of a Bernoulli trial with balanced outcome probabilities.

3.3.3. Inference. We implemented our probabilistic model in webppl (Goodman and Stuhlmüller (2014)), a Turing-complete probabilistic programming language. The hyperparameters, including those defined in Table 1 and ϵ , were set as follows: $p_E, p_G = 0.8$; $p_{part} = 0.3$; $N_{new} = 5$; $\epsilon = 4 \times \log(2) \times (|\mathbf{T}| + |\mathbf{\Omega}| + 0.5|\mathbf{\Omega}|(|\mathbf{\Omega}| - 1))$. 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 serving 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. We ran the inference on a desktop with an Intel i7-7700 processor.

While the inference procedure is handled directly by webppl, we provide a brief example of the MCMC sampling algorithm using the Metropolis-Hastings acceptance criterion. Consider a domain with two subtasks w_1 and w_2 . The demonstrator provides two demonstrations in the dataset along with the acceptability labels, $D = \{([\alpha]_1, y_1), ([\alpha]_2, y_2)\}$. Let’s assume that in both demonstrations, the demonstrator completed the subtask w_1 before subtask w_2 in the first demonstration and w_2 before w_1 in the second.

Further, let’s assume that both demonstrations were labeled acceptable. For brevity, let’s assume that all positional propositions, π_i , always hold in this case.

Further, assume that we select Prior 2, which induces a prior probability distribution $P(\varphi)$ over the hypothesis space as per the probabilistic program described in Table 1. To estimate the posterior belief conditioned on observing the dataset $P(\varphi | \mathbf{D})$, we use approximate inference by sequential sampling from a proposal distribution (in this case, the prior distribution) and either accepting or rejecting the new sample.

We begin with an initial sample from the prior distribution, for example, $\varphi_1 = \mathbf{F} w_1 \wedge \mathbf{F} w_2$, i.e., the agent must complete both subtasks w_1 , and w_2 but in any order. The posterior probability, $P(\varphi_1 | \mathbf{D}) \propto P(\varphi_1)P(\mathbf{D} | \varphi_1)$. By Equation (7), $P(\mathbf{D} | \varphi_1) = P([\alpha]_1 | \varphi_1, y_1)P([\alpha]_2 | \varphi_1, y_2)$. The individual demonstration likelihoods are calculated per Equations (8) and (10) as applicable.

At the next iteration of the MCMC algorithm, let the sample be $\varphi_2 = \mathbf{F} w_1 \wedge \mathbf{F} w_2 \wedge \neg w_2 \mathbf{U} w_1$, which encodes the requirement of the demonstrator to complete both w_1 and w_2 , but enforces the ordering constraint that w_1 must precede w_2 . As per the Metropolis-Hastings criterion (Metropolis et al., 1953; Hastings, 1970), this sample is accepted with the probability defined as follows:

$$\begin{aligned} P(\text{accept}) &= \min\left(1, \frac{P(\varphi_2 | \mathbf{D})}{P(\varphi_1 | \mathbf{D})}\right) \\ &= \min\left(1, \frac{P(\varphi_2)P(\mathbf{D} | \varphi_2)}{P(\varphi_1)P(\mathbf{D} | \varphi_1)}\right) \end{aligned} \quad (12)$$

Thus if the demonstration $[\alpha]_1$ is acceptable, and $[\alpha]_2$ is not, then the sample $\varphi - 2$ will be accepted as long as $P(\varphi_2)/P(\varphi_1) > P(\varphi_1 | \mathbf{D})/P(\varphi_2 | \mathbf{D})$. Note that as per the size principle encoded in the likelihood function, the likelihood of φ_2 is greater than that of φ_1 . However, if both demonstrations were acceptable, $P(\varphi_1 | \mathbf{D})/P(\varphi_2 | \mathbf{D}) \gg 1$, therefore φ_2 is very likely to be rejected as the next sample.

The MCMC algorithm evaluates a fixed number of new sample proposals. The burn-in parameter, n_{burn} , discards the first n_{burn} samples for estimating the posterior distribution. Finally, the frequency of the accepted samples is used to estimate the posterior belief distribution.

4. Evaluations

We evaluated the performance of our model across three diverse domains. First, we developed a synthetic domain where an agent must navigate a 2D space with fully observed threats and waypoints. In each mission, the agent must visit a set of waypoints while respecting temporal precedence constraints, and it must avoid a subset of the threats. This domain was chosen so that many satisfying demonstrations can be readily generated for various environment configurations and ground truth specifications. Such an evaluation across a diverse set of specifications is

especially important for research into specification inference and policy explanation. We initially proposed the synthetic domain (Shah et al., 2018), and it has been further utilized by Sobti et al. (2023) for their extension of specification inference with optimal experiment design incorporated into the data generation process and by Sanneman et al. (2021) for explaining autonomous systems policy for temporal tasks.

Next, we applied our model to the real-world task of setting a dinner table. Such tasks are characteristics of manipulation tasks performed by a single agent in a house-scale environment. This task is a proxy for multi-step manipulation tasks that are common use cases for robots in domestic assistance or manufacturing. It is also a task familiar to a non-expert user demographic; thus, high-quality satisfying demonstrations are readily available. Therefore, the task of setting a dinner table has been widely studied in prior works on LfD (Toris et al., 2015; Ahmadzadeh et al., 2017; Rana et al., 2018).

Finally, we evaluated our approach within the domain of evaluation of large-force exercises (LFE). Large-force exercises are simulated air-combat games used to train combat pilots. Any given LFE will typically include multiple aircraft of heterogeneous capabilities and will only feature sparse coordination between friendly aircraft. In addition to cooperative aircraft, these domains include adversarial aircraft and ground forces. Such large-scale multi-agent decentralized domains are understudied in robotics but are common in large-scale systems such as disaster response, assembly line design and planning, and logistics. These domains are high-impact domains for deploying robots and other autonomous coordination systems, and our proposed approach can be valuable in inferring team-level objectives from previous mission execution data. In this paper, we developed a simulation environment for large-force exercises using joint-semi automated forces (JSAF), a constructive software environment. We used our proposed specification inference model to infer mission objectives using mission execution data, and annotations from a subject matter expert (in this case, the mission commander who designs the scenario and debriefs and evaluates the participating pilots).

4.1. Metrics

The evaluation metrics used to test the quality of the inferred specifications depend upon whether the ground-truth specifications are known. For domains in which it is known (the synthetic and dinner table domains), the ground-truth specification is defined using subsets τ^* , \mathcal{W}_1^* , and \mathcal{W}_2^* (as per Equations 3, 4, and 5), and a candidate formula φ is defined by subsets τ , \mathcal{W}_1 , and \mathcal{W}_2 . In such cases, we define the degree of similarity using the Jaccard index (Jaccard, 1912) as follows:

$$L(\varphi) = \frac{|\{\tau^* \cup \mathcal{W}_1^* \cup \mathcal{W}_2^*\} \cap \{\tau \cup \mathcal{W}_1 \cup \mathcal{W}_2\}|}{|\{\tau^* \cup \mathcal{W}_1^* \cup \mathcal{W}_2^*\} \cup \{\tau \cup \mathcal{W}_1 \cup \mathcal{W}_2\}|} \quad (13)$$

The maximum possible value of $L(\varphi)$ is one such that both formulas are equivalent. One 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 of the inferred distribution from the ground truth. We also report the maximum value of $L(\varphi)$ among the top 5 candidates in the posterior distribution. We classify the inferred orders in \mathcal{W}_2 as correct if they are included in the ground truth, incorrect if they reverse any constraint within the ground truth, and “extra” otherwise. (Extra orders over-constrain the problem, but do not induce incorrect behaviors.)

For the LFE domain, where the ground-truth specifications are unknown but SME annotations for whether the mission objectives were accomplished are provided for the dataset, we use the weighted F1 score for both “achieved” and “failed” labels. This score is evaluated on a test set that is held out while using the remaining examples in the dataset to infer the specifications. We also measure the true negative rate in a setting where the training set only contains positive examples, where the mission objectives were achieved by the friendly forces, while the test set contained a mixture of positive and negative mission executions.

4.2. 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. The state of the agent \mathbf{x} represents the position of that agent within the task space.

Let $\tau = \{1, 2, \dots, n_{threats}\}$ represent a set of threats positioned at $\mathbf{x}_T \forall i \in \tau$, respectively. A proposition τ_i is associated with each threat location $i \in \tau$ such that:

$$\tau_i = \begin{cases} \text{true,} & \|\mathbf{x} - \mathbf{x}_T\| \geq \epsilon_{threat} \\ \text{false,} & \text{otherwise} \end{cases} \quad (14)$$

The proposition τ_T holds if the agent is not within the avoidance radius ϵ_{threat} of the threat location.

Let $\Omega = \{1, 2, \dots, n_{POI}\}$ represent the set of POIs positioned at $\mathbf{x}_P \forall i \in \Omega$. A proposition ω_i is associated with each POI such that:

$$\omega_i = \begin{cases} \text{true,} & \|\mathbf{x} - \mathbf{x}_P\| \leq \epsilon_{POI} \\ \text{false,} & \text{otherwise} \end{cases} \quad (15)$$

ω_i evaluates as true if the agent is within a tolerance radius ϵ_{POI} of the POI.

Finally, propositions $\pi_i \forall i \in \Omega$ are conditions propositions that denote the accessibility of the POI i , and are defined as follows:

$$\pi_i = \begin{cases} \text{false,} & \exists j \text{ such that } \|\mathbf{x}_{P_i} - \mathbf{x}_{T_j}\| \leq \epsilon_{\text{threat}} \\ \text{true,} & \text{otherwise} \end{cases} \quad (16)$$

π_i evaluates as false if the POI i is inside the avoidance region of any of the threats.

The agent can be programmed to visit the accessible POIs and avoid threats as per the ground-truth specification. The ground-truth specifications are stated by defining the following: a set $\mathbf{T} \subseteq \tau$ that represents the subset of threats that the agent must avoid; a set $\mathbf{W}_1 \subseteq \Omega$ that represents the subset of POIs the agent must visit; and the ordering constraints defined by \mathbf{W}_2 , a set of feasible pairwise precedence constraints between the POIs.

Here, we demonstrate the results of applying our inference model to three scenarios with differing ground-truth specifications. Each scenario had five threats, and five waypoints, but the specification over the order in which the waypoints were to be visited varied in each scenario. In: Scenario 1, all waypoints had to be visited in a particular order. In: Scenario 2, there were precedence constraints over a subset of waypoints, but not all of them. In: Scenario 3, there were no ordering constraints at all. These scenarios were specifically selected to examine the inductive biases induced by each of our priors in cases where the ground truth specification was fully constrained, partially constrained and most relaxed respectively.

4.2.1. Scenario 1. In Scenario 1, we placed five threats in the task-domain, and their positions were sampled from a uniform distribution for each demonstration. There were four points of interest, labeled 1,2,3,4, and their positions were fixed across all demonstrations. The agents were required to visit the POIs in a fixed order ([1,2,3,4]). Example trajectories from this scenario are depicted in Figure 2.

The posterior distribution was computed using prior 1 (defined in Table 1), with both CB (Equation 8) and CI (Equation 9) likelihood functions. The expected and maximum values among the top 5 a posteriori formula candidates of $L(\varphi)$ are depicted in Figure 3. We observed that the CB likelihood function performed better than the CI likelihood function at inferring the complete specification. Using the CI function resulted in a higher posterior probability assigned to formulas with high prior probability that were satisfied by all demonstrations. These tended to be simple, non-informative formulas; the CB function assigned higher probability mass to more-complex formulas that explained the demonstrations correctly. Figure 3(b) 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 training set size increased; this effect was not observed with the CI function.

The posterior distribution was also computed using priors 2 and 3 with the CB likelihood function. The expected and maximum values among the top 5 a posteriori formula candidates of $L(\varphi)$ are depicted in Figure 4(a). Prior 3 aligned better with the ground-truth specification with fewer training examples. With a larger training set, prior

2 recovered the exact specification, while prior 3 failed to do so. Figure 4(b) depicts the expected value of the correct and extra orders in the candidate formulas included in the posterior distribution. The a priori bias of prior 3 toward longer chains is apparent, as it recovered more correct orders with fewer training demonstration in comparison to prior 2. Prior 2 recovered all correct priors with more training examples; however, prior 3 failed to do so with 30 training examples.

4.2.2. Scenario 2. Scenario 2 contained five POIs 1,2,3,4,5 and five threats. Like Scenario 1, the threat positions were sampled uniformly for each demonstration. All the POIs, if accessible, had to be visited. A partial ordering constraint was imposed such that POIs [1,3,5] had to be visited in that specific order, while POIs {2, 4} could be visited in any order. Some demonstrations generated for Scenario 2 are depicted in Figure 5.

For Scenario 2, the posterior distribution was computed using priors 2 and 3, as the ground-truth specification did not lie in support of prior 1. The expected and maximum values among the top 5 formula candidates of $L(\varphi)$ are depicted in Figure 6(a). Given a sufficient number of training examples, both priors were able to infer the complete formula; with 10 or more training examples, both priors returned the ground-truth formula among the top 5 candidates with regard to posterior probabilities. Figure 6(b) depicts the correct and extra orders inferred in Scenario 2. Prior 3 assigned a larger prior probability to longer task chains compared with prior 2, but both priors converged to the correct specification given enough training examples.

4.2.3. Scenario 3. Scenario 3 included five threats and five POIs labeled {1, 2, 3, 4, 5}, respectively. The threat positions were uniformly sampled for each scenario. Each of the POIs, if accessible, had to be visited; however, there were no constraints placed on the order in which they were visited. Figure 7 depicts some of the example demonstrations.

Again, the posterior distribution was computed using priors 2 and 3. The expected and maximum values among the top 5 formula candidates of $L(\varphi)$ are depicted in Figure 8(a). In: this scenario, both priors performed equally well with regard to recovering the ground-truth specification. With 10 or more demonstrations, both priors returned the ground-truth specification as the maximum a posteriori estimate. The expected value of the extra orders contained in the posterior distributions is depicted in Figure 8(b). Once again, the tendency of prior 3 to return longer chains is apparent, as more formulas in the posterior distribution returned a greater number of extra ordering constraints as compared with prior 2.

4.2.4. Complexity considerations. The runtime for MCMC inference is a function of the number of samples generated, the number of demonstrations in the training set, and

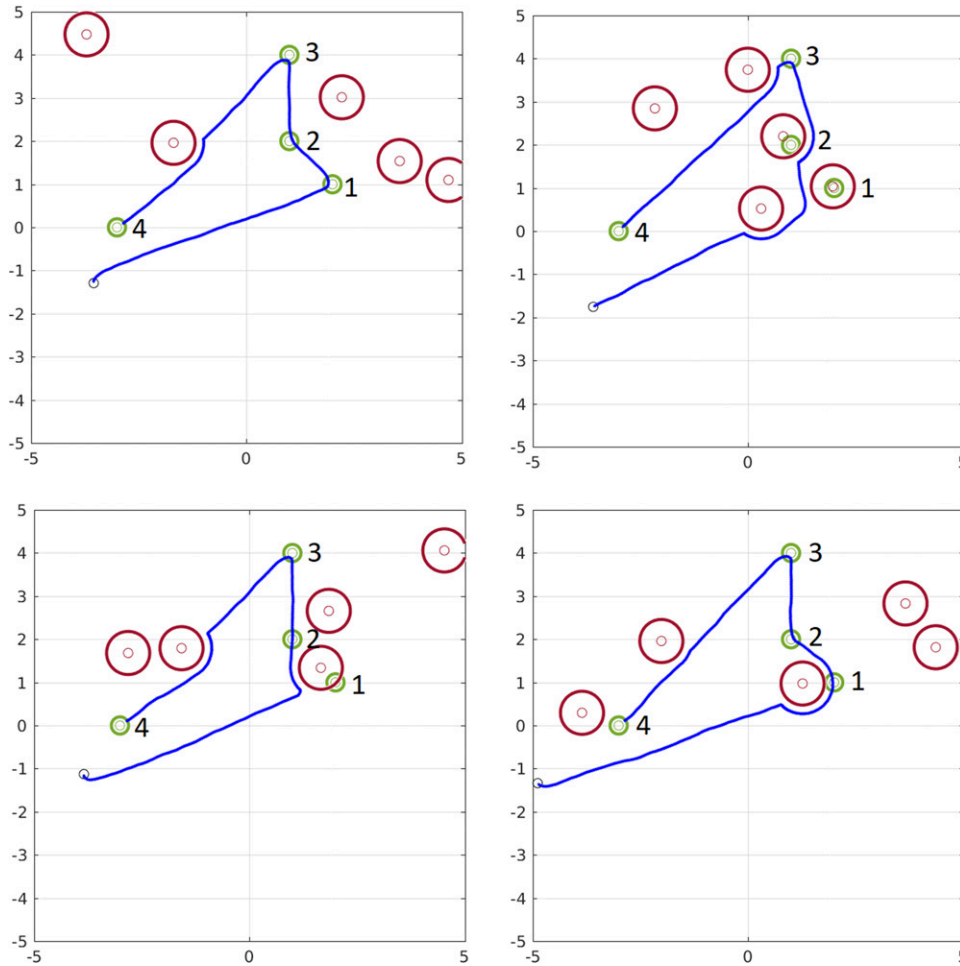


Figure 2. Example trajectories from Scenario 1. Green circles denote the POIs; red circles denote the avoidance zones of threats.

demonstration length. Scenarios 1 and 2 required an average runtime of 10 and 90 min for training set sizes of 5 and 50, respectively.

TempLogIn (Kong et al., 2017) required 33 min to terminate with three PSTL clauses. For all the scenarios, the mined formulas did not capture any of the temporal behaviors in Section, indicating that additional PSTL clauses were required. However, with five and 10 PSTL clauses, the algorithm did not terminate within the 24-h runtime cutoff. Scaling TempLogIn to larger formula lengths is difficult, as the size of the search graph increases exponentially with the number of PSTL clauses, and the algorithm must evaluate all formula candidates of length n before candidates of length $n + 1$.

4.2.5. Discussion. Our experiments within the synthetic domain indicate our model’s capability of inferring a variety of ground-truth specifications with different ordering constraints. The comparison between CI and CB likelihood functions indicated that a likelihood model that followed the size principle (Tenenbaum, 1999) was key to inferring the correct task specifications inductively. Further experiments comparing Priors 2 and 3 over three scenarios with different

ordering constraints indicated the difference in inductive biases encoded within the prior distributions, but more importantly, the experiments also demonstrate the prior biases can be overruled with adequate evidence from observed data.

4.3. Dinner table domain

We also tested our model on a real-world task: setting a dinner table. This task featured eight dining set pieces that had to be organized on a table while the demonstrator avoided contact with a centerpiece. Figure 9(a) depicts each of the final configurations of the dining set pieces, depending upon the type of food served. The pieces placed on the table were varied for each of the eight configurations; however, the positions of the pieces remained constant across all final configurations. A total of 71 demonstrations were collected, with six participants providing multiple demonstrations for each of the four configurations. (Figure 10)

The eight dinner set pieces included a large dinner plate, a smaller appetizer plate, a bowl, a fork, a knife, a spoon, a water glass, and a mug; the set of pieces is represented by

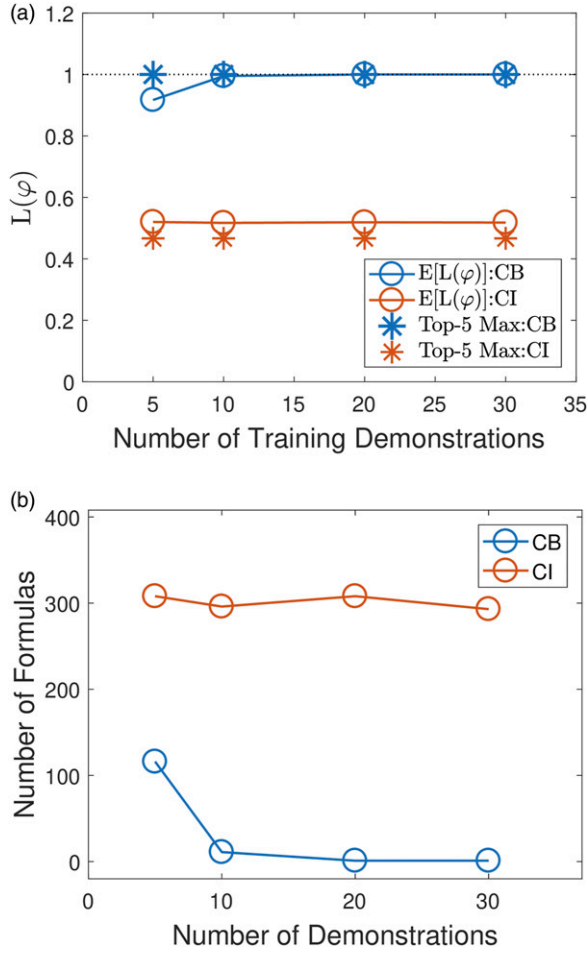


Figure 3. (a) depicts the results from Scenario 1, with the dotted line representing the maximum possible value of $L(\varphi)$. (b) shows the number of unique formulas in the posterior distribution.

Ω . Each piece was tracked with a motion-capture system over the course of the demonstration, with the pose of an object $i \in \Omega$ in the world frame represented by T_i^O . In addition, the pose of the wrists of the demonstrators T_{h1}^O and T_{h2}^O were also tracked throughout the demonstration. We defined propositions that tracked whether an object was in its correct position or whether a demonstrator's wrist was too close to the centerpiece using task-space region (TSR) constraints proposed by Berenson et al. (2011).

The origin for each TSR constraint is located at the desired final position of each object. The pose $T_{w_i}^O$ represents the transform between the origin frame and the TSR frame for the object, i . The bounds for B_i represent the translation and rotational tolerances of the constraint. Finally, P_i represents the set of poses in the TSR frame that fall within the tolerance bounds. The pose of object i with respect to the TSR frame is given by $T_i^{w_i} = (T_{w_i}^O)^{-1} T_i^O$. A proposition ω_i is associated with object i as follows:

$$\omega_i = \begin{cases} \text{true,} & T_i^{w_i} \in P_i \\ \text{false.} & \text{otherwise} \end{cases} \quad (17)$$

Thus, the proposition ω_i evaluates as true if the pose of object i satisfies the TSR constraints, and false otherwise.

A TSR constraint is also associated with the centerpiece, where T_c^O represents the pose of the centerpiece with respect to the world frame, and the bounds of the constraint are defined by B_c , with P_c representing the set of poses that fall within the tolerances. The poses of the demonstrator's wrists with respect to this TSR frame are given by $T_{h_i}^c$ for $i \in \{1, 2\}$. A proposition τ_c is associated with the centerpiece, and is defined as follows:

$$\tau_c = \begin{cases} \text{false,} & T_{h1}^c \in P_c \vee T_{h2}^c \in P_c \\ \text{true,} & \text{otherwise} \end{cases} \quad (18)$$

τ_c evaluates as false if either of the wrist poses falls within the TSR bounds, and evaluates as true otherwise.

Finally, condition propositions $\pi_i \forall i \in \Omega$ encode whether the object i must be placed. Their values are set prior to the demonstration and held constant for its duration. These propositions encode the fact that serving certain

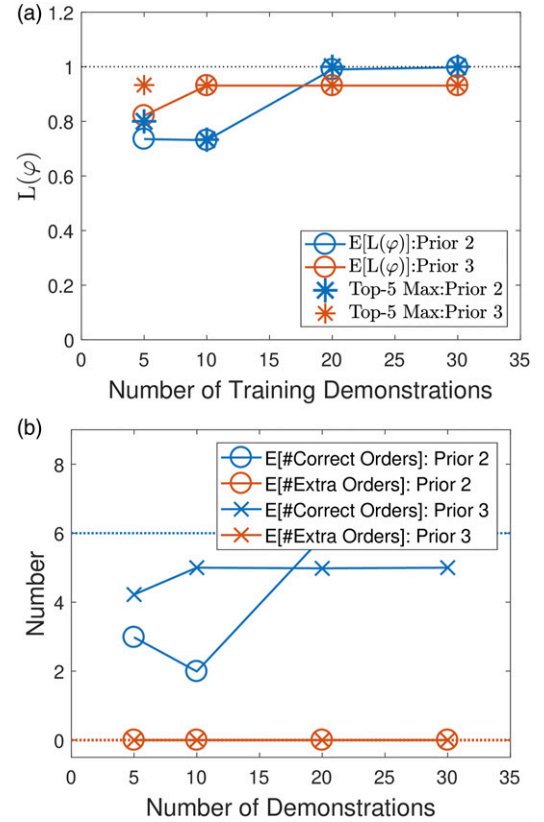


Figure 4. (a) depicts the results from Scenario 1 using priors 2 and 3, with the dotted line representing the maximum possible value of $L(\varphi)$. (b) depicts the expected value of the number of correct and extra orders in the posterior distribution.

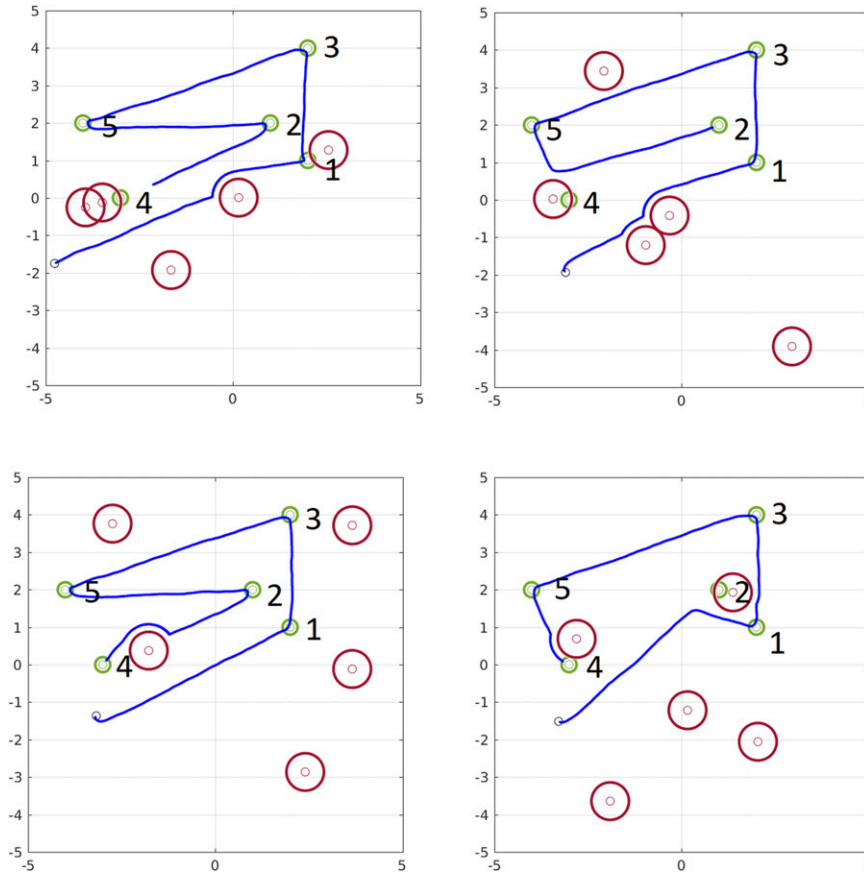


Figure 5. Example trajectories from Scenario 2. Green circles denote the POIs; red circles denote the avoidance zones of threats.

courses during a meal requires specific placement of certain dinner pieces.

Based on the propositions defined above and the configurations of the dinner table, the ground-truth specifications of this task are as follows: the demonstrator's wrists should never enter the centerpiece's TSR region (global satisfaction); if π_i is true, then the corresponding dinner piece must be placed on the table (eventual completion); and the large plate must be placed before the smaller plate, which in turn must be placed before the bowl (ordering). We constructed the posterior distributions over candidate specification using priors 2 and 3 by incorporating subsets of the training demonstrations of varying sizes, and evaluated the similarity between the inferred specifications and the ground truth using the expected and maximum values among the top 5 a posteriori candidates of the metric $L(\varphi)$.

With prior 2, our model correctly identified the ground truth as one of the top 5 a posteriori formula candidates in all cases. With prior 3, the inferred formula contained additional ordering constraints compared with the ground truth. Using all 71 demonstrations, the MAP candidate had one additional ordering constraint: that the fork be placed prior to the spoon. Upon review, it was observed that this condition was not satisfied in only 4 of the 71 demonstrations.

4.4. Evaluating large force exercises

Large-force exercises (LFE) are combat flight training exercises that involve multiple aircraft groups, with each group playing a designated role in the completion of the mission. Evaluating a LFE execution is a challenging task for the mission commander. The raw state-space of the domain includes the navigation data for each aircraft involved in the scenario (up to 36 aircrafts were included in the scenarios we simulated), along with configuration settings for each of those aircrafts (weapon stores, weapon deployments, etc.) and outcomes of combat engagements that occur throughout the scenario. The mission commander must distill this time-series and evaluate the mission based on multiple output modalities. He or she must first identify the transition points between predetermined scenario phases, then evaluate the overall success of the mission's execution in terms of a finite number of predetermined objectives. Evaluation of the mission objectives depends not only upon the final state of the scenario, but also on the behavior of the aircrafts throughout the mission, thus making LTL a suitable grammar for representing mission objective specifications.

We evaluated the capabilities of our model to infer LTL specifications that match a mission commander's evaluations of mission objective completion. In: this section, we

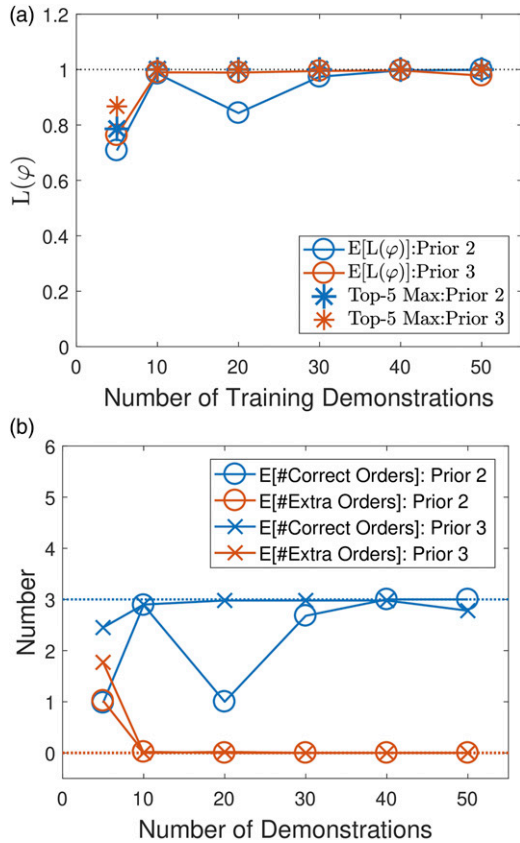


Figure 6. (a) indicates the $L(\varphi)$ values for Scenario 2, and (b) depicts the correct and extra orderings inferred in Scenario 2. The dotted lines represent the number of orderings in the true specification.

begin by describing the nature of the offensive counter air (OCA) mission that serves as the subject of our study. Next, we describe how these missions are evaluated by experts, and how the stated mission objectives are well-suited for use with the temporal behavioral templates we use in our candidate formulas. Finally, we describe the results obtained when applying our model to the LFE domain dataset.

4.4.1. LFE scenario description. Each LFE for the OCA mission we modeled consists of 18 friendly aircrafts and a variable number of enemy aircrafts and ground-based threats. Among the friendly aircrafts, there are eight escort aircrafts that are capable air-to-air fighters, eight SEAD (suppression of enemy air defenses) aircrafts capable of attacking ground-based threats, and two strike aircrafts that carry the ammunition that must be deployed in order to attack a designated ground target within a time-on-target (TOT) window. The aircrafts' starting positions during a typical scenario are depicted in Figure 11. The role of the mission commander is to debrief the participants once a LFE scenario execution is completed. During debriefing, the LFE-OCA scenario is segmented into four phases by design as follows:

- Escort Push

- Strikers Push
- Time-On-Target (TOT)
- Egress

The mission commander must identify the times that correspond to the transitions between these mission phases, and also provide an assessment of whether the following three mission objectives were achieved:

- **MO1:** Gain and maintain air superiority.
- **MO2:** Destroy an assigned target within the TOT window.
- **MO3:** Friendly attrition should not exceed 25%.

Each of the mission objectives is a Boolean-valued function of the raw state-space of the LFE scenario, and the mapping between them is not explicitly known. Inputs from subject matter experts (SMEs) were also utilized to represent the mission execution in terms of certain Boolean propositions over which we can apply our probabilistic model. The propositions were defined as follows:

1. Enemy aircraft attrition (50%, 75%, 100%) (three propositions).
2. Either strike aircraft fired upon.
3. Either strike aircraft shot down.
4. Last munition released by strikers.
5. Strike aircrafts flying in on-target flight phase.
6. Assigned target hit.
7. Friendly aircraft attrition (25%, 50%, 75%) (three propositions, each turn false if the corresponding attrition is reached).

In order to generate realistic demonstrations of how the different executions unfold, the scenarios were defined in Joint Semi-Automated Forces (JSAF)—a constructive environment capable of simulating realistic aircraft behavior. The data collected for each demonstration included the position, speed, attitude, and rates of each of the aircrafts (both friendly and hostile); the individual mission phase of each aircraft (a discrete set of phases by which the aircraft specific mission timeline can be labeled); and the firing times, designated targets, detonation times, and outcomes of each weapon deployment over the course of the scenario. The mapping from the collected data to the Boolean propositions stated above is well defined.

In order to apply our probabilistic model to the LFE domain, we defined the sets τ and Ω . The propositions 7, 2, and 3 were included in the set τ as candidates for global satisfaction. The propositions 1, 4, 5, and 6 were included in Ω as candidates for eventual completion.

4.4.2. Data collection. A total of 24 instances of LFEs were simulated and included in the dataset. Each instance had a different outcome with respect to the mission objectives,

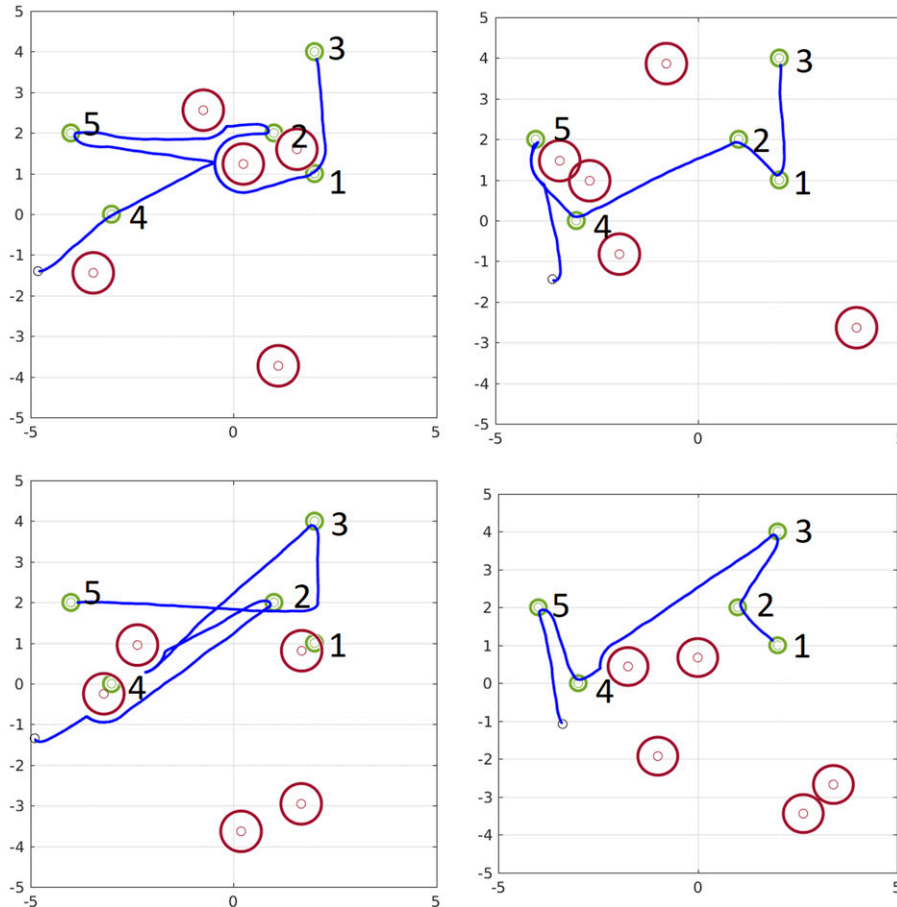


Figure 7. Example trajectories from Scenario 3. The green circles denote the POIs; the red circles denote the threat avoidance zones.

based on the different outcomes of engagements between friendly and hostile forces. Each scenario was evaluated by an SME acting as a mission commander performing a manual debrief. The primary annotation task was to evaluate whether each of the objectives was successfully achieved upon mission completion. The secondary annotation task was to determine the segmentation points among the four scenario phases on the mission timeline. The segmentation task is not directly relevant to specification inference, but we used the labels to simultaneously train a secondary classifier in one of the baselines.

4.4.3. Benchmarks. The training data for evaluations of LFEs consists of both acceptable and unacceptable demonstrations, along with the label for that demonstration; thus, it can be viewed as a supervised learning problem. We decided to compare the classification accuracy of our model against a classifier trained with neural-network-based classifiers trained using gradient-based learning with both recurrent architectures (Hochreiter and Schmidhuber, 1997; Graves et al., 2005; Ordóñez and Roggen, 2016), and the state-of-the-art transformer architectures (Vaswani et al., 2017) for time series classification.

1. **Standalone:** Here, the recurrent neural network is trained to jointly optimize the binary cross-entropy for the classification of each of the three mission objectives. The loss functions for all the mission objectives are equally weighted. The recurrent neural networks are composed of long and short-term memory (LSTM) modules (Hochreiter and Schmidhuber, 1997), along with their bidirectional variants (Graves et al., 2005). Such models were proposed for time series classification tasks demonstrating promising performance (Ordóñez and Roggen, 2016). Additionally, we adopted the transformer architecture (Vaswani et al., 2017) by average-pooling the learned feature embeddings over the time dimension. These models—hereafter referred to as “LSTM,” “Bi-LSTM,” and “Transformer,” respectively—were trained using only the time series of the propositions as inputs.
2. **Coupled:** In prior research, performance improvements on a primary task have been observed due to simultaneous training on a secondary related task (Sohn et al., 2015). We hypothesized that simultaneously training the classifier on the secondary task of identifying scenario phases might improve classification accuracy compared with a standalone RNN. The loss functions

used were binary cross-entropy for each mission objective and categorical cross-entropy for the scenario phase identification. The overall loss function was an equally weighted sum of the individual cost functions. These models were also composed of LSTM modules, their bi-directional counterparts, and the transformer models; they are referred to as “LSTM Coupled,” “Bi-LSTM Coupled,” and “Transformers Coupled,” respectively. These models were trained using the propositions and collected flight phase data.

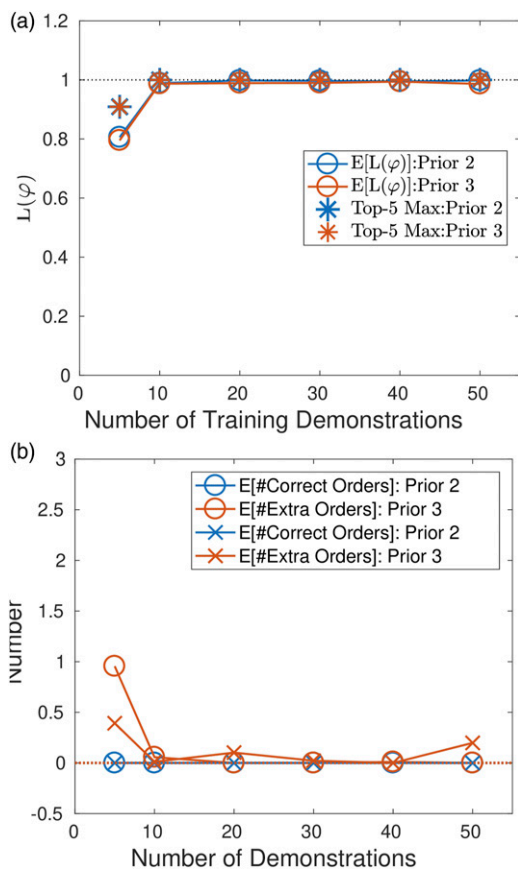


Figure 8. 8(a) indicates the $L(\phi)$ values for Scenario 3, and (b) depicts the correct and extra orderings inferred in Scenario 3. The dotted lines represent the number of orderings in the true specification.

4.4.4. Evaluations. The classification models were evaluated through a four-fold cross-validation wherein the training dataset was divided into four equal partitions, with three of the partitions used for training (18 scenarios) and testing performed on the remaining partition (6 scenarios); this was repeated across all partitions, and the metrics were averaged across each of the partitions. We report the F1 score of each classifier. Next, we evaluated the performance of the best performing neural-network architecture against our model in a purely inductive learning setting. We partitioned the dataset into training set consisting of 80% of satisfying mission executions for each of the mission objectives. Correspondingly, the test set consisted of all the failed mission executions and 20% of the successful task executions. In: the inductive learning scenario, we report both the F1 score and the true negative detection rate of the classifiers. This allows us to evaluate whether the classifier can learn a meaningful decision boundary from purely positive examples.

We also applied our model to the entire dataset in order to analyze which of the propositions were included in the maximum a posteriori estimate of the specifications. The overall accuracy of the classifiers was evaluated using the F1 score on all the predictions for both the possible outcomes of the mission objectives (“Achieved” and “Failed”) for each mission objective.

4.4.5. Results. As presented in Table 2, our model outperformed RNN-based supervised learning models. However, the transformer based classifier outperformed our method for MO1 and MO2. At the same time we notice that contrary to our hypothesis, addition of the secondary task and features resulted in degraded classification performance. We also notice that with Bayesian specification inference model, prior 2 outperformed prior 3; and a possible explanation for this phenomenon is that prior 3 demonstrates an inductive bias towards longer task chains; therefore, it demonstrated a higher false negative rate.

We also noticed the tendency of RNN models to collapse to predicting the most commonly occurring outcome in the training set for all values of inputs. Thus, the model was unable to achieve high accuracies even on the training set,

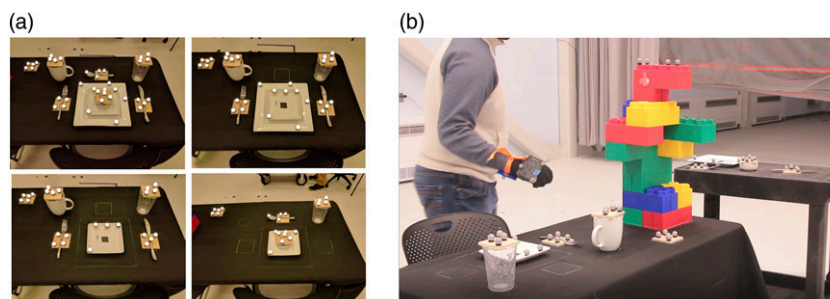


Figure 9. (a) depicts all the final configurations. (b) depicts the demonstration setup. (Photographed by the authors in April 2017.)

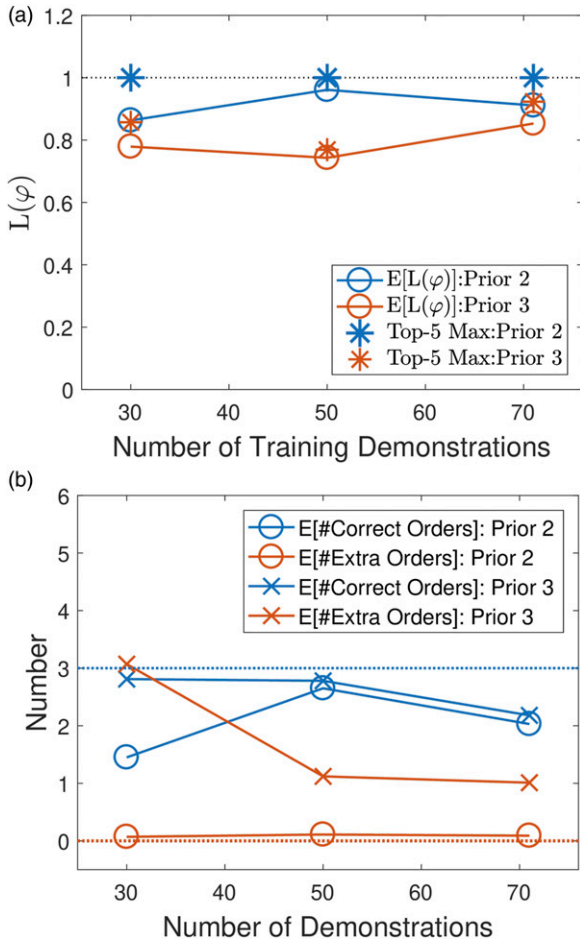


Figure 10. (a) depicts the $L(\varphi)$ values for the dinner table domain, with the dotted line representing the maximum possible value. (b) depicts the correct and extra orderings inferred within this domain; the dotted lines represent the number of orderings in the true specification.

suggesting that it is not only the small size of the dataset that results in poor performance. This might indicate that either greater model capacity or a different model architecture may be required. The accuracy achieved by the transformer-based classifier provides evidence for this. The multi-headed attention mechanism, and positional encodings can act as event detectors and temporal reasoners, respectively, resulting in high classification accuracy even with very few training examples.

Table 3 presents the classification performance metrics for the best performing neural architecture, and the best performing specification inference model in the inductive learning setting. Due to the absence of negative examples from the training set, the transformer model also demonstrates mode collapse, and always classifies the execution as having achieved the mission objectives. Thus, while supervised learning models require training from a balanced set of positive and negative examples, our approach is capable of learning meaningful LTL specifications even from an extremely biased training set containing only positive examples. This is especially relevant for the LFE domain where there is no guarantee on the availability of balanced datasets.

Finally, we analyzed the maximum a posteriori formula returned by our model using prior 2, and the F1 scores obtained were 0.959, 0.918, and 0.959 for the three mission objectives, respectively. The compositional structure of the model allowed us to examine the propositions included in the formulas and interpret the decision boundaries of the classifiers; the results were as follows:

- MO1 (Gain and maintain air-superiority)** The propositions included in φ_{global} were 7, 3, and 2; these correspond to a maximum allowable friendly attrition

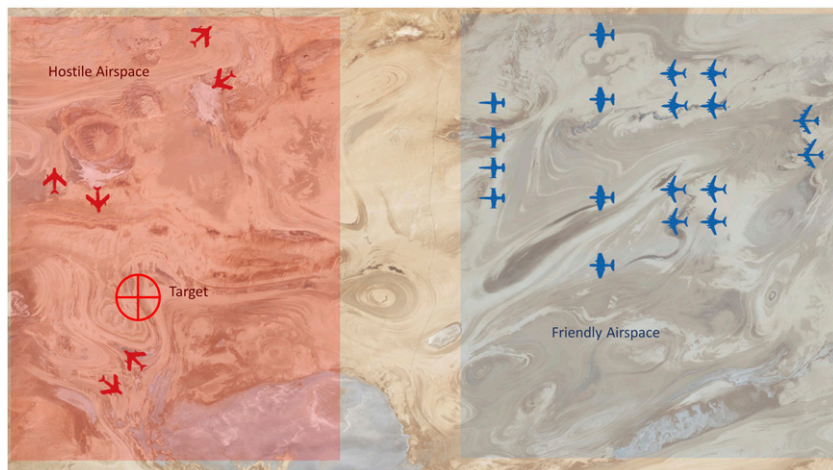


Figure 11. The starting configuration of a large-force exercise scenario. The red aircrafts are the hostile forces, and the blue are friendly forces.

Table 2. Weighted F1 scores for both scenario outcomes for each of the classifiers.

Classifier	MO1	MO2	MO3
LSTM	0.53	0.53	0.48
Bi-LSTM	0.53	0.53	0.48
LSTM Coupled	0.53	0.53	0.48
Bi-LSTM Coupled	0.53	0.53	0.48
Transformer	0.86	0.75	0.83
Transformer Coupled	0.79	0.73	0.82
BSI (Prior 2)	0.67	0.71	0.88
BSI (Prior 3)	0.67	0.66	0.88

Bold Values indicate best performance

Table 3. Results weighted F1 score, and true negative detection rate for inductive learning evaluation.

Classifier	MO1		MO2		MO3	
	F1	True -ve	F1	True -ve	F1	True -ve
Transformer	0.50	0.00	0.50	0.00	0.47	0.00
BSI (Prior 2)	0.44	0.63	0.66	1.00	0.6	0.63

Bold Values indicate best performance

rate of less than 25%, and enforcing the condition that the strikers were never fired upon or shot down, respectively. (This is consistent with the definition of air superiority.) The propositions included in $\varphi_{eventual}$ were 4, 1, and 5; these correspond to strikers eventually releasing their weapons, the friendly forces shooting down 75% of the enemy fighters, and strike aircrafts eventually reaching their on-target flight phase, respectively. (Again, the included propositions indicate that gaining air superiority allowed strikers to operate freely.) Finally, φ_{order} enforced that friendly forces shot down 50% of the hostile air threats before strikers released their weapons.

2. **MO2: (Destroy assigned target)** The propositions included in φ_{global} were 7 and 3; these represent a maximum friendly attrition of 50%, and only enforcing that the strikers were never shot down, respectively. (Note that this does not enforce the condition that strikers were never fired upon.) $\varphi_{eventual}$ included 1, 4, 5, and 6; these represent eventually shooting down all hostile aircrafts (which would seem unnecessary), strikers entering their on-target flight phase, eventually releasing their weapons—and, most importantly, attacking the assigned target. φ_{order} enforced the condition that the friendly aircrafts had to shoot down all hostiles before the close of the TOT window.
3. **MO3: No more than 25% friendly losses:** The propositions in φ_{global} included 7, 2, and 3; these correctly enforced that no more than 25% friendly aircrafts could be shot down, and also that the strikers were never shot down or fired upon. $\varphi_{eventual}$ included 1, 4, and 5,

representing 75% hostile force attrition, and enforced that the strikers had to eventually enter their on-target phase and deploy their weapons. No orders were included in the formula. The propositions that enforced weapon deployment by strikers and requisite hostile attrition were not required for this objective to be fulfilled; however, they were included by the model due to their frequent occurrence with objective completion. The compositional nature of the model allows the user to identify constraints that will be easily enforced.

4.5. Discussion

We demonstrated the efficacy of our model in identifying task specifications in three domains. Our experiments within the synthetic domain were conducted to vary the ground-truth formulas with a varying degree of diversity, thus demonstrating the robustness of the model. Our proposed model correctly identified the ground truth specification for each scenario when the model observed an adequate number of training examples. Our experiments within the synthetic domain were also instrumental in highlighting the importance of the size principle (Tenenbaum, 2000; 1999) in inferring specifications in an inductive learning setting.

Next, we demonstrated the capability of our proposed model to identify the ground truth specification for setting a dinner table. This typical real-world task is often studied in the context of LfD. The task specifications were inferred from observations of human volunteers setting the table with minimal constraints and structure enforced on the demonstrators. This demonstrates the viability of our model in inferring specifications for typical single-robot tasks expected to arise in the context of domestic robotics, service robotics, and manufacturing robotics in independent task cells. The synthetic domain and table-setting task were examples of inferring task specification inductively.

Finally, we evaluated our model in the large force exercise (LFE) domain, where we collected data streams from multiple independent decision-making agents at various levels of abstraction. The task executions were performed in a multi-agent simulated domain (with no central decision-making authority). This multi-agent setting is common in real-world domains such as combat operations, exploratory expeditions, and disaster response domains, where multiple agents with heterogenous capabilities operate towards individual and shared objectives. Our model demonstrated equivalent alignment with the mission commanders assessment's compared to the state-of-the-art trajectory classification approaches based on transformer-based classification models (Vaswani et al., 2017) when trained on a mix of acceptable and unacceptable execution traces through four-fold cross-validation. However, if the training set only included acceptable mission execution traces, our proposed model vastly outperformed the transformer classifier. The added interpretability of the inferred logical formulas makes our model valuable as a decision-support

system for mission commanders evaluating such mission executions.

While we demonstrated the utility of our approach for real-world-sized problems, there are several open problems. First, while our approach was only operationalized for LTL, it applies to any discrete specification grammars with a fragment defined by templates composed through conjunctive composition. Currently, we also assume that the propositions are given to the model; future development of this approach would also consider the scenario where the classifiers that evaluate the truth value of propositions are inferred simultaneously with the formula structure using a hierarchical Bayesian approach. Finally, our approach is further limited in expressivity due to the user-defined templates. However, a future extension that extends the template library using the commonly occurring specification patterns would be an interesting research direction.

5. Conclusion

In this work, we presented a probabilistic model to infer task specifications in terms of three behaviors encoded as LTL templates. We presented three prior distributions that allow for efficient sampling of candidate formulas as per the templates. We also presented a likelihood function that depends only upon the number of conjunctive clauses in the candidate formula, and is transferable across domains as it requires no information about the domain itself. Finally, we demonstrated our model on three distinct evaluation domains. On the domains where the ground-truth specifications were known, we demonstrated the capability of our model to identify the ground-truth specification with up to 90% similarity, in both a low-dimensional synthetic domain and a real-world dinner table domain. In: the large-force exercise domain, where the ground-truth specifications are not known, we showed the ability of our model to align its predictions with those of an expert to a greater extent than supervised learning techniques. We also demonstrated our model's ability to explain its decision boundaries due to the compositional nature of the formula template.

Acknowledgments

We would like to acknowledge Dr Kevin Gluck, and Dr Donald Duckro at the Airman Systems Directorate and Zachary "Zen" Wallace at the Rickard Consulting Group for their expertise in mission design and analysis that were instrumental in creating the LFE scenario. We would also like to thank David Macannuco at the Lockheed Martin corporation for his expertise in modeling and simulation.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been funded by the Lockheed Martin corporation and the Air Force Research Laboratory.

ORCID iD

Ankit Shah  <https://orcid.org/0000-0001-6818-0827>

References

- Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning, Banff, Alberta, Canada, July 2004. ISBN 1581138385. DOI: [10.1145/1015330.1015430](https://doi.org/10.1145/1015330.1015430)
- Ahmadzadeh SR, Rana MA and Chernova S (2017) Generalized cylinders for learning, reproduction, generalization, and refinement of robot skills. In: *Proceedings of Robotics: Science and Systems*, Boston, MA, USA, July 2017. DOI: [10.15607/RSS.2017.XIII.021](https://doi.org/10.15607/RSS.2017.XIII.021)
- Aldous DJ (1983) Exchangeability and Related Topics. *École d'Été de Probabilités de Saint-Flour XIII, Lecture Notes in Mathematics, vol 1117*. Berlin, Heidelberg, Germany: Springer.
- Argall BD, Chernova S, Veloso M, et al. (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.
- Arnold T, Kasenberg D and Scheutz M (2017) Value alignment or misalignment—what will keep systems accountable. In: 3rd International Workshop on AI, Ethics, and Society, San Francisco, CA, February 2017.
- Berenson D, Srinivasa S and Kuffner J (2011) Task space regions: a framework for pose-constrained manipulation planning. *The International Journal of Robotics Research* 30(12): 1435–1460.
- Billard A, Mirrazavi S and Figueroa N (2022) *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Cambridge, MA: MIT Press.
- Breazeal C, Brooks A, Gray J, et al. (2004) Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robotics* 1(02): 315–348.
- Brown D, Goo W, Nagarajan P, et al. (2019) Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In: Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, June 2019. PMLR, pp. 783–792.
- Brown DS, Goo W and Niekum S (2020) Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In: Conference on Robot Learning, Virtual, November 2020. PMLR, pp. 330–359.
- Camacho A and McIlraith SA (2019) Learning interpretable models in linear temporal logic. In: Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, Berkeley, CA, USA, July 2019, pp. 621–630.
- Chen L, Paleja R and Gombolay M (2020) Learning from suboptimal demonstration via self-supervised reward regression. In: Proceedings of Conference on Robot Learning, Virtual, November 2020.

- Chernova S and Thomaz AL (2014) Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(3): 1–121.
- Chivilikhin D, Ivanov I and Shalyto A (2015) Inferring temporal properties of finite-state machine models with genetic programming. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15. New York, NY, USA: ACM. ISBN 978-1-4503-3488-4, pp. 1185–1188. URL: <https://doi.acm.org/10.1145/2739482.2768475>
- Chou G, Ozay N and Berenson D (2022) Learning temporal logic formulas from suboptimal demonstrations: theory and experiments. *Autonomous Robots* 46(1): 149–174.
- Cusumano-Towner MF, Saad FA, Lew AK, et al. (2019) Gen: a general-purpose probabilistic programming system with programmable inference. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, pp. 221–236. ISBN 978-1-4503-6712-7. URL: <http://doi.acm.org/10.1145/3314221.3314642>
- de la Higuera C (2010) *Grammatical Inference: Learning Automata and Grammars*. Cambridge, UK: Cambridge University Press, DOI: [10.1017/CBO9781139194655](https://doi.org/10.1017/CBO9781139194655).
- Dwyer MB, Avrunin GS and Corbett JC (1999) *Patterns in property specifications for finite-state verification*. In: Proceedings of the 21st International Conference on Software Engineering, Los Angeles, CA, USA, May 1999. ACM, pp. 411–420.
- Ellis K, Solar-Lezama A and Tenenbaum J (2015) Unsupervised learning by program synthesis. In: *Advances in Neural Information Processing Systems*, Montreal, Canada, December 2015. pp. 973–981.
- Ellis K, Morales L, Sablé-Meyer M, et al. (2018a) Learning libraries of subroutines for neurally-guided bayesian program induction. In: S Bengio, H Wallach, H Larochelle, et al. (eds) *Advances in Neural Information Processing Systems, volume 31*. URL <https://proceedings.neurips.cc/paper/2018/file/7aa685b3b1dc1d6780bf36f7340078c9-Paper.pdf>
- Ellis K, Ritchie D, Solar-Lezama A, et al. (2018b) Learning to infer graphics programs from hand-drawn images. In: *Advances in Neural Information Processing Systems, volume 31*. URL <https://proceedings.neurips.cc/paper/2018/file/6788076842014c83cedadbe6b0ba0314-Paper.pdf>
- Figueroa N and Billard A (2018) A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In: Proceedings of the Conference on Robot Learning, Zurich, Switzerland, October 2018, pp. 927–946.
- Freer CE, Roy DM and Tenenbaum JB (2014) Towards common-sense reasoning via conditional simulation: legacies of turing in artificial intelligence. *Turing's Legacy* 42: 195–252.
- Gabel M and Su Z (2008) *Symbolic mining of temporal specifications*. In: Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, May 2008. ACM, pp. 51–60.
- Gabel M and Su Z (2010) Online inference and enforcement of temporal properties. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering—Volume, Cape Town, South Africa, May 2010. ACM, 1, pp. 15–24.
- Gerth R, Peled D, Vardi MY, et al. (1995) Simple on-the-fly automatic verification of linear temporal logic. In: *International Conference on Protocol Specification, Testing and Verification*, Warsaw, Poland, June 1995. Springer, pp. 3–18.
- Goodman ND and Stuhlmüller A (2014) *The Design and Implementation of Probabilistic Programming Languages*. <http://dippl.org> Accessed: 2018-4-9.
- Goodman ND, Mansinghka VK, Roy D, et al. (2008) *Church: a language for generative models*. In: Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence, Helsinki, Finland, July 2008. AUAI Press, pp. 220–229. ISBN 0974903949.
- Graves A, Fernández S and Schmidhuber J (2005) Bidirectional lstm networks for improved phoneme classification and recognition. In: *International Conference on Artificial Neural Networks*, Warsaw, Poland, September 2005. Springer, pp. 799–804.
- Hadfield-Menell D, Milli S, Abbeel P, et al. (2017) Inverse reward design. In: *Advances in Neural Information Processing Systems*, Long Beach, CA, December 2017, pp. 6765–6774.
- Hastings WK (1970) Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* 57(1). Oxford, UK: Oxford University Press.
- Hochreiter S and Schmidhuber J (1997). Long short-term memory. *Neural Computation* 9: 1735–1780.
- Jaccard P (1912) The distribution of the flora in the alpine zone. 1. *New Phytologist* 11(2): 37–50.
- Jha S and Seshia SA (2017) A theory of formal synthesis via inductive learning. *Acta Informatica* 54(7): 693–726.
- Kasenberg D and Scheutz M (2017) Interpretable apprenticeship learning with temporal logic specifications. In: IEEE Conference on Decision and Control, Melbourne, Australia, December 2017.
- Kim J, Banks CJ and Shah JA (2017) Collaborative planning with encoding of users' high-level strategies. In: Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, February 2017, pp. 955–962.
- Kong Z, Jones A, Medina Ayala A, et al. (2014) Temporal logic inference for classification and prediction from data. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, ACM, pp. 273–282.
- Kong Z, Jones A and Belta C (2017) Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control* 62(3): 1210–1222.
- Konidaris G, Kuindersma S, Grupen R, et al. (2012) Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research* 31(3): 360–375.
- Kress-Gazit H, Fainekos GE and Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics* 25(6): 1370–1381.
- Laplace PS and Dale AI (1951) *Pierre-Simon Laplace Philosophical Essay on Probabilities, Translated from the fifth French edition of 1825 With Notes by the Translator*. NY, USA: Springer New York.
- Lemieux C, Park D and Beschastnikh I (2015) General LTL specification mining. In: 30th IEEE/ACM International Conference on Automated Software Engineering, Lincoln, NE, November 2015. IEEE, pp. 81–92.
- Li X, Vasile CI and Belta C (2017) *Reinforcement learning with temporal logic rewards*. In: Intelligent Robots and Systems

- (IROS), 2017 IEEE/RSJ International Conference on, Vancouver, Canada, September 2017. IEEE, pp. 3834–3839.
- Littman ML, Topcu U, Fu J, et al. (2017) Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*.
- Luebbers MB, Brooks C, Mueller CL, et al. (2020) ARC-LfD: using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration. In: IEEE International Conference on Robotics and Automation, Virtual, June 2020.
- Menghi C, Tsigkanos C, Pelliccione P, et al. (2019) Specification patterns for robotic missions. *IEEE Transactions on Software Engineering* 47(10): 2208–2224.
- Metropolis N, Rosenbluth AW, Rosenbluth MN, et al. (1953) Equation of state calculations by fast computing machines. *The journal of chemical physics* 21(6): 1087–1092.
- Mueller C, Venix J and Hayes B (2018) *Robust robot learning from demonstration and skill repair using conceptual constraints*. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, October 2018. IEEE, pp. 6029–6036.
- Ng AY and Russell SJ (2000) Algorithms for inverse reinforcement learning. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 663–670. ISBN 1-55860-707-2. URL <http://dl.acm.org/citation.cfm?id=645529.657801>
- Niekum S, Osentoski S, Konidaris G, et al. (2015) Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research* 34(2): 131–157.
- Ordóñez FJ and Roggen D (2016) Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16(1). URL. DOI: [10.3390/s16010115](https://doi.org/10.3390/s16010115). <http://www.mdpi.com/1424-8220/16/1/115>
- Pérez-D'Arpino C and Shah JA (2017) *C-learn: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy*. In: 2017 IEEE International Conference on Robotics and Automation, Singapore, June 2017. IEEE, pp. 4058–4065.
- Pnueli A (1977) *The temporal logic of programs*. In: 18th Annual Symposium on Foundations of Computer Science, Providence, RI, Oct 1977. IEEE, pp. 46–57.
- Raman V, Donzé A, Sadigh D, et al. (2015) *Reactive synthesis from signal temporal logic specifications*. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, Seattle, WA, USA, April 2015. ACM, pp. 239–248.
- Rana MA, Mukadam M, Ahmadzadeh SR, et al. (2018) *Learning generalizable robot skills from demonstrations in cluttered environments*. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, October 2018. IEEE, pp. 4655–4660.
- Ranchod P, Rosman B and Konidaris G (2015) Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany, September 2015, pp. 471–477. DOI: [10.1109/IROS.2015.7353414](https://doi.org/10.1109/IROS.2015.7353414)
- Ravichandar H, Polydoros AS, Chernova S, et al. (2020) Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems* 3: 297–330.
- Sanneman L, Fourie C and Shah JA (2021) The state of industrial robotics: Emerging technologies, challenges, and key research directions. *Foundations and Trends® in Robotics* 8(3): 225–306. URL. DOI: [10.1561/23000000065](https://doi.org/10.1561/23000000065)
- Schaal S (2006) *Dynamic movement primitives—a framework for motor control in humans and humanoid robotics*. *Adaptive Motion of Animals and Machines*. Tokyo, Japan: Springer, 261–280.
- Shah A, Kamath P, Shah JA, et al. (2018) Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems* 31: 3804–3813.
- Shepard R (1987) Toward a universal law of generalization for psychological science. *Science* 237(4820): 1317–1323. URL. DOI: [10.1126/science.3629243](https://doi.org/10.1126/science.3629243). <http://science.sciencemag.org/content/237/4820/1317>
- Silver T, Allen KR, Lew AK, et al. (2020) Few-shot Bayesian imitation learning with logical program policies. *Proceedings of the AAAI Conference on Artificial Intelligence* 34: 10251–10258.
- Sobti S, Shome R and Kavraki LE (2023) Efficient inference of temporal task specifications from human demonstrations using experiment design. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, May 2023, pp. 9764–9770. DOI: [10.1109/ICRA48891.2023.10160692](https://doi.org/10.1109/ICRA48891.2023.10160692)
- Sohn K, Lee H and Yan X (2015) Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems* 28: 3483–3491.
- Tenenbaum JB (1999) *A Bayesian framework for concept learning*. PhD Thesis. Massachusetts Institute of Technology.
- Tenenbaum JB (2000) Rules and similarity in concept learning. *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 59–65.
- Toris R, Kent D and Chernova S (2015) Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing. In: 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, May 2015, pp. 4504–4510. DOI: [10.1109/ICRA.2015.7139823](https://doi.org/10.1109/ICRA.2015.7139823).
- Unhelkar VV and Shah JA (2019) Learning models of sequential decision-making with partial specification of agent behavior. *Proceedings of the AAAI Conference on Artificial Intelligence* 33: 2522–2530.
- Vardi MY (1996) An automata-theoretic approach to linear temporal logic. *Logics for Concurrency*. Berlin, Heidelberg, Germany: Springer, 238–266.
- Vaswani A, Shazeer N, Parmar N, et al. (2017) Attention is all you need. *Advances in Neural Information Processing Systems* 30. NY, USA: Curran Associates Inc.
- Vazquez-Chanlatte M, Jha S, Tiwari A, et al. (2018) Learning task specifications from demonstrations. *Advances in Neural Information Processing Systems* 31: 5368–5378.
- Xu Z, Ornik M, Julius AA, et al. (2019) Information-guided temporal logic inference with prior knowledge. In: *2019 American Control Conference 2019*, Philadelphia, PA, July 2019. IEEE, pp. 1891–1897.
- Ziebart BD, Maas AL, Bagnell JA, et al. (2008) Maximum entropy inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 8: 1433–1438.