Supervised Bayesian Specification Inference from Demonstrations

Ankit Shah[1], Pritish Kamath[1], Shen Li[1], Patrick Craven[2], Kevin Landers[2], Kevin Oden[2] and Julie Shah[1]

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.

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 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. (2012); Goodman and Stuhlmüller (2014) are key to our modeling approach; indeed, probabilistic programming languages enabled Ellis et al. (2017, 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 incorporate a synthetic domain and a real-world task involving setting a dinner table, both of which are representative of candidate tasks for robots to learn from demonstrations. For both these domains, the ground-truth specifications are known, and we report the capability

---

[1]CSAIL, MIT
[2]Lockheed Martin Corporation

Corresponding author:
Ankit Shah Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambrdige, MA 02139, USA.
Email: ajshah@mit.edu

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 aircrafts. The LFE domain is particularly challenging, as it incorporates multiple decision-making participants, some of which act cooperatively and some in an adversarial fashion. 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 present an advancement of that work and apply the model to newer evaluation domains. First, we extend the probabilistic model to be capable of learning both inductively (from positive examples only) and from positive and negative examples. We also extend the evaluation presented by Shah et al. (2018) to include the large-force exercise domain.

Related Work

One common approach discussed in prior research frames learning from demonstration as an inverse reinforcement learning (IRL) problem. Ng and Russell (2000) and Abbeel and Ng (2004) first formalized the problem of inverse reinforcement learning as one of optimization in order to identify the reward function that best explains observed demonstrations. Ziebart et al. (2008) introduced algorithms to compute optimal policy for imitation using the maximum entropy criterion. Konidaris et al. (2012) and Niekum et al. (2015) framed IRL in a semi-Markov setting, allowing for implicit representation of the temporal structure of the task. Surveys by Argall et al. (2009) and Chernova and Thomaz (2014) provided a comprehensive review of techniques built upon these works as applied to robotics. However, according to Arnold et al. (2017), one drawback of inverse reinforcement learning is the non-triviality of extracting task specifications from a learned reward function or policy. Our method bridges this gap by directly learning the specifications for acceptable execution of a given task.

Temporal logics, introduced by Pnueli (1977), are an expressive grammar used to describe the desirable temporal properties of task execution. Temporal logics have previously served as a language for goal definitions in reinforcement learning algorithms (Li et al. (2017) and Littman et al. (2017)), reactive controller synthesis (Kress-Gazit et al. (2009)) and Raman et al. (2015)), and domain-independent planning (Kim et al. (2017)).

Kasenberg and Scheutz (2017) explored mining globally persistent specifications from optimal traces of a finite-state Markov decision process (MDP). Jin et al. (2015) proposed algorithms for mining temporal specifications similar to rise and setting times for closed-loop control systems. Works by Kong et al. (2014), Kong et al. (2017), Yoo and Belta (2017), and Lemieux et al. (2015) are most closely related to our own, as our work incorporates only the observed state variable (and not the actions of the demonstrators)

as input to the model. Lemieux et al. (2015) introduced Texada, a general-specification mining tool for software logs. Texada outputs all possible satisfied instances of a particular formula template; however, it treats each time step as a string, with all unique strings within the log treated as unique propositions. Texada would treat a system with n propositions as a system with 2^n distinct propositions; thus, interpreting a mined formula is non-trivial. Kong et al. (2014), Kong et al. (2017), and Yoo and Belta (2017) mined PSTL specifications for given demonstrations while simultaneously inferring signal propositions akin to our own user-defined atomic propositions by conducting breadth-first search over a directed acyclic graph (DAG formed by candidate formulas. Our prior specifications allow for better connectivity between different formulas, while using MCMC-based approximate inference enables fixed runtimes.

We adopt a fully Bayesian approach to model the inference problem, allowing our model to maintain a posterior distribution over candidate formulas. This distribution provides a measure of confidence when predicting the acceptability of a new demonstration that the aforementioned approaches do not.

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 $[\boldsymbol{\alpha}]$ of the set of propositions, $\boldsymbol{\alpha}$. The notation $[\boldsymbol{\alpha}], t \models \varphi$ indicates that φ holds at time t. The trace $[\boldsymbol{\alpha}]$ satisfies φ (denoted as $[\boldsymbol{\alpha}] \models \varphi$) iff $[\boldsymbol{\alpha}], 0 \models \varphi$. The minimal syntax of LTL can be described as follows:

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

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 $t \geq t_1$.

Bayesian Specification Inference

A large number of tasks comprised of multiple subtasks can be represented by a combination of three temporal behaviors among those defined by Dwyer et al. (1999) — namely, global satisfaction of a proposition, eventual completion of a subtask, and temporal ordering between subtasks. With φ_{global}, $\varphi_{eventual}$, and φ_{order} representing LTL formulas for these behaviors, the task specification is written as follows:

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

We represent the task demonstrations as an observed sequence of state variables, $[\boldsymbol{x}]$. Let $\boldsymbol{\alpha} in \{0, 1\}^n$ represent a

vector of finite dimension formed by n Boolean propositions. The propositions are related to the state variables through a labeling function, $\boldsymbol{\alpha} = f(\boldsymbol{x})$, 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 $\boldsymbol{D} = \{([\boldsymbol{\alpha}]_i, y_i) \; ; \; i \in \{1, 2, \ldots, n_{demo}\}\}$ consists of n_{demo} demonstrations along with the label. The output, again, is a probability distribution $P(\varphi|\boldsymbol{D})$.

Formula Template

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

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

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

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

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

Temporal ordering: Every set of feasible ordering constraints over a set of subtasks is mapped to a DAG over nodes representing these subtasks. Each edge in the DAG corresponds to a binary precedence constraint. Let $\boldsymbol{W_2}$ be the set of binary temporal orders defined by $\boldsymbol{W_2} = \{(w_1, w_2) : w_1 \in \boldsymbol{V}, w_2 \in \text{Descendants}(w_1)\}$, where $\boldsymbol{V}$ is the set of all nodes 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:

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

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

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

Specification Learning as Bayesian Inference

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

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

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

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

Algorithm 1 SampleSetsOfLinearChains

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

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.

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

Table 1. Prior definitions and hyperparameters.

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

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 $1 - p_{part}$).

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 $\mathbf{P}$, 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 ith 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 $\mathbf{P}$, 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 $\{1, 2, \ldots, \mathcal{F}.\text{length} + 1\}$ 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 (1985)). If the outcome of the draw is $\mathcal{F}$.length + 1, then a new tree with root node a is created in line 10; otherwise, InsertIntoForest is called recursively to add a to the forest $\mathcal{F}[i]$.descendants, as per line 12.

Algorithm 2 SampleForestofSubtasks

```
1: function SAMPLEFORESTOFSUBTASKS(Ω, N_new)
2:     P ← random permutation(Ω)
3:     F ← []
4:     for a ∈ P do
5:         F = InsertIntoForest(F, a)
6:     return F
7: function INSERTINTOFOREST(F, a)
8:     i ← Categorical([F[1].size, F[2].size, . . . , F[F.length].size, N_new])
9:     if i = F.length + 1 then
10:         Create new tree F[F.length + 1].root = a
11:     else
12:         F[i].descendants = InsertIntoForest(F[i].descendants, a)
13:     return F
```

Three prior distributions based on the four probabilistic programs are described in Table 1. In all the priors, φ_{global} and $\varphi_{eventual}$ are sampled using SampleSubset($\mathbf{T}, p_G$) and SampleSubset($\mathbf{\Omega}, p_E$), respectively.

Likelihood function The likelihood distribution, $P(\{[\boldsymbol{\alpha}]_i\} \mid \varphi, \{y_i\})$, 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:

$$P(\{[\boldsymbol{\alpha}]_i\} \mid \varphi, \{y_i\}) = \prod_{i \in \{1,2,\ldots,n_{demo}\}} P(\varphi) P([\boldsymbol{\alpha}]_i \mid \varphi, y_i).$$

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 (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 $\boldsymbol{\tau_1}, \boldsymbol{W_{1_1}}$, and $\boldsymbol{W_{2_1}}$ 1) and φ_2 (defined by subsets $\boldsymbol{\tau_2}, \boldsymbol{W_{1_2}}$, and $\boldsymbol{W_{2_2}}$) the likelihood odds ratio if $y_i = 1$ is defined as follows:

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

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

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

We must define likelihood functions for both acceptable and unacceptable demonstrations. Note that the likelihood function defined by Equation 7 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:

$$\frac{P([\boldsymbol{\alpha}]_i \mid \varphi_1)}{P([\boldsymbol{\alpha}]_i \mid \varphi_2)} = \begin{cases} \frac{2^{N_{conj_1}}(2^{N_{conj_2}}-1)}{2^{N_{conj_2}}(2^{N_{conj_1}}-1)} & , [\boldsymbol{\alpha}] \nvDash \varphi_2 \\ \frac{2^{N_{conj_1}}}{(2^{N_{conj_1}}-1)\epsilon} & , [\boldsymbol{\alpha}] \models \varphi_2 \end{cases} \quad (9)$$

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

$$P([\boldsymbol{\alpha}] \mid \varphi) = \begin{cases} 1 - \epsilon, & \text{if } [\boldsymbol{\alpha}] \nvDash \varphi \\ \epsilon, & \text{Otherwise} \end{cases} \quad (10)$$

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

Inference We implemented our probabilistic model in webppl (Goodman and Stuhlmüller (2014)), a Turing-complete probabilistic programming language. The hyper-parameters, including those defined in Table 1 and ϵ, were set as follows: $p_E, p_G = 0.8$; $p_{part} = 0.3$; $N_{new} = 5$; $\epsilon = 4 \times log(2) \times (|\boldsymbol{T} + |\boldsymbol{\Omega}| + 0.5|\boldsymbol{\Omega}|(|\boldsymbol{\Omega}| - 1))$. These values were held constant for all evaluation scenarios. The equation for ϵ was defined such that evidence of a single non-satisfying demonstration would negate the contribution of four satisfying demonstrations to the posterior probability. The posterior distribution of candidate formulas is constructed using webppl's Markov chain Monte Carlo (MCMC) sampling algorithm from 10,000 samples, with 100 samples 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.

Evaluations

We evaluated the performance of our model across three different domains. We developed a synthetic domain with a low dimensional state-space where we could easily vary the complexity of the ground-truth specifications. We also applied our model to a real-world task of setting a dinner table – a task often incorporated into studies of learning from demonstrations (Toris et al. (2015)). This task has a large raw state-space incorporating the poses of the objects included in the domain. This domain demonstrates the benefits of using propositions to represent task specifications, as the complexity of the problem depends upon the number of Boolean propositions and not the dimensionality of the raw state-space. (Note that the ground-truth specifications are known in both of these domains, and it is easy to measure the quality of the solution by comparing it to the ground-truth specification.)

Finally, we also applied our inference model to the large-force exercise (LFE) domain. Large-force exercises are simulated air-combat games used to train combat pilots. We developed simulation environments using joint semi-automated forces (JSAF), a constructive environment for generating examples of LFE executions, and used our model to infer specifications for successful completion of mission objectives. In this domain, the true specifications are not known, and we only have annotations of the demonstrated scenario from a subject matter expert (in this case, the mission commander who designs the scenario and debriefs participating pilots).

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 $\boldsymbol{\tau}^*$, $\boldsymbol{W_1^*}$, and $\boldsymbol{W_2^*}$ (as per Equations 3, 4, and 5), and a candidate formula φ is defined by subsets $\boldsymbol{\tau}$, $\boldsymbol{W_1}$, and $\boldsymbol{W_2}$. In such cases, we define the degree of similarity using the Jaccard index (Paul (1912)) as follows:

$$L(\varphi) = \frac{\mid \{\boldsymbol{\tau}^* \cup \boldsymbol{W_1^*} \cup \boldsymbol{W_2^*}\} \cap \{\boldsymbol{\tau} \cup \boldsymbol{W_1} \cup \boldsymbol{W_2}\} \mid}{\mid \{\boldsymbol{\tau}^* \cup \boldsymbol{W_1^*} \cup \boldsymbol{W_2^*}\} \cup \{\boldsymbol{\tau} \cup \boldsymbol{W_1} \cup \boldsymbol{W_2}\} \mid} \quad (11)$$

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 $\boldsymbol{W_2}$ 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.

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 $\boldsymbol{x}$ represents the position of that agent within the task space.

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

$$\tau_i = \begin{cases} \text{true,} & \|\boldsymbol{x} - \boldsymbol{x}_{T_i}\| \geq \epsilon_{threat} \\ \text{false,} & \text{otherwise} \end{cases} \quad (12)$$

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

Let $\boldsymbol{\Omega} = \{1, 2, \ldots, n_{POI}\}$ represent the set of POIs positioned at $\boldsymbol{x}_{P_i} \ \forall \ i \in \boldsymbol{\Omega}$. A proposition ω_i is associated with each POI such that:
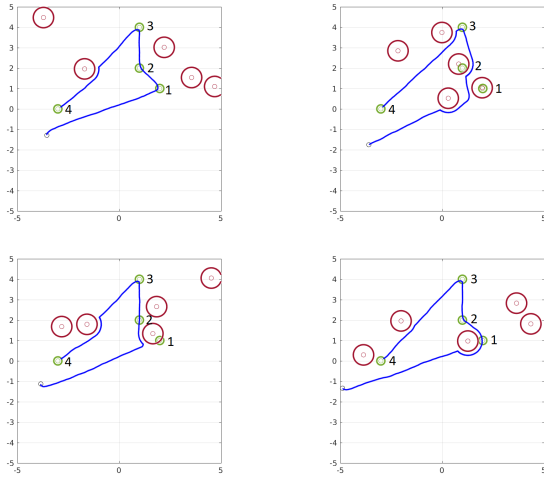
Figure 1. Example trajectories from Scenario 1. Green circles denote the POIs; red circles denote the avoidance zones of threats.

$$\omega_i = \begin{cases} \text{true}, & \|\boldsymbol{x} - \boldsymbol{x}_{P_i}\| \leq \epsilon_{POI} \\ \text{false}, & \text{otherwise} \end{cases} \tag{13}$$

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

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

$$\pi_i = \begin{cases} \text{false}, & \exists \, j \text{ such that } \|\boldsymbol{x}_{P_i} - \boldsymbol{x}_{T_j}\| \leq \epsilon_{threat} \\ \text{true}, & \text{otherwise} \end{cases} \tag{14}$$

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 $\boldsymbol{T} \subseteq \boldsymbol{\tau}$ that represents the subset of threats that the agent must avoid; a set $\boldsymbol{W_1} \subseteq \boldsymbol{\Omega}$ that represents the subset of POIs the agent must visit; and the ordering constraints defined by $\boldsymbol{W_2}$, a set of feasible pairwise precedence constraints between the POIs.
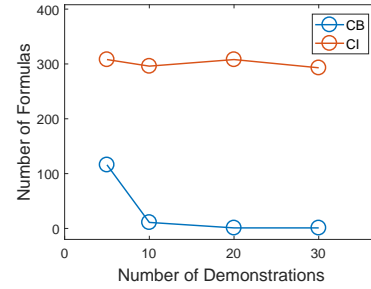
Here, we demonstrate the results of applying our inference model to three scenarios with differing ground-truth specifications.

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 1.

The posterior distribution was computed using prior 1 (defined in Table 1), with both CB (Equation 7) and CI (Equation 8) likelihood functions. The expected and maximum values among the top 5 a posteriori formula candidates of $L(\varphi)$ are depicted in Figure 2. We observed that the CB likelihood function performed better than the CI likelihood function at inferring the complete specification.
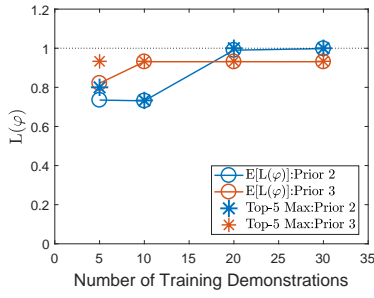

(a) Scenario 1

(b) Scenario 1

Figure 2. Figure 2a depicts the results from Scenario 1, with the dotted line representing the maximum possible value of $L(\varphi)$. Figure 2b shows the number of unique formulas in the posterior distribution
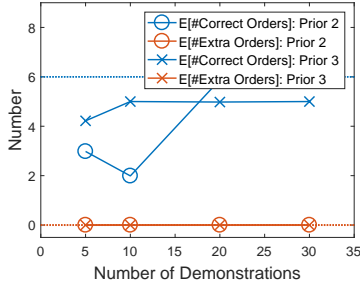
Using the CI function resulted in a higher posterior probability assigned to formulas with high prior probability that were satisfied by all demonstrations. (These tended to be simple, non-informative formulas; the CB function assigned higher probability mass to more-complex formulas that explained the demonstrations correctly.) Figure 2b depicts the number of unique formulas in the posterior distributions. The CB likelihood function resulted in posteriors being more peaky, with fewer unique formulas as 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 3a. 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 3b 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.

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 4.
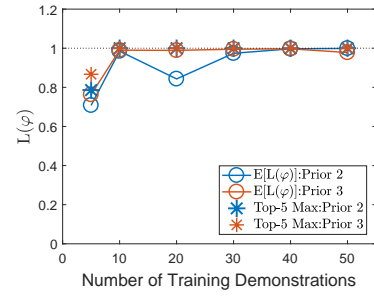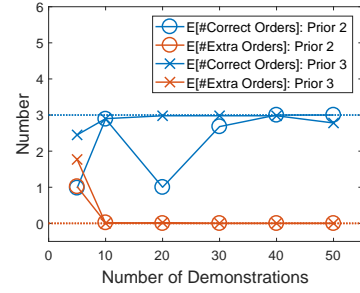
(a) Scenario 1

(b) Scenario 1

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

Figure 4. Example trajectories from Scenario 2. Green circles denote the POIs; red circles denote the avoidance zones of threats.

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 5a. 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 5b 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,

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

(b) Scenario 2 orders

Figure 5. Figure 5a indicates the $L(\varphi)$ values for Scenario 2, and Figure 5b depicts the correct and extra orderings inferred in Scenario 2. The dotted lines represent the number of orderings in the true specification.
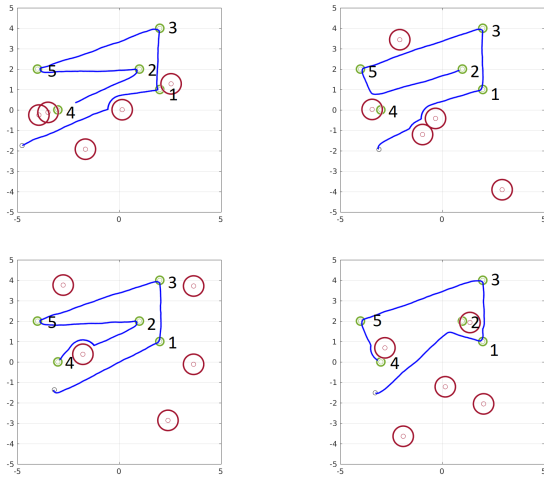
but both priors converged to the correct specification given enough training examples.

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 6 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 7a. 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 7b. 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.

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

TempLogIn (Kong et al. (2017)) required 33 minutes 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-hour runtime cutoff.
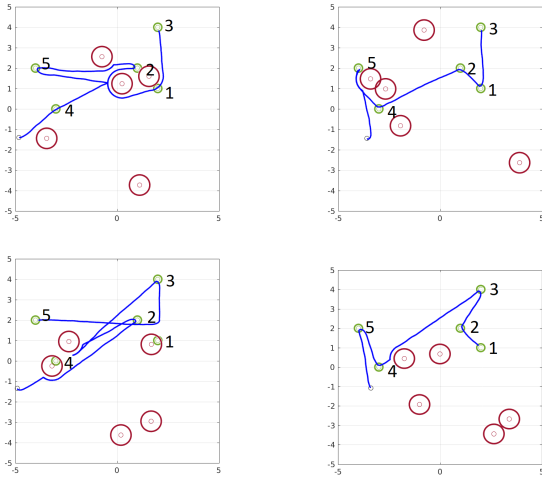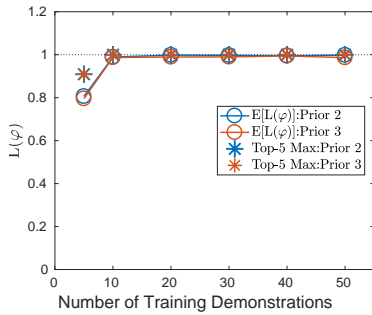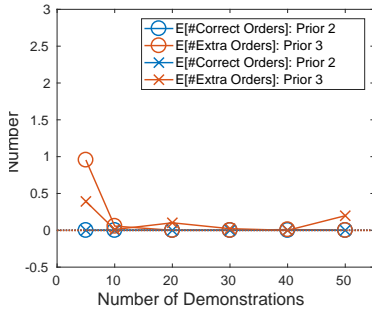
Figure 6. Example trajectories from Scenario 3. The green circles denote the POIs; the red circles denote the threat avoidance zones.

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

(b) Scenario 2 orders

Figure 7. Figure 7a indicates the $L(\varphi)$ values for Scenario 3, and Figure 7b depicts the correct and extra orderings inferred in Scenario 3. The dotted lines represent the number of orderings in the true specification.

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 $n + 1$.

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 8a 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.

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 $\boldsymbol{\Omega}$. Each piece was tracked with a motion-capture system over the course of the demonstration, with the pose of an object $i \in \boldsymbol{\Omega}$ in the world frame represented by $\boldsymbol{T}_i^O$. In addition, the pose of the wrists of the demonstrators $\boldsymbol{T}_{h1}^O$ and $\boldsymbol{T}_{h2}^O$ 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 $\boldsymbol{T}_{w_i}^O$ represents the transform between the origin frame and the TSR frame for the object, i. The bounds for $\boldsymbol{B}_i$ represent the translation and rotational tolerances of the constraint. Finally, $\boldsymbol{P}_i$ 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 $\boldsymbol{T}_i^{w_i} = (\boldsymbol{T}_{w_i}^O)^{-1} \boldsymbol{T}_i^O$. A proposition ω_i is associated with object i as follows:

$$\omega_i = \begin{cases} \text{true,} & \boldsymbol{T}_i^{W_i} \in \boldsymbol{P}_i \\ \text{false.} & \text{otherwise} \end{cases} \quad (15)$$

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 $\boldsymbol{T}_c^O$ represents the pose of the centerpiece with respect to the world frame, and the bounds of the constraint are defined by $\boldsymbol{B}_c$, with $\boldsymbol{P}_c$ 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 $\boldsymbol{T}_{h_i}^c$ for $i \in \{1, 2\}$. A proposition τ_c is associated with the centerpiece, and is defined as follows:

$$\tau_c = \begin{cases} \text{false,} & \boldsymbol{T}_{h1}^c \in \boldsymbol{P}_c \vee \boldsymbol{T}_{h2}^c \in \boldsymbol{P}_c \\ \text{true,} & \text{otherwise} \end{cases} \quad (16)$$

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

Finally, condition propositions $\pi_i \, \forall \, i \in \boldsymbol{\Omega}$ 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 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).
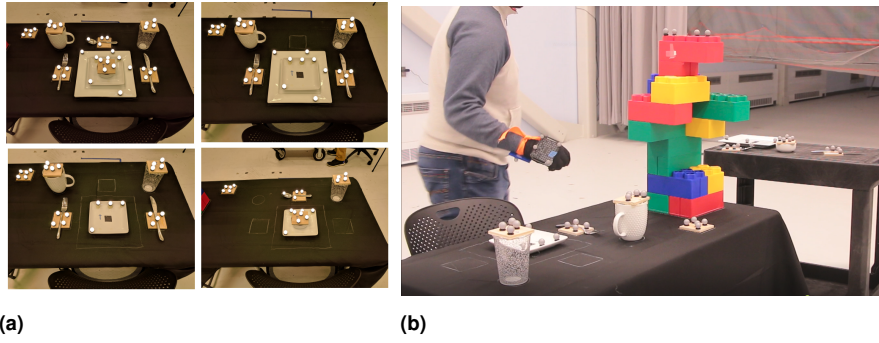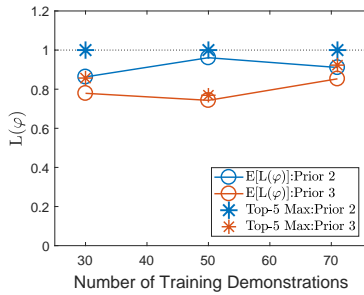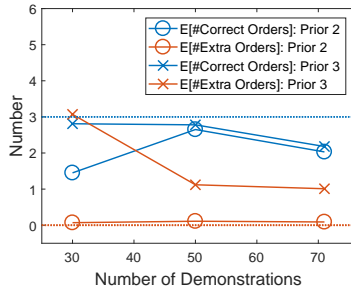
Figure 8. Figure 8a depicts all the final configurations. Figure 8b depicts the demonstration setup. (Photographed by the authors in April 2017.)

(a) Dinner table $L(\varphi)$

(b) Dinner Table orders

Figure 9. Figure 9a depicts the $L(\varphi)$ values for the dinner table domain, with the dotted line representing the maximum possible value. Figure 9b depicts the correct and extra orderings inferred within this domain; the dotted lines represent the number of orderings in the true specification.

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 four of the 71 demonstrations.

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 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.

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 10. 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.

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, 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
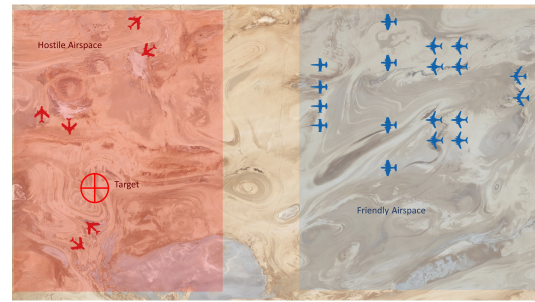

Figure 10. The starting configuration of a large-force exercise scenario. The red aircrafts are the hostile forces, and the blue are friendly forces.

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.

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 a recurrent neural network as the underlying architecture.

1. **Stand-alone:** Here, the recurrent neural network is trained to jointly optimize the binary cross-entropy for 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 have shown state-of-the-art performance during time-series classification tasks (Ordóñez and Roggen (2016)). These models – henceforth referred to as 'LSTM' and 'Bi-LSTM,' 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 of the mission objectives 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 and their bidirectional counterparts, and are referred to as 'LSTM Coupled' and 'Bi-LSTM Coupled,' respectively. These models were trained using the propositions and collected flight phase data.

Evaluations The classification models were evaluated through a four-fold cross-validation wherein the training

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

Classifier	MO1	MO2	MO3
LSTM	0.533	0.533	0.481
Bi-LSTM	0.533	0.533	0.481
LSTM Coupled	0.533	0.533	0.481
Bi-LSTM Coupled	0.533	0.533	0.481
BSI (Prior 2)	**0.674**	**0.712**	**0.877**
BSI (Prior 3)	**0.674**	0.676	**0.877**

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. The predictions of the model for each of the scenarios were assimilated at the end. 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.

Results As presented in Table 2, our model outperformed RNN-based supervised learning models. With a four-fold split of training and test data, prior 2 seemed to outperform prior 3; one possible explanation would be that the bias of prior 3 toward longer task chains might result in a higher rate of false negatives.

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, 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.

Next, 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:

1. **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 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.

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.

Lockheed Martin corporation for his expertise in modeling and simulation.

References

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 1.

Aldous DJ (1985) Exchangeability and related topics. In: Hennequin PL (ed.) *École d'Été de Probabilités de Saint-Flour XIII — 1983*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-39316-0, p. 92.

Argall BD, Chernova S, Veloso M and Browning B (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*.

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.

Chernova S and Thomaz AL (2014) Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(3): 1–121.

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*. ACM, pp. 411–420.

Ellis K, Ritchie D, Solar-Lezama A and Tenenbaum JB (2017) Learning to infer graphics programs from hand-drawn images. *arXiv preprint arXiv:1707.09627* .

Ellis K, Solar-Lezama A and Tenenbaum J (2015) Unsupervised learning by program synthesis. In: *Advances in neural information processing systems*. pp. 973–981.

Freer CE, Roy DM and Tenenbaum JB (2014) Towards common-sense reasoning via conditional simulation: legacies of turing in artificial intelligence. In: Downey R (ed.) *Turing's Legacy*, *Lecture Notes in Logic*, volume 42. Cambridge University Press. ISBN 9781107338579, pp. 195–252.

Goodman N, Mansinghka V, Roy DM, Bonawitz K and Tenenbaum JB (2012) Church: a language for generative models. *arXiv preprint arXiv:1206.3255* .

Goodman ND and Stuhlmüller A (2014) The Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Accessed: 2018-4-9.

Graves A, Fernández S and Schmidhuber J (2005) Bidirectional lstm networks for improved phoneme classification and recognition. In: *International Conference on Artificial Neural Networks*. Springer, pp. 799–804.

Hochreiter S and Schmidhuber J (1997) Long short-term memory 9: 1735–80.

Jin X, Donzé A, Deshmukh JV and Seshia SA (2015) Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(11): 1704–1717.

Kasenberg D and Scheutz M (2017) Interpretable apprenticship learning with temporal logic specifications. *arXiv preprint arXiv:1710.10532* .

Kim J, Banks CJ and Shah JA (2017) Collaborative planning with encoding of users' high-level strategies. In: *AAAI*. pp. 955–962.

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.

Kong Z, Jones A, Medina Ayala A, Aydin Gol E and Belta C (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.

Konidaris G, Kuindersma S, Grupen R and Barto A (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 (1951) A philosophical essay on probabilities, translated from the 6th french edition by frederick wilson truscott and frederick lincoln emory.

Lemieux C, Park D and Beschastnikh I (2015) General ltl specification mining (t). In: *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. 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*. IEEE, pp. 3834–3839.

Littman ML, Topcu U, Fu J, Isbell C, Wen M and MacGlashan J (2017) Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341* .

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

Niekum S, Osentoski S, Konidaris G, Chitta S, Marthi B and Barto AG (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). DOI:10.3390/s16010115. URL http://www.mdpi.com/1424-8220/16/1/115.

Paul J (1912) The distribution of the flora in the alpine zone.1. *New Phytologist* 11(2): 37–50. DOI:10.1111/j.1469-8137.1912.tb05611.x.

Pnueli A (1977) The temporal logic of programs. In: *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, pp. 46–57.

Raman V, Donzé A, Sadigh D, Murray RM and Seshia SA (2015) Reactive synthesis from signal temporal logic specifications. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, pp. 239–248.

Shah A, Kamath P, Li S and Shah J (2018) Bayesian inference of temporal task specifications from demonstrations. In: *NIPS (to appear)*.

Shepard R (1987) Toward a universal law of generalization for psychological science. *Science* 237(4820): 1317–1323. DOI: 10.1126/science.3629243.

Sohn K, Lee H and Yan X (2015) Learning structured output representation using deep conditional generative models. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M and Garnett R

(eds.) *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 3483–3491.

Tenenbaum JB (2000) Rules and similarity in concept learning. In: *Advances in neural information processing systems*. pp. 59–65.

Toris R, Kent D and Chernova S (2015) Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 4504–4510.

Yoo C and Belta C (2017) Rich time series classification using temporal logic. In: *Robotics: Science and Systems*.

Ziebart BD, Maas AL, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI*, volume 8. Chicago, IL, USA, pp. 1433–1438.