Interactive Robot Training for Complex Tasks

by

Ankit Jayesh Shah
B.Tech., Indian Institute of Technology Bombay (2013)
S.M., Massachusetts Institute of Technology (2016)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Aeronautics and Astronautics
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
August 2021

Author . . . . . .
Department of Aeronautics and Astronautics
August 17, 2021

Certified by . . . . . .
Julie A. Shah
Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Certified by . . . . . .
Nicholas Roy
Bisplinghoff Professor of Aeronautics and Astronautics, MIT
Thesis Committee Member

Certified by . . . . . .
Hadas Kress-Gazit
Geoffrey S.M. Hedrick Sr. Professor, Sibley School of Mechanical and
Aerospace Engineering, Cornell University
Thesis Committee Member

Certified by . . . . . .
George Konidaris
John E. Savage Assistant Professor of Computer Science, Brown University
Thesis Committee Member

Accepted by . . . . . .
Jonathan How
Chairman, Department Committee on Graduate Theses

Interactive Robot Training for Complex Tasks

by

Ankit Jayesh Shah

Submitted to the Department of Aeronautics and Astronautics
on August 17, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Aeronautics and Astronautics

Abstract

Domains such as high-mix manufacturing, domestic robotics, space exploration, etc., are key areas of interest for robotics. In these domains, it is difficult to anticipate the exact role of the robot apriori, therefore defining the robot specifications is challenging. This presents a crucial hurdle to widespread adoption of robots in these domains. Developing robots that can be re-programmed easily during deployment by domain experts, through the modification of the task specifications, without requiring extensive programming knowledge is a key research thrust of this dissertation.

I present a multi-modal framework for training a robot through demonstrations and acceptability assessments provided by the teacher as per their intended task specification. I adopt an online Bayesian approach, where the robot maintains a belief over the teacher's intended task specification, and each input provided by the teacher iteratively updates the robot's belief. Further, I enabled the robot to infer task specifications that require satisfaction of temporal properties by utilizing a well-defined fragment of linear temporal logic (LTL). Towards developing this framework, I address three key research questions.

I begin by presenting a novel approach to inferring formal temporal specifications from labeled task executions, called Bayesian specification inference. This approach can learn tasks expressed by an expressive but relevant fragment of LTL while modeling the ambiguity of demonstrations as a belief distribution over candidate LTL formulas. We demonstrate the utility of this approach in inferring task specifications for the representative multi-step manipulation task of setting a dinner table. We also utilize this model to learn an assessment model for multi-aircraft combat missions that shows a high degree of alignment with the assessments provided by a domain expert.

Next, I present planning with uncertain specifications (PUnS), a novel formulation that enables planning with a belief distribution over the true specification. I propose four evaluation criteria that capture the semantics of satisfying a belief over logical formulas and demonstrate the existence of an equivalent Markov decision process (MDP) for every instance of a PUnS problem. We show that the robot policies produced through the PUnS formulation demonstrate flexibility by generating distinct valid task executions and result in a low error rate by simultaneously satisfying a maximal subset of the specifications in the belief distribution.

Finally, I present an integrated specification inference framework that interleaves inference and planning through active learning. Our models for active learning allow the robot to identify whether a task demonstration or an assessment of its task execution provided by the teacher would be most beneficial in refining its belief. Further, we developed algorithms that enable the robot to identify and perform the task execution that would be most informative in refining its uncertainty. We explore the impact of different information utility functions and the degree of teacher's pedagogical selectivity on the robot's learning performance, and demonstrate that allowing the robot to select the ideal learning modality allows it to overcome the limitations of a non-pedagogical teacher, and still converge to the true task specification. We also demonstrate our framework through a study involving users teaching a robot to set a dinner table with only five task executions.

Thesis Supervisor: Julie A. Shah
Title: Professor of Aeronautics and Astronautics, MIT

Thesis Committee Member: Nicholas Roy
Title: Bisplinghoff Professor of Aeronautics and Astronautics, MIT

Thesis Committee Member: Hadas Kress-Gazit
Title: Geoffrey S.M. Hedrick Sr. Professor, Sibley School of Mechanical and Aerospace Engineering, Cornell University

Thesis Committee Member: George Konidaris
Title: John E. Savage Assistant Professor of Computer Science, Brown University

Dissertation Reader: Stefanie Tellex
Title: Associate Professor of Computer Science, Brown University

Dissertation Reader: Vaibhav Unhelkar
Title: Assistant Professor of Computer Science, Rice University

Acknowledgments

As I conclude my journey through graduate school, I would like to thank my advisor Prof. Julie Shah. Ever since the first research meeting I had with her before joining the Interactive Robotics Group, she has empowered me as a researcher to seek out challenging problems and supported me intellectually and materially to make impactful contributions towards them. The autonomy to explore under-studied research areas and her encouragement to think strategically about long-term research goals has made me a stronger researcher. Her work to promote social and ethically responsible computing practices has inspired me to think deeply about the impact of technologies that I will help develop on society at large.

I would like to extend my gratitude to my thesis committee members Prof. Nicholas Roy, Prof. Hadas Kress-Gazit, and Prof. George Konidaris. Nick's research on making robots operate robustly in challenging environments was the inspiration for me to get into robotics in the first place. Through our discussions, he has always motivated me to think about broader problems in the field of robotics as a whole to inform my research direction. Hadas's work on synthesis was what inspired me to learn more about formal logics, and the power of systems that are correct-by-construction. She has taught me to critically appraise my work to carefully identify its scope and limitations. George's work on skill chaining, abstraction selection, and symbol learning has deeply influenced the technical contributions of my research. I am very grateful for the extensive technical discussions that we've had through the years. His attention to detail and careful feedback have helped me frame my thesis contributions with a lot more clarity. I am very grateful to have had the support and guidance of my amazing thesis committee throughout my Ph.D. research.

I am also very grateful to Prof. Michael Littman, who served as an external evaluator for my thesis proposal defense, and Prof. Stefanie Tellex, and Prof. Vaibhav Unhelkar, who served as dissertation readers. Their inputs have been crucial to strengthening the thesis's technical arguments and determining the directions that my research has taken.

I have been extremely fortunate to work with Pritish Kamath, Shen Li, Samir Wadhwania, Patrick Craven, Kevin Oden, and Kevin Landers on the various projects completed during my graduate school career.

I am especially grateful to Julia M. Finn, Elizabeth Zotos, and Beth Marois for their administrative support in navigating everything from equipment purchases to procedures to run user studies to compliance with department and institute policies.

Interactive Robotics Group (IRG) has been a home to me during my time here, and I am fortunate to not only count its members as my colleagues, but also very close friends. A heartfelt thanks to Matthew, Stefanos, Chongjie, Pem, Joe, Vaibhav, Ramya, Claudia, Been, Kyle, Abhi, Jorge, Keren, Alex, Jessie, Brad, Rebecca, Lindsay, Chris, Serena, Mycal, Yilun, Yi-Shiuan, Bilkit, Thavishi, Sarah, Tariq, Nadia, and Heitor. From preparing for quals, reviewing my research, and introducing me to new hobbies, they have enriched my journey through grad school. A special shout out to Pem for always being ready to sail or plan a hike or a ski-trip.

Moving to a new country far away from my old life would have been unthinkable if not for the love and support of many incredible friends. In particular, Saurabh, Shilpa, Chaitanya, Sami, Kairav, Reetik, and Maya have been like family. And while I was away from India, I have always felt the power of aspirations of my family and my teachers and mentors from school and college, driving me to reach my goals.

I would especially like to thank Drasti, my fiance, for being a keel in my life. She has been a wonderful companion, my best friend, and a sounding board to discuss many of life's challenges. I can not ever forget her incredible effort in making the day of my dissertation defense special for me.

Most importantly, I want to thank my parents, Jayesh Shah and Neena Shah, for always believing in me and for teaching me to dare to aspire. From getting me interested in STEM fields as a child to pushing me to strive for excellence, they have molded me into the person I am today.

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

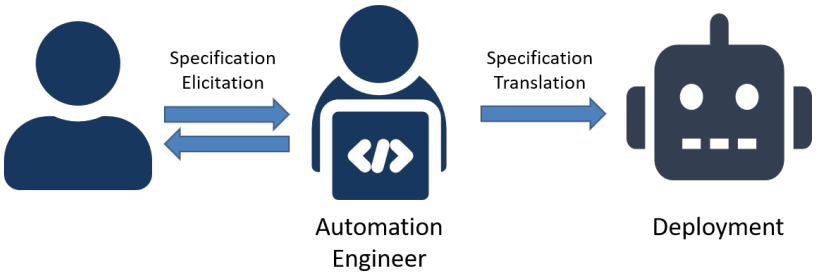
THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

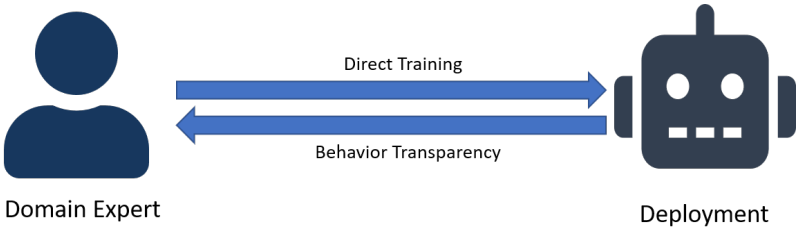
Introduction

The promise of robotics and autonomous systems lies in deploying competent autonomous robots for wide-ranging real-world applications. Tremendous recent progress in robotics hardware capabilities, perception systems, and planning algorithms has made these goals feasible, yet widespread real-world deployment has been elusive. A key feature in many real-world domains is the difficulty in anticipating the autonomous system's actual task during deployment. The problem of task definition and assignment makes expert oversight during deployment valuable, as the task and roles are defined and assigned on the fly. This requisite flexibility is incompatible with the prevalent model of deploying automation, where an automation engineer typically acts as an intermediary between task experts and the robot (Figure 1-1a). This thesis envisions a paradigm of automation where the domain expert directly trains the robot to perform the task and holds the ability to modify robot behavior during deployment (Figure 1-1b), thus addressing the inability to program a robot to perform a new task during deployment.

The guiding principle of this thesis is to empower users who may not have robotics expertise to deploy robotic systems by allowing them to "program the task and not the robot". This concept of allowing users to train and deploy robotic systems directly was identified as one of the key research directions in the comprehensive survey of industrial robotics development by Sanneman, Fourie, and Shah [104]. Programming by demonstration [6, 31, 100] has been a promising approach towards this goal. There is also significant research interest in using inputs such as preference elicitation through active learning [19, 18, 17, 102],



(a)



(b)

Figure 1-1: Paradigms of automation.

corrections [11, 12, 87], or natural language commands [65], as teaching modalities available to the teacher. However, the restriction to Markovian task specifications–specifications that do not depend on the entire history of execution to determine satisfaction–has thus far limited the complexity of tasks that the robot can learn. Enabling easy programming for complex tasks with the temporal constraints widely prevalent in the real world was an essential motivator for our research.

Conversely, prior research into specification mining for temporal tasks has primarily focused on identifying a single valid specification that describes the observed data [67, 66, 141, 133, 129]. In an inductive learning setting where the majority of the data available is acceptable and where limited data is available, multiple candidate specifications accurately describe the observed demonstrations. In such settings, generating a point estimate requires an implicit tie-breaking mechanism that favors one explanation over the others. To mitigate that, we adopt a Bayesian approach inspired by cognitive models for concept learning [125, 123] that maintain a belief over candidate specifications, and any valid hypothesis might gain favorability as more observations become available.

The central contribution of this thesis is the development of a Bayesian framework for robot training through specification inference depicted in Figure 1-2. The framework allows

a robot learner to learn complex tasks through intuitive modalities–namely, demonstrations, remote operations, and acceptability assessments. Further, it will enable the learner to reason about its uncertainty over the task specification it must fulfill. Finally, it allows the learner to guide its learning by selecting when and how to seek information from a domain expert. These capabilities are valuable in real-world settings where the robot's role is challenging to define a priori. Concretely, this thesis addresses the following challenges towards developing the framework depicted in Figure 1-2:

Bayesian specification inference: Formal languages are ideal for unambiguously defining non-Markov task specifications. However, formal languages can be unwieldy for a domain expert compared to providing task demonstrations. On the other hand, a domain expert can readily provide new demonstrations of the task execution, or assess the acceptability of the learner's task execution. However, natural modalities are ambiguous as a set of task executions can be described by many potential formal specifications. Our key contribution was modeling the epistemic uncertainty (uncertainty due to lack of knowledge) caused by this ambiguity as a belief over a set of unambiguous formal specifications within a Bayesian formulation. We used this model to learn task specifications for applications ranging from manipulation tasks to multi-aircraft simulated combat.

We used this model to learn task specifications for applications ranging from multi-aircraft flight missions to multi-step manipulation tasks. The work on identifying mission objectives for multi-aircraft flight missions is at the core of automated mission analysis and review systems developed in collaboration with Lockheed-Martin. Automated evaluation of flight missions generated by our system had over 95% agreement with the assessment provided by an expert mission commander while learning from only 25 labeled examples. In comparison, widely used sequence classification models struggled to perform better than random predictions. Furthermore, the logical structure of the formulas enabled the mission commander to interpret the model's reasoning in case of disagreements. We formally define the problem of Bayesian specification inference in Section 3.2.1.

Planning with uncertain specifications: Prior research into planning with formal specifications has assumed the task specifications to be known unambiguously. However, the inherent epistemic uncertainty over the true task specifications necessitated a novel

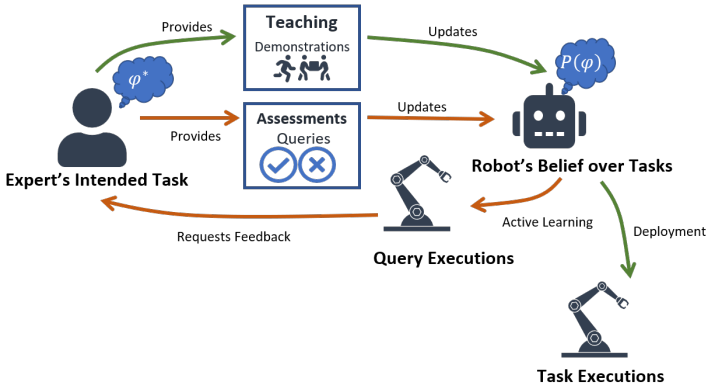


Figure 1-2: Bayesian Framework for Interactive Robot Training

approach. We developed planning with uncertain specifications (PUnS), a novel formalism that allows the robot to reason about its uncertainty over task specifications. Further, we demonstrated that every instance of a PUnS problem is equivalent to a provably minimal Markov decision process making it compatible with MDP planning algorithms.

Admitting non-Markov specifications makes PUnS suitable for multi-step tasks with multiple paths towards task completion commonly found within our domains of interest. For example, in a manufacturing setting, a set of parts is assigned to a station, but the worker decides the actual order of installation; everyday tasks for household assistance such as setting a dinner table, clearing clutter on a table, and stocking the pantry require satisfaction of temporal constraints. We chose the task of setting the dinner table, which retains the temporal complexities of non-Markov specifications while being amenable to a laboratory setting. The task policies computed using the PUnS formulation based on the specifications inferred through just 30 demonstrations were estimated to have a low error rate of $\approx 0.01\%$ while demonstrating multiple unique sequences for setting the table. We have demonstrated the efficacy of PUnS in generating policies that satisfy beliefs over task specifications for a wide range of specifications. We formally define the problem of planning with uncertain specifications in Section 3.2.2.

Online Multi-Modal Robot Training: Our Bayesian framework (Figure 1-2) enables combining specification inference and planning with uncertain specifications to iteratively refine the learner's belief over task specifications. Under this framework, the learner can first decide what learning modality to learn from, i.e., choosing whether to ask the teacher

to demonstrate a task execution or perform a query execution and elicit an acceptability label from the teacher. In developing the framework, we also explored the impact of a pedagogically selective teacher, i.e., a teacher capable of providing teaching data that best guides the learner to the desired task specification. We conducted simulation experiments that demonstrated our proposed active learning framework, characterized the impact of pedagogical selectivity on the learner's performance, and evaluated different fixed and adaptive learning modality schedules over a wide range of ground-truth task specifications.

We also deployed the framework in a user study based on the table-setting task with 18 participants with two different table configuration. All our participants could successfully teach the robot to correctly set the table without errors with just five task executions. The use of belief over logical formulas allowed the participants to train the robot through multiple teaching modalities like physical demonstrations, tele-command of the robot, and acceptability assessments of the robot's task execution. This study provides evidence that active learning for complex non-Markov tasks in a real-world setting like manufacturing is viable in the near future. We formally define the problem of online multi-modal robot training in Section 3.2.3.

1.1 Overview of the Thesis

The remainder of the dissertation is organized as follows: Chapter 2 surveys related research and contextualizes the novelty of our approaches to the challenges described above. Chapter 3 first provides a brief summary of linear temporal logic (LTL) and Markov decision processes (MDP), which are fundamental to the approaches we develop as a part of this dissertation. This is followed by a formal statement of the technical problems formalized and addressed in this dissertation.

Chapter 4 addresses the challenge of Bayesian specification inference described here. The technical content in this chapter draws from the following works:

- Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31*, pages 3804–3813. 2018

- Ankit Shah, Pritish Kamath, Shen Li, Patrick Craven, Kevin Landers, Kevin Oden, and Julie Shah. Supervised Bayesian specification inference from demonstrations. *arXiv preprint arXiv:2107.02912*, 2021

The Bayesian specification inference framework described in Chapter 4 was used as the backend for the mission analysis and review system developed in collaboration with the Lockheed Martin Corporation (Craven et al. [34], [35]). It has also been used to generate contrastive plan explanations in symbolic planning domains in collaboration with Kim et al. [64].

Chapter 5 addresses our approach to planning with uncertain specifications (PUnS), a novel problem formulation that enables reasoning over epistemic uncertainty in task specifications. We developed algorithms to automatically compile any PUnS problem into an equivalent Markov decision process (MDP), thus enabling compatibility with state-of-the-art planning and reinforcement learning algorithms in solving PUnS problems. The technical content in this chapter is based on the following work:

- Ankit Shah, Shen Li, and Julie Shah. Planning with uncertain specifications (PUnS). *IEEE Robotics and Automation Letters*, 2020

Chapter 6 addresses the problem of iteratively refining the learner's belief over task specifications by leveraging both actively sought demonstrations from the teacher and acceptability assessments of the learner's task execution provided by the teacher. The technical content in this chapter is partly derived from the following work:

- Ankit Shah, Samir Wadhwania, and Julie Shah. Interactive robot training for non-Markov tasks. *arXiv preprint arXiv:2003.02232*, 2020

Finally, Chapter 7 summarizes the thesis contributions and the key insights from the research underlying this dissertation. We also describe the promising directions for future work for advancing the theory and practice for empowering task experts to better train robots.

Chapter 2

Motivation

Deploying resilient and reliable robots in real-world domains such as flexible manufacturing floors, space exploration, disaster response, and domestic robotics remains challenging. In addition to unpredictable dynamics, defining the task of the autonomous system apriori is not always possible. Often determining the tasks to be performed requires extensive input from domain experts in an iterative fashion, such as in tactical planning for rover operations described by Smith et al. [120]. The DARPA robotics challenge (DRC) also witnessed competent robotic hardware performing the tasks at a lower autonomy level than anticipated [8, 59, 122]. Successful teams relied on hierarchical supervisory control to accomplish the tasks, but relied on operator expertise to generate high-level task plans. Thus, the cognitive burden of task-planning and adopting to a novel environment fell on the human operator and not the robot. Further, in the field of manufacturing, Sanneman, Fourie, and Shah [104] state that the challenge of adapting a robotic system to perform a new task on a manufacturing line led to difficulties in reusing existing robotic hardware. Further, their study identifies the ease of programming robots as one of the critical enabling technological hurdles towards greater adoption of robots in manufacturing.

In this chapter, we survey the prior work on formalisms for defining task specifications and their expressivity, learning from human experts, paradigms, algorithms for robot decision-making, and the work at the intersection of these fields. Based on the survey of prior research, we identified research opportunities that informed the development of our Bayesian framework (Figure 1-2).

2.1 Temporal Logics for Robot Task Specifications

Successful completion of a task, in general, can be defined as a combination of two notions, namely: maximizing a utility function over a given time horizon (optimizing tasks) and performing within the bounds of given constraints within that horizon (satisfying tasks). In this dissertation, we operate within the framework of satisfying tasks. As satisfaction of constraints is binary, it can be modeled through the use of logical statements. Linear temporal logic is a modal logic with time as the modality first introduced by Pnueli [93]. LTL formulas encode properties of the future path of atomic Boolean propositions used to construct the formula. LTL is expressive enough to satisfy a wide range of temporal properties relevant to robotics applications; however, there are limits to its expressive power, particularly for branching-time properties [14, 33, 43]. Manna and Pnueli proposed a hierarchy of temporal properties expressible in LTL [78].

The relationship between LTL formulas as automata is the basis of most works exploiting temporal logics for control, symbolic planning, or reinforcement learning. Each LTL formula can be translated to a Büchi automaton [132]; the resulting automata also differ qualitatively (determinism, and nature of accepting conditions) based on the class of temporal hierarchy to which a formula belongs [29]. Many useful fragments of LTL, such as GR(1) [92], and syntactic (co-)safe LTL [72, 119] have been studied for their properties that make planning tractable.

Another important line of work is proving the equivalence of and defining the translation between various formalisms for expressing non-Markov task specifications. Recent work by Camacho et al. [26] that defined the relationship between the 'Obligations' class from Manna and Pnueli's temporal hierarchy [78] and the reward machines [129, 128], is of particular relevance to this dissertation. We restrict the formulas considered within this dissertation to a well-defined fragment of LTL belonging to Manna and Pnueli's [78] 'Obligation' class of properties. We demonstrate how a belief over LTL formulas can be compiled into an equivalent reward machine, as opposed to a single formula, thus enabling planning or reinforcement learning algorithms to compute the decision policy.

2.2 Learning from Human Experts

The primary motivation for our research is to empower domain experts to train robotics systems to perform complex tasks directly. The first sub-problem we address is that of learning passively from labeled task executions provided by the expert. We start by reviewing the prior research in robot learning from demonstration, followed by the work in specification mining from software execution traces–an analog of learning from demonstration for software traces. Next, we summarize the progress in probabilistic programming languages for Bayesian inference on arbitrary distributions described through probabilistic programs. This motivates our approach of framing learning from labeled expert demonstrations as Bayesian inference on the concept class of candidate LTL formulas.

2.2.1 Learning from Demonstrations

Argall et al. [6], Chernova et al. [31], and Ravichandar et al. [100] provide a comprehensive survey of the research in LfD applied to robotics. The field can be broadly organized into three major paradigms: imitating task demonstrations, learning task specifications, and learning the modes of interaction with the environment. Each of these approaches is best suited for learning distinct types of robot behaviors.

When LfD is framed as a problem of imitating motion level trajectories, the task objective is to minimize a distance metric between the demonstrated trajectories and the trajectory generated by the learner. This objective may be achieved using techniques such as dynamic motion primitives [105], generalized cylinders [2], or an inference based approach to learning and motion planning [97]. A recent method proposed by Billard et al., [16], and Figueroa et al. [44] leverage dynamical systems theory and non-parametric priors to learn the sequence of stable controllers from demonstration while guaranteeing performance while allowing robot compliance to ensure safe interactions. These approaches are best suited for cloning the demonstrated motions. The objective is to learn motion-level paths for the robot, and therefore, the training curriculum for the robot comprises learning the task one skill at a time.

The next class of LfD models lies at the intersection of policy imitation and specification inference, namely, the problem of inverse reinforcement learning (IRL). Here demonstrations are provided as state-action tuples. Algorithms for IRL are designed to best align the learned policy with the demonstrator's policy in a given system state. The task specification is implicitly encoded through a 'reward' function for performing the task. IRL was first introduced by Ng and Russel [86] and Abbeel and Ng [1] as an optimization problem to identify a reward function that optimally explains the observed demonstrations. Ziebart et al. [143] developed algorithms for computing an estimate of the policy function that satisfies the maximum-entropy criterion. Hadfield-Menell et al. [53] propose a Bayesian approach to reward inference in a Markov setting. Finally, recent works by Chen et al. [30], and Brown et al. [22, 22] utilize ranking information or addition of synthetic noise to generate counterfactual trajectories to learn a reward function that performs better than noisy human demonstrations towards accomplishing the actual task.

These works frame IRL in a setting where the underlying decision process is a Markov decision process (MDP) where the Markov policy only considers the current state in selecting the action. Konidaris et al. [69], Niekum et al. [87], Ranchod et al. [98], and Michini and How [79] attempt to frame the IRL problem in semi-Markov settings, where the policy decided between primitive actions that last for a single time step or temporally abstracted skills that can last for multiple time steps. The notion of temporally abstracted skills is important for learning decompositions of the demonstrated tasks as these skills can be combined in different ways to perform variants of the task. The learned skills might even be a part of performing an entirely different task. This set of approaches relies on representing the task specification through reward functions or policies. Finally, Unhelkar and Shah [131] introduced the Agent Markov Model. This hierarchical approach models demonstrator's policy as piece-wise Markov with discrete latent control modes that are inferred using a non-parametric prior. Indeed Arnold et al. [7] challenge the notion of expressing task specifications as a reward function due to its difficulty in aligning optimal behavior with the user's intention (the problem of value alignment.) However, the choice of a suitable formalism for task specifications remains an open problem.

Some works directly use task specification as an intermediary between the teacher and

the learner. The learner does not directly learn 'how' to perform the task from the teacher but infers a binary function that indicates whether a task execution is successful or not. Final poses of objects [126], hierarchical task networks [54], a finite set of tasks [21], or a sequence of relative poses between the robot and objects [91, 85] are various approaches to represent task specifications. While these works capture the notion of separating the definition of the task from methods of performing the task, the formats used to state the specifications are limited in their expressivity.

Finally, another set of works relevant to the problem of LfD deal with learning the environment interaction models from demonstrations. Konidaris et al. [68] present an approach to construct high-level symbolic representations of continuous domains through exploration. Pasula et al. [90] and Xia et al. [137] present methods for constructing factored transitions models for actions in high-dimensional state spaces. These works provide a crucial link to modeling complex environments in terms of discrete symbols that can then construct a logical specification. While this dissertation focuses on inferring specifications formed through compositions of pre-specified propositions through temporal operators, these works focus on learning the nature of the symbols themselves.

2.2.2 Specification Inference over Temporal Logics

Mining specifications of a software program from its execution traces are analogous to LfD in the software domain. Jha and Seshia [58] provided a theoretical analysis of the problem of inductive synthesis from examples and counterexamples. In the context of software systems, Gabel and Su [47] and Chivilikhin et al. [32] proposed algorithms to mine temporal specifications from execution traces. Lemieux et al. [75] proposed an algorithm to mine all formulas that satisfy a given formula template based on an output log. Here each line of the log is considered a unique proposition. Similarly, Camacho et al. [27], and Kim et al. [64] addressed the problem of learning LTL-based logical classifiers for execution traces.

In the context of sequential decision making, Kasenberg and Scheutz [62] and Xu et al. [140] proposed algorithms to mine properties of decision-making agents acting in a Markov decision process. Our approach intends to infer specifications based only on the observed states rather than state-action tuples. Xu et al. [138] developed a specification inference

approach for learning unique specifications using an integer programming approach.

The most closely related work to ours is by Kong et al. [67], [66], and Yoo and Belta [141]. These works proposed algorithms for simultaneously mining STL propositions and formulas based on the parametric STL grammar that they define. Recent work by Vazquez-Chanlatte et al. [133] proposes a maximum likelihood framework for specification inference with a maximum entropy estimate of the likelihood function that aligns with our approach of defining a likelihood function under certain conditions as described in Section 4.2.2.

Another approach towards learning non-Markov task specification infers the underlying automaton directly, rather than inferring a logical formula. Toro-Icarte et al. [129], and Xu et al. [139] adopt a learning approach based on reward machines as the hypothesis space for task specifications. By contrast, Araki et al. [5] model specifications inferred from noisy and ambiguous demonstrations as probabilistic automata.

2.2.3 Opportunity: Bayesian Inference of Complex Task Specification

We identified two key challenges in inferring specifications from demonstrations that are not addressed in prior research. The, first is the ambiguity of the demonstrations, where there may be multiple candidate LTL formulas that are all satisfied by the demonstrations observed. While longer and more numerous demonstrations might alleviate the problem, disambiguated data is not always guaranteed. We propose adopting a Bayesian concept learning [125, 123] approach towards specification inference, where the learner can encode the ambiguity over multiple specifications as a belief over candidate LTL formulas, thus not being restricted to infer a single LTL formula. The second key challenge is the recursive grammar used to define all possible LTL formulas. The relevant formulas require generating many LTL grammar branches, and most formulas generated are not relevant or vacuous. We propose leveraging probabilistic programming approaches combined with template-based hypothesis spaces defined using a subset of templates identified by Dwyer et al. [39]. This allows us to generate prior distributions that maintain support over relevant formulas without requiring a sample path with many branches.

Leveraging probabilistic programming languages is the key to our approach. The idea of a universal probabilistic programming language was formalized by Freer et al.

[46], and Goodman et al. [49]. These ideas have resulted in the development of Turing-complete probabilistic programming languages such as Church [49], webppl [50], and Gen [37]. Probabilistic programming languages have been instrumental in enabling a Bayesian approach to grammatical inference [38] by allowing the composition of modular inference algorithms over arbitrary distributions. Ellis et al. [42, 41] have demonstrated the success of probabilistic programming approaches to inferring graphics programs and language rules. Silver et al. [116] demonstrated the utility of such an approach towards learning robotic manipulation policies.

2.3 Robot Behavior Generation

In a general case, the robot learning operating in the environment can be modeled as a hybrid (continuous and discrete) stochastic dynamical system with partial observability of the environment state [60]. There are many commonly used abstractions for robot planning that relax some of these assumptions. The Markov decision process (MDP) [13] formalism stresses the probabilistic transitions in the system while assuming full observability of the state. The symbolic planning abstraction [45, 48] allowed for factored state representation, but with deterministic and sparse transitions. Finally, non-deterministic symbolic planning formulations allow for factored state representation while allowing for sparse non-determinism in the state transitions [28]. We begin by surveying how planning to fulfill goals and constraints expressed in temporal logic has been studied in prior work.

2.3.1 Decision-Making with Temporal Logic Constraints

The approaches to decision-making with temporal logics can be categorized into automata-based methods, optimization-based methods, and reinforcement learning or planning-based approaches.

LTL formulas can be translated into automatons that naturally induce a discrete controller. Thus an approach to planning would be to compose local continuous state controllers with discrete mode controllers induced by LTL as proposed by Kress-Gazit et al. [70]. Such controllers are verifiably correct if certain assumptions on the continuous system dynamics

hold. Camacho et al. [28] leverage this conversion to an automaton with symbolic planners whose state spaces are defined with planning domain description languages (PDDL).

A second approach is to translate LTL formulas into mixed-integer constraints and use them along with SAT solvers or optimization frameworks to generate plans that satisfy the specifications. Karaman and Frazzoli [61] provide a mixed-integer encoding for LTL formulas and use those techniques for vehicle routing problems. Raman et al. [94] propose a model predictive control scheme that encodes STL specifications as mixed-integer constraints. Sadigh and Kapoor [103] proposed an approach to encode nondeterministic constraints expressed in probabilistic STL to ensure that the specifications are satisfied under a chance-constrained framework.

Finally, temporal logic formulas can also be leveraged to define non-Markovian rewards in the reinforcement learning setting. Littman et al. [77] propose Geometric LTL (G-LTL), a variant of LTL with temporal semantics that are allowed to expire after a finite time length controlled by a geometric distribution. This is akin to the idea of discounting cumulative rewards to stabilize temporal difference learning. LTL formulas are converted to a specification MDP (Spec-MDP) with well-defined terminal states in this approach. A cross product of Spec-MDP with the original system MDP is used with a traditional reinforcement learning algorithm. The modified semantics nudge the system behavior towards satisfying the LTL specification for the time scales where the formulas hold. A second approach is to treat non-Markovian rewards as piecewise Markovian. Toro-Icarte et al. [127] propose an algorithm to systematically decompose LTL formulas into a curriculum of reward functions based on possible progressions of the formula and using curriculum learning approaches to compute an optimal policy. Toro-Icarte et al. [128, 127] and Camacho et al. [26] further proposed a unified framework called reward machines, along with a suite of reinforcement learning algorithms that exploit their structure to specify non-Markov properties. Finally, Oh et al. [88] have developed a system to translate natural language commands to temporal logic specifications with propositions grounded to an object-oriented MDP, thus allowing a robot to generate behaviors conforming to a language command.

Yet another area of prior work deals with the problem of expressing LTL specifications as rewards. While the G-LTL [77] and the LPOPL framework allow only for binary rewards

representing satisfaction or non-satisfaction of LTL constraints, Aksaray et al. [3] proposed using variants of STL with quantitative semantics as the reward function in an MDP setting. Bacchus et al. [9] defined temporally extended reward functions (TERF) over a set of past-tense LTL formulas, thus modeling rewarding behaviors as preferences of LTL formulas.

2.3.2 Reward Design for User Intent

Prior research into planning and learning for MDPs has indicated great promise at addressing challenging domains. The breakthroughs have addressed large dimensional state-spaces such as Atari games ([84]), continuous action spaces [83], sparse rewards ([40, 116]), or all these challenges together ([135]). However, these successes have been limited to domains with a clear reward function. The problem of value alignment with the user's intention is well-known [7, 82].

This has led to studies in reward design and models for handling uncertainty in the actual reward function to better align with the user's objectives. This problem has primarily been studied in the context of Markov reward functions. Singh et al. [118] first defined the problem of optimal reward design for a distribution of target environments. Ratner et al. [99] and Hadfield-Menell et al. [53] defined inverse reward design as the problem of inferring the true desiderata of the task from proxy reward functions provided by the users for a set of task environments. Regan and Boutillier [101] proposed algorithms for computation of robust policies that satisfy the minimax regret criterion rather than the expected reward.

2.3.3 Opportunity: Accounting for Uncertainty in Task Specifications

Prior research into handling epistemic uncertainty in task specifications has been primarily restricted to the Markov setting. In this dissertation, we address planning with a belief over task specifications expressed as LTL formulas. We introduce a novel formulation called *planning with uncertain specifications (PUnS)*. Towards this, our key contributions involve identifying evaluation criteria that capture the semantics of satisfying a belief over logical formulas. Further, we demonstrate that every PUnS problem can be compiled into an equivalent reward machine [128], making PUnS compatible with any planning or

reinforcement learning algorithm that accepts an MDP problem definition.

2.4 Interactive Approaches to Robot Learning

The research opportunities described thus far treat the problem of specification inference and planning independently as the learner is expected to learn passively from labeled task execution data. However, approaching these problems in a unified setting permits the learner to actively guide its learning by deciding which data points to learn from and deciding what modality to learn from. Such a learner is closer to a human apprentice who learns not just by observing but also by asking questions and actively seeking information.

Yet another aspect that is of interest to us is that the robot can be trained by the task expert through multiple modalities apart from demonstrations, provided that the robot's representation of the task specifications has the appropriate abstraction. When multiple learning modalities are available, deciding which modality to learn from is yet another choice available to the learner. In this section, we survey prior works that aim to develop an active learning agent by selecting what data points are most suitable to guide its learning and asserting a choice over what learning modality to use.

2.4.1 Active Learning for Robotics

In an active learning setting, the learning agent can interactively query an information source to annotate new data points. An active learning algorithm relied on acquisition functions, i.e., an estimator for the utility gained through annotating a given data point. Settles [106] surveyed a wide range of active learning paradigms prevalent in machine learning research.

Recently, there has been considerable interest in developing active learning algorithms for sequential decision-making tasks relevant to robotics. One expected benefit of an active approach is that the learner can guide the teacher's feedback to impact the learner's behavior optimally. Cakmak et al. [25, 24] developed a taxonomy of queries that allow a learner to refine its understanding of the task specifications. Sadigh et al. [102] proposed an active learning framework for sequential decision-making problems that relies upon pairwise preference between candidate trajectories selected according to a maximum volume removal

heuristic; Biyik et al. [18] then extended this to generate queries using maximum information gain criterion. Biyik and Sadigh [19] proposed a batch-active framework for preference-based learning wherein multiple queries are generated simultaneously instead of one at a time. Cui and Niekum [36] proposed an active learning model based on information gain that operates on individual state-action pairs, allowing segments of the trajectory to be labeled "esirable" and "undesirable." However, present research into active learning for robotics has primarily focused on formulations representing the underlying task as a Markov decision process (MDP), with the state space known a priori.

2.4.2 Intuitive Training Modalities for Robots

A second thrust for prior research has been into identifying intuitive training modalities for robots. The natural first modality is learning from demonstrations as surveyed in Section 2.2.1. However in addition modalities such as natural language instructions, [51, 88]; corrections [11, 11]; preference elicitation [19, 18, 17, 102]; or even from a simple stop command [53]. Particularly notable is the work by Jeon et al. [57] on modeling teacher-learner interaction as that of the teacher providing a reward-rational implicit choice among alternatives as a teaching input. This motivates a unifying framework where all teacher inputs are treated as means to alter the learner's belief over task parameters. In such a setting, a learner that models the teacher's inputs can estimate the value of each of the learning modalities available to it and select the most fruitful one–also described as meta-choice.

Parallel to this work is the problem of modeling rational teaching behavior. Humans act differently when simply performing a task in contrast to teaching a task to a novice [73, 96]. Brown et al. [23] developed algorithms for pedagogical sampling, and in turn, adopted it to refine the likelihood model for a learner, resulting in an inverse reinforcement learning algorithm that learned faster than a naive learner. Milli et al. [80] also demonstrated the utility of assuming teacher's pedagogical selectivity. However, Milli and Dragan [81] performed studies that indicated that while peak performance for models that assumed pedagogical reasoning is better, this is brittle when used in conjunction with teachers with an unknown degree of pedagogical selectivity. A common thread in these works is the Bayesian pedagogical reasoning framework formalized by Shafto et al. [107, 108, 109]. The

benefit of enabling pedagogical reasoning for robots and the appropriate cognitive models for pedagogical reasoning is an active research area.

2.4.3 Opportunity: Deciding How to Learn–A Multi-modal approach

Prior work in online and multi-modal learning for robots has primarily focused on the Markov setting. In contrast, the complex, multi-step tasks that are a focus for this dissertation are not amenable to a naive Markov representation. We propose developing the ideas of active learning, pedagogical human input, and meta-choice in a scenario with non-Markov task specifications. The contribution of this dissertation towards this includes developing the complete Bayesian framework for inferring task specifications from different teacher inputs, including demonstrations and assessments of the learner's task execution, and generating the robot policy while accounting for the uncertainty in task specification; in addition to this, we develop active-learning models that first addresses the problem of meta-choice–i.e., what input modality to learn from next–and identifying and performing an informative task execution to best refine the learner's own belief over the true task specifications.

Chapter 3

Problem Statement

We formalize the problem of training a robot learner to competently perform an intended task through the lens of Bayesian concept learning [123, 125]. We study the setting where the teacher intends to teach the learner a specific task that they intend the learner to perform. Following a Bayesian approach, the learner always maintains a belief over its intended task specification, its interactions with the teacher serve as evidence to update the learner's belief as per the Bayes' rule. One of the contributions of this thesis is enabling robot training for non-Markov task specifications–where the execution history is required to determine the satisfaction of the specifications. We rely on linear temporal logic (LTL) [93] as our chosen specification language due to its well-studied expressivity [78, 132], and its widespread use in algorithms for synthesis [70, 94, 95, 71]. Further, we intend to deploy the robot learners in non-deterministic domains; therefore, we rely on the Markov decision process formalism to describe the robot environment.

In this chapter we begin with a short introduction to LTL, its *safe*, and *co-safe* fragments, along with the definition of the Markov decision process (MDP) formalism in Section 3.1. We then define the problems of Bayesian specification inference, planning with uncertain specifications, and iterative refinement of belief over task specifications in Section 3.2.

3.1 Preliminaries

3.1.1 Linear Temporal Logic

Linear temporal logic (LTL), introduced by Pnueli [93], provides an expressive grammar for describing temporal behaviors. An LTL formula is composed of atomic propositions (discrete time sequences of the truth values of Boolean propositions) 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:

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi_1 \mid \varphi_1 \mathbf{U} \varphi_2. \tag{3.1}$$

Here, p is an atomic proposition, and φ_1 and φ_2 represent 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 $t_1 \leq t \leq t_2$. We also use the additional propositional 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 $t \geq t_1$.

The "safe" and "co-safe" subsets of LTL formulas have been identified in prior research ([72], [132], [78]). A "co-safe" formula can always be verified by a trace of a finite length, whereas a finite trace can always falsify a "safe" formula. Any formula produced by the following grammar is considered "co-safe":

$$\varphi_{co-safe} ::= \top \mid p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \varphi_1 \mathbf{U} \varphi_2. \tag{3.2}$$

Similarly, any formula produced by the following grammar is considered "safe":

$$\varphi_{safe} ::= \bot \mid p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi \mid \varphi_1 \mathbf{R} \varphi_2. \tag{3.3}$$

A formula expressed as $\varphi = \varphi_{safe} \wedge \varphi_{co-safe}$ belongs to the Obligation class of formulas presented in Manna and Pnueli's [78] temporal hierarchy.

Finally, a progression $\text{Prog}(\varphi, \alpha_t)$ over an LTL formula with respect to a truth assignment α_t at time t is defined such that $\forall[\alpha]$: $[\alpha], t \models \varphi$ *iff* $[\alpha, t+1] \models \text{Prog}(\varphi, \alpha_t)$. Thus, a progression of an LTL formula with respect to a truth assignment is a formula that must hold at the next time step in order for the original formula to hold at the current time step. Bacchus and Kabanza [10] defined a list of progression rules for the temporal operators in Equations 3.1, 3.2, and 3.3.

3.1.2 Markov Decision Processes

A Markov decision process (MDP) is a planning problem defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where $\mathcal{S}$ represents the set of all possible states; $\mathcal{A}$ is the set of actions available to the learner; $T := P(s' \mid s, a)$ is a probability distribution over the next state $s' \in \mathcal{S}$ given current state $s \in S$, and the action $a \in A$ executed at the current time step; and $R : \mathcal{S} \to \mathbb{R}$ is the reward function that returns a scalar value given the current state.

Watkins and Dayan proposed Q-learning [136], an off-policy, model-free algorithm to compute optimal policies in discrete MDPs. The Q-value function $Q_\pi(s, a)$ is the expected discounted value under a policy $\pi(a \mid s)$. In a model-free setting, the transition function is not known to the learner, and the Q-value is updated by the learner acting within the environment and observing the resulting reward. If the Q-value is updated while not following the current estimate of the optimal policy, it is considered "off-policy" learning. Given an initial estimate of the Q-value $Q(s, a)$, the agent performs an action a from state s to reach s' while collecting a reward r and a discounting factor $\gamma \in [0, 1)$. The Q-value function is then updated as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a')). \tag{3.4}$$

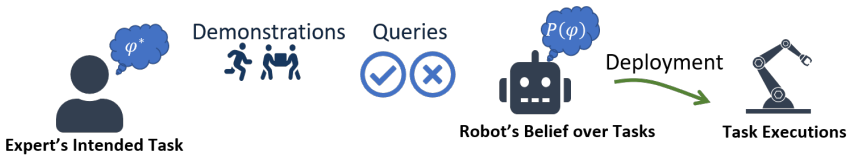


Figure 3-1: The scope of this thesis involves scenarios where a teacher attempts to teach a learner how to act in an environment as per an intended task specification. The teacher only interacts with the learner through two natural modalities: demonstrations of acceptable task executions, and assessments of executions performed by the learner.

3.2 Robot Training through Specification Inference

This thesis deals with specification inference and behavior generation in the face of epistemic uncertainty in task specifications. Figure 3-1 depicts our problem setting that involves a teacher and a learner. The teacher intends to teach a desired task to the learner through two natural modalities–demonstrations of task executions that are considered acceptable as per the intended task specifications and acceptability assessments of task executions performed by the learner.

Following a Bayesian approach, the learner maintains a belief distribution over the teacher's intended task specification, and all inputs provided by the teacher serve to update the learner's belief. The teacher's intended task specification is encoded by a linear temporal logic (LTL) [93] formula, φ^*. We restrict the hypothesis space of the task specifications to a template-based distribution described in Section 4.1.

The task is performed within the scope of an environment $\mathcal{M}_\mathcal{X} = \langle \mathcal{X}, \mathcal{A}, T_\mathcal{X} \rangle$. Further, we assume that a set of n_{prop} Boolean propositions $\boldsymbol{\alpha} \in \{0,1\}^{n_{prop}}$ is sufficient to evaluate the truth value of the task specification φ^*. A task execution performed by either the teacher or the learner is represented by a sequence of environment states $[x]$. This sequence is mapped injectively to a sequence of truth values of the propositions, $\boldsymbol{\alpha}$, through a pre-defined labeling function $\boldsymbol{\alpha} = f(x)$. We further assume that the environment state is fully observable by the learner; thus, $\boldsymbol{\alpha}$ at any given time can be readily determined by the learner.

This thesis tackles three sub-problems of teaching a robot to perform tasks while reasoning about and refining the epistemic uncertainty in task specifications. First, we consider the problem of inferring a belief over task specifications purely from observation

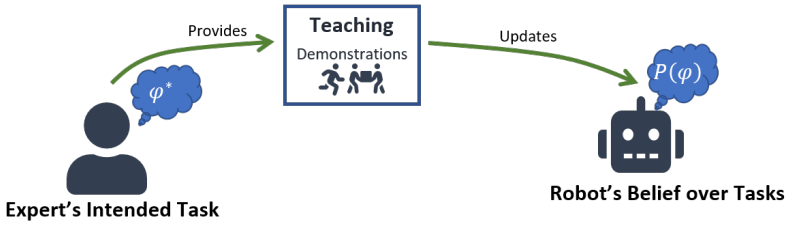


Figure 3-2: Bayesian specification inference problem setting where the learner infers the task specifications passively from observed executions data.

of a teacher's task executions. Next, we consider the problem of planning with uncertain specifications, where we define the semantics of maximally satisfying a distribution over logical formula and develop algorithms to compute the learner's policy. Finally, we develop a unified iterative approach combining specification inference and planning to rapidly infer and refine beliefs over task specifications using both teacher's demonstrations and assessments of the learner's task executions as learning modalities.

3.2.1 Bayesian Specification Inference

This sub-problem deals with inferring task specifications passively from labeled task executions performed by the teacher. Each task execution is represented as a sequence of environment states $[x]$ that is injectively mapped to a sequence of truth values of the propositions $[\boldsymbol{\alpha}]$. For this thesis, we restrict the specifications that the learner can infer to a fragment of LTL formulas that represent conjunctive compositions of three temporal behaviors among those identified by Dwyer et al. [39]. These behaviors include the satisfaction of a constraint, φ_{global}, the eventual completion of a subtask, $\varphi_{eventual}$, and mutual ordering among the subtasks, φ_{order}, defined as follows:

$$\varphi = \varphi_{global} \wedge \varphi_{eventual} \wedge \varphi_{order}. \tag{3.5}$$

We further consider two forms of inference problems: the batch learning setting, where the learner begins with no previous task-specific information, the and iterative learning setting, where the learner receives the teacher's inputs iteratively.

1. **Batch Learning:** In the batch learning setting, the learner begins with a prior belief

over task specification that has non-zero support over all possible LTL formulas that conform to a the template defined in Eq 3.5. The teacher provides a set of task executions along with the boolean acceptability labeled, $\mathcal{D} = \{\langle[\boldsymbol{\alpha}]_1, \mathcal{L}_1\rangle, \langle[\boldsymbol{\alpha}]_2, \mathcal{L}_2\rangle, \ldots\}$. Here $\mathcal{L}_i \in 0, 1$ is a binary acceptability label for the trace $[\boldsymbol{\alpha}]_i$. The learner updates its belief by computing the Bayesian posterior $P(\varphi \mid \mathcal{D})$.

2. **Iterative Learning:** In the iterative learning setting, the learner updates its belief iteratively one trajectory at a time, i.e. the dataset, $\mathcal{D}$, consists of a single labeled trajectory. The belief at end of iteration *i* is represented by $P(\varphi)_i$. The learner observes the sequence of propositions and an acceptability label, $\mathcal{D} = \langle[\boldsymbol{\alpha}]_i, \mathcal{L}_i\rangle$. The learner must then compute the updated posterior belief $P(\varphi \mid \mathcal{D}; P(\varphi)_i)$. In the following iteration, the belief $P(\varphi \mid \mathcal{D}; P(\varphi)_i)$ becomes the prior belief $P(\varphi)_{i+1}$.

In both these settings, the belief distribution is approximated by a discrete probability distribution over a finite, but a very large set of candidate LTL formulas with the probability mass function $P : \{\varphi\} \to \mathbb{R}$. The success of the learning is assessed using two quantitative metrics. The certainty of the learner's belief is evaluated using the entropy [114] of the belief distribution. For the learner's belief $P(\varphi)_i$, this is computed as follows:

$$H(\varphi)_i = \sum_{\varphi \in \{\varphi\}} -P(\varphi)_i \log(P(\varphi)_i). \tag{3.6}$$

We assess the accuracy of the inferred specification by computing the similarity to the LTL formula representing the ground truth task specification. Note that we restrict the scope of candidate specifications to conjunctively composed LTL formulas. Consider two formulas, φ_1 and φ_2 that are conjunctive compositions of the clauses contained within sets $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ respectively. We define the similarity between these formulas using an intersection-over-union as follows:

$$L(\varphi_1, \varphi_2) = \frac{|\, \boldsymbol{C}_1 \cap \boldsymbol{C}_2 \,|}{|\, \boldsymbol{C}_1 \cup \boldsymbol{C_2} \,|}. \tag{3.7}$$

Therefore, given a belief distribution $P(\varphi)_i$ and the ground-truth specification encoded by the formula φ^*, we compute the similarity to the ground-truth as follows:

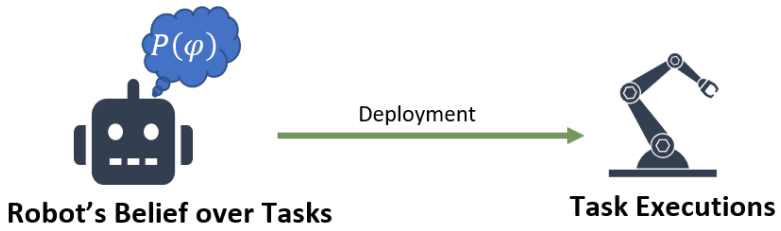


Figure 3-3: Planning with uncertain specifications (PUnS) problem setting

$$L_{\varphi^*}(P(\varphi)_i) = \mathbb{E}_{P(\varphi)_i}[\![L(\varphi, \varphi^*]\!] \tag{3.8}$$

Chapter 4 focuses on the problem of Bayesian specification inference from passive observations of the teacher.

3.2.2 Planning with Uncertain Specifications

This sub-problem deals with learner behavior generation in the face of epistemic uncertainty of its task specifications as depicted in Figure 3-3. The robot is deployed to act in the environment domain $\mathcal{M}_{\mathcal{X}}$, and it has a belief over its intended task specification $P(\varphi)$. The robot must compute a stochastic policy, $\pi(a \mid x; P(\varphi))$, such that the resulting task executions are diverse–i.e. demonstrate many valid paths towards completing the task–and satisfy the belief over logical formulas. Here $a \in \mathcal{A}$, and $x \in \mathcal{X}$. Our contributions towards this novel problem formulation and algorithms for compiling it into an equivalent Markov decision process (MDP) are described in Chapter 5.

3.2.3 Online Multi-Modal Robot Training

The Bayesian specification inference formulation is agnostic to whether the task executions are generated by the teacher acting independently in the task environment, the teacher commanding the learner's action within the environment, or the learner completing its own task executions. Leveraging that along with the algorithms developed for PUnS problem formulation, we tackle the problem of making the learner an active decision-maker in guiding its task learning through an interactive training model. Figure 3-4 depicts the Bayesian

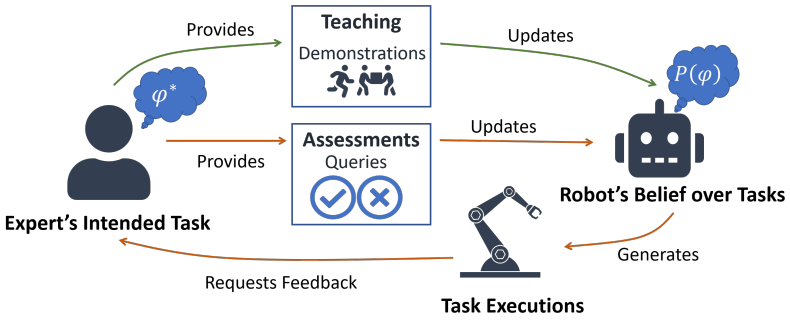


Figure 3-4: Online multi-modal Bayesian interactive training framework.

interactive specification inference framework that allows the robot to learn both from labeled task executions passively, and it also enables the robot to guide its learning by generating informative task executions. Formally, this involves solving the following problems:

1. **Determining optimal informative query execution:** Given the learner's current belief $P(\varphi)_i$ over its task specification, the robot must compute a policy $\pi_{query}(a \mid x; P(\varphi))$, that when executed results in a task execution trace $[\alpha]$ for which an acceptability label $\mathcal{L} \in 0, 1$ provided by the teacher would be most informative.

2. **Determining learning modality for the next iteration:** Given the learner's current belief, it must decide what modality to learn from next. Either the learner can choose between asking the teacher for an additional demonstration that satisfies the teacher's intended task specification, or the learner can perform a query task execution and elicit an acceptability label from the teacher. Thus the learner must compute a meta policy $\pi(P(\varphi)_i)$ that maps to a binary variable that encodes the next learning modality.

As in the Bayesian specification inference setting, the learner's success is determined by comparing the inferred task specification to the ground-truth task specification. The learner's certainty of its learned specification is measured using the entropy of the posterior distribution. Chapter 6 focuses on our proposed approaches to the problems described here.

Chapter 4

Bayesian Specification Inference

This chapter introduces our Bayesian approach to inferring task specifications from demonstrations defined formally in Section 3.2.1. Temporal logics have been used as a specification language to describe desirable system behaviors. The compositional nature of temporal logics allows for expressing complex task specifications through logical and temporal comopsition of simpler properties, thus allowing for a very expressive language.

However, the flexibility and expressivity of LTL in specifying behaviors also represent a key challenge concerning specification inference due to a large hypothesis space. Inferring the intended task specification requires sampling from the space of candidat LTL formulas to identify ones that best explain the observed behavior. However, relevant specifications only form a small fraction of naïvely sampled valid LTL formulas. We address this by defining a template-based prior over a small but relevant fragment of LTL. A second key challenge is the ambiguity inherent in learning from demonstration. Given a set of labeled task executions–particularly if the learner observes only acceptable task executions–multiple candidate LTL formulas are satisfied by all the demonstrated executions. We address this ambiguity by framing the inference problem as Bayesian concept learning [123, 125] where the learner maintains a belief over the true task specification and updates it conditioned on the observed task executions.

We demonstrate the efficacy of our proposed model in the inductive learning setting in both a synthetic domain that allows for easily modifying the ground truth formulas and in inferring the specifications for the task of setting a dinner table, where the inferred

specifications align with the ground truth specification with greater than 90% similarity. We also demonstrate the utility of our model in automatically evaluating mission objective completion for simulated multi-aircraft air combat exercises. The model's predictions were well-aligned with the expert's evaluation of the mission objectives. The formula's logical structure allowed for an interpretable representation of the model's decision-making. This model is a part of the Intelligent mission analysis and review system developed in collaboration with the Lockheed Martin Corporation [34, 35].

4.1 Defining the Expressivity of Candidate Specifications

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. [39]–namely, global satisfaction of constraints, the eventual completion of a subtask, and temporal ordering between subtasks. With φ_{global}, $\varphi_{eventual}$, and φ_{order} representing LTL formulas for these behaviors, the task specification is written as follows:

$$\varphi = \varphi_{global} \wedge \varphi_{eventual} \wedge \varphi_{order}. \tag{4.1}$$

Recall from Section 3.2.1 that $\boldsymbol{\alpha}$ represents the set of Boolean propositions necessary to evaluate the successful completion of the candidate task. We adopt a template-based approach towards expressing the scope of the candidate task specifications. The three templates that we consider for the scope of this thesis are as follows:

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

$$\varphi_{global} = \left(\bigwedge_{\tau \in \boldsymbol{\tau}} (\mathbf{G}(\tau)) \right). \tag{4.2}$$

   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.

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

$$\varphi_{eventual} = \left(\bigwedge_{w \in \mathbf{W_1}} (p_w \rightarrow \mathbf{F}\omega_w) \right). \tag{4.3}$$

3. **Temporal ordering:** Every set of feasible ordering constraints over a set of subtasks maps to a directed acyclic graph (DAG) over nodes representing these subtasks. Each edge in the DAG corresponds to a binary precedence constraint. Let $\boldsymbol{W_2}$ be the set of binary temporal orders defined by $\boldsymbol{W_2} = \{(w_1, w_2) : w_1 \in \boldsymbol{V}, w_2 \in \text{Descendants}(w_1)\}$, where $\boldsymbol{V}$ is the set of all nodes within the task graph. Thus, the ordering constraints include an enumeration of the edges in the task graph and all descendants of a given node. For subtasks w_1 and w_2, the ordering constraint is written as follows:

$$\varphi_{order} = \left(\bigwedge_{(w_1, w_2) \in \mathbf{W_2}} (p_{w_1} \rightarrow (\neg\omega_{w_2} \mathbf{U} \omega_{w_1})) \right). \tag{4.4}$$

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

The set of propositions α is constructed from a union of three sets: $\boldsymbol{T}$, the set of candidate constraints that must always be satisfied; $\{p_w : \forall\, w \in \Omega\}$, the set of conditional propositions that control the necessity of completion subtasks $w \in \Omega$; and $\{\omega_w : \forall\, w \in \Omega\}$, the set of propositions representing the completion of the subtasks in the set Ω. There may be a non-empty intersection between the subtask propositions $\{\omega_w\}$ and the set of constraints $\boldsymbol{T}$, but these must be disjoint from the conditional propositions $\{p_w\}$.

Under these assumptions, and more generally in a template-based setting, the problem of inferring the correct formula is equivalent to identifying the correct subsets, $\boldsymbol{\tau}$, $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$.

4.2 Specification Learning as Bayesian Inference

Bayes' theorem is the cornerstone of our approach to specification inference. Let φ represent the set of all LTL formulas that conform to the template represented by eq 3.5. Following a Bayesian approach, the learner holds an initial belief distribution with the probability mass function $P : \varphi \to [0,1]$. Given some observed data $\mathcal{D}$, and the prior belief, Bayes' theorem is applied as follows:

$$P(\varphi \mid \mathcal{D}) = \frac{P(\varphi)P(\mathcal{D} \mid \varphi}{\sum_{\varphi \in \varphi} P(\varphi)P(\mathcal{D} \mid \varphi)}. \tag{4.5}$$

Here $P(\mathcal{D} \mid \varphi)$ represents the likelihood model of observing the observed labeled trajectories given the ground truth specification. It is usually computationally intractable to compute the denominator of eq 4.5 exactly; therefore, we employ approximate sampling-based inference algorithms to estimate the posterior distribution.

Our key technical contributions was developing an inference model based on universal probabilistic programming languages that allow for a modular inference approach over complex. The second key contribution was defining a domain-independent approximation of the likelihood function that enables our model to be widely applicable.

4.2.1 Prior Definitions

While sampling candidate formulas as per the template depicted in Equation 4.1, we treat the sub-formulas in Equations 4.2, 4.3, and 4.4 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`(Ω, 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.

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. It 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 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 $1-p_{part}$).

Algorithm 1 SampleSetsOfLinearChains

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

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 $\boldsymbol{P}$, 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, $\mathscr{F}$. The i^{th} tree has two attributes: a root node, $\mathscr{F}[i]$.root, and a 'descendant forest,' $\mathscr{F}[i]$.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, $\mathscr{F}$.length, is the

number of trees included in that forest. The size of a tree, $\mathscr{F}[i]$.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 $\boldsymbol{P}$, 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 $\{1,2,\ldots,\mathscr{F}.\text{length}+1\}$ as the possible outcomes. The probability of each outcome is proportional to the size of the trees in the forest, while the probability of $\mathscr{F}.\text{length}+1$ being the outcome is proportional to N_{new}, a user-defined parameter. This sampling process is similar in spirit to the Chinese restaurant process [4]. If the outcome of the draw is $\mathscr{F}.\text{length}+1$, then a new tree with root node a is created in line 10; otherwise, `InsertIntoForest` is called recursively to add a to the forest $\mathscr{F}[i]$.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
```

4.2.2 Likelihood Function

The likelihood distribution, $P(\mathcal{D} \mid \varphi)$, 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:

$$P(\mathcal{D} \mid \varphi) = \prod_{i \in \{1,2,\ldots,n_{demo}\}} P(\varphi) P(\langle [\boldsymbol{\alpha}]_i, \mathcal{L}_i \rangle \mid \varphi) \tag{4.6}$$

The probability of observing a given trajectory demonstration depends on the underlying dynamics of the domain, the characteristics, and the policy 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. A second choice for a likelihood function, inspired by Shepard et al. [115], is defined as the SIM model by [123]; we call this the 'complexity-independent' (CI) likelihood function. We define these likelihood functions as follows:

1. **Complexity-based (CB):** The CB likelihood function is founded upon the classical interpretation of probabilities championed by Laplace [74], and on the size principle described by Tenenbaum [123]. The classical interpretation of probability involves computing probabilities in terms of an enumerated set of equally likely outcomes. In contrast, the size principle states that the likelihood odds must favor the more restrictive of the two hypotheses for given observed data that conforms to two hypotheses. Here we demonstrate that approximating the likelihood of generating the observed set of demonstrations using the classical interpretation of probability naturally satisfies the size principle.

   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 $\boldsymbol{W}_{2_1}\mathbf{1}$) and φ_2 (defined by subsets $\boldsymbol{\tau_2}, \boldsymbol{W}_{1_2}$, and $\boldsymbol{W}_{2_2}$) the likelihood odds ratio for an acceptable demonstration is defined as follows:

$$\frac{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 1\rangle \mid \varphi_1)}{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 1\rangle \mid \varphi_2)} = \begin{cases} \frac{2^{N_{conj_1}}}{2^{N_{conj_2}}} & , [\boldsymbol{\alpha}] \models \varphi_2 \\ \frac{2^{N_{conj_1}}}{\varepsilon} & , [\boldsymbol{\alpha}] \not\models \varphi_2. \end{cases} \tag{4.7}$$

   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, suppose the candidate formula is not the true specification. In that case, a

larger set of demonstrations is more likely to include non-satisfying examples, thereby substantially decreasing the posterior probability of the candidate formula.

Similarly following the classical probability argument as before for demonstrations deemed unacceptable ($\mathcal{L} = 0$), with N_{conj} conjunctive clauses in a candidate formula, there are $2^{N_{conj}} - 1$ truth evaluations of the individual clauses, that would result in the given demonstration not satisfying the candidate formula. Thus the likelihood odds for unacceptable demonstrations are defined as follows:

$$\frac{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 0 \rangle \mid \varphi_1)}{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 0 \rangle \mid \varphi_2)} = \begin{cases} \frac{2^{N_{conj_1}}(2^{N_{conj_2}}-1)}{2^{N_{conj_2}}(2^{N_{conj_1}}-1)} & , [\boldsymbol{\alpha}] \nvDash \varphi_2 \\ \frac{2^{N_{conj_1}}}{(2^{N_{conj_1}}-1)\varepsilon} & , [\boldsymbol{\alpha}] \models \varphi_2. \end{cases} \tag{4.8}$$

Chanlatte-Vazquez et al. [134] also proposed a maximum-likelihood model for inferring specifications expressed in formal logics. Their approach of approach results in a likelihood function that is proportional to the ratio of the observed satisfaction rate of a specification to the satisfaction rate under an assumed random behavior. While the initial approaches were published simultaneously, the proposed likelihood functions are aligned, assuming a highly competent teacher.

2. **Complexity-independent (CI)**: This likelihood model does not follow the size principle; therefore, it equally favors any two hypotheses that are supported by the observed data. In case of acceptable demonstrations, the CI likelihood odds ratio is defined as follows:

$$\frac{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 1 \rangle \mid \varphi_1)}{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 1 \rangle \mid \varphi_2)} = \begin{cases} 1 - \varepsilon, & \text{if } [\boldsymbol{\alpha}] \models \varphi \\ \varepsilon, & \text{Otherwise.} \end{cases} \tag{4.9}$$

CI likelihood function for unacceptable demonstrations is defined as follows:

$$\frac{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 0 \rangle \mid \varphi_1)}{P(\langle [\boldsymbol{\alpha}], \mathcal{L} = 0 \rangle \mid \varphi_2)} = \begin{cases} 1 - \varepsilon, & \text{if } [\boldsymbol{\alpha}] \nvDash \varphi \\ \varepsilon, & \text{Otherwise.} \end{cases} \tag{4.10}$$

4.2.3 Inference Algorithms

We implemented our probabilistic model in webppl ([50]), a Turing-complete probabilistic programming language. 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.

Table 4.1: 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}

Batch Inference: In the batch inference mode, the learner is assumed to have no task specific knowledge at the beginning of the learning phase. The priors are defined as per one of the probabilistic programs defined in Section 4.2.1. The hyperparameters, including those defined in Table 4.1 and ε, were set as follows: $p_E, p_G = 0.8$; $p_{part} = 0.3$; $N_{new} = 5$; $\varepsilon = 4 \times log(2) \times (|\boldsymbol{T} + |\Omega| + 0.5|\Omega|(|\Omega| - 1))$. These values were held constant for all evaluation scenarios. The value of ε was chosen so that evidence of a single non-satisfying demonstration would negate the contribution of four satisfying demonstrations to the posterior probability.

Iterative Inference: In the iterative inference mode, the posterior distribution of the previous iteration is set as the prior for the subsequent iteration. Thus the prior in interactive mode will always be a discrete categorical distribution over a finite set of LTL formulas. Here the Bayesian inference can either be computed exactly.

4.3 Experiments

In this chapter, we focus only on the batch inference model; Chapter 6 focuses on the iterative mode of Bayesian specification inference operating in conjunction with the PUnS formulation presented in Chapter 5. We evaluated the performance of the proposed model across three different domains. First, we developed a synthetic domain with a low dimensional state-space where we can easily vary the ground-truth task specifications and readily generate satisfying task demonstrations. We utilized this domain to evaluate the performance of the model on ground-truth formulas with different temporal dependencies. We also demonstrated that the complexity-based (CB) likelihood function is more suitable to our model than the complexity-independent (CI) concept learning models initially proposed by Shepard et al. [115].

We also applied our model to a real-world task of setting a dinner table–a task often incorporated into studies of learning from demonstration [126]. This task has a large state-space incorporating the poses of the objects included in the domain. This domain demonstrates the benefits of using propositions to represent task specifications. The complexity of the problem depends upon the number of Boolean propositions and not the dimensionality of the raw state-space. Note that both the synthetic and table-setting domains represent scenarios where the ground-truth specification is already known. Note that we only consider these domains in the context of inductive learning, i.e., learning only from acceptable task demonstrations.

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. 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).

4.3.1 Synthetic Domain

An agent navigates within a two-dimensional space in our synthetic domain 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:

$$\tau_i = \begin{cases} \text{true}, & \|\boldsymbol{x} - \boldsymbol{x}_{T_i}\| \geq \varepsilon_{threat} \\ \text{false}, & \text{otherwise.} \end{cases} \tag{4.11}$$

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:

$$\omega_i = \begin{cases} \text{true}, & \|\boldsymbol{x} - \boldsymbol{x}_{P_i}\| \leq \varepsilon_{POI} \\ \text{false}, & \text{otherwise.} \end{cases} \tag{4.12}$$

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

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

$$p_i = \begin{cases} \text{false}, & \exists \; j \text{ such that } \left\|\boldsymbol{x}_{P_i} - \boldsymbol{x}_{T_j}\right\| \leq \varepsilon_{threat} \\ \text{true}, & \text{otherwise.} \end{cases} \tag{4.13}$$

p_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 we sampled their positions 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 4-1.

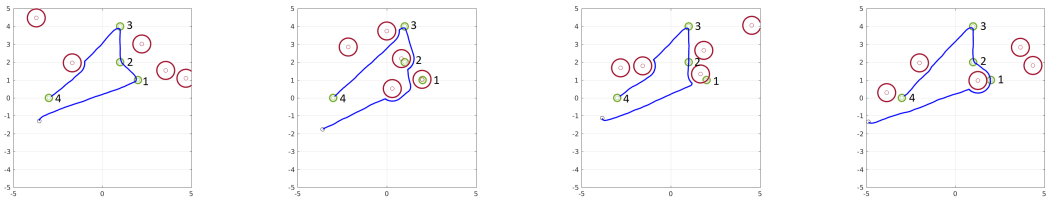
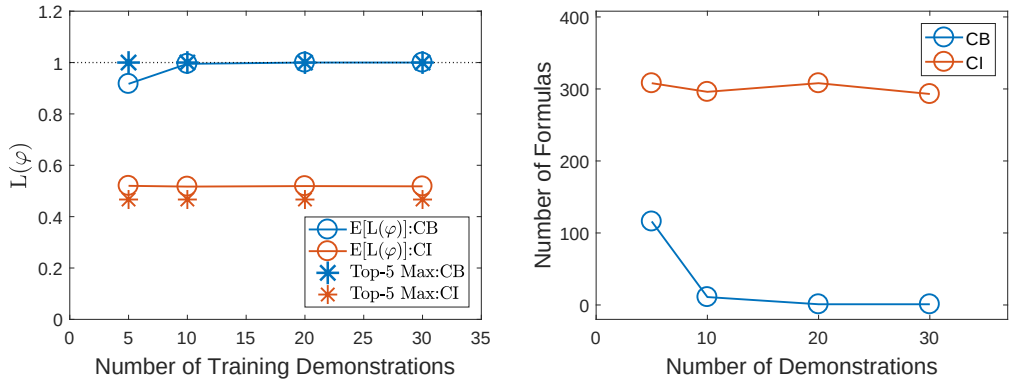


Figure 4-1: Example trajectories from Scenario 1 within the synthetic domain. Green circles denote the POIs; red circles denote the avoidance zones of threats.

The posterior distribution was computed using prior 1 (defined in Table 4.1), with both CB (Equation 4.7) and CI (Equation 4.9) likelihood functions. The expected and maximum values among the top 5 a posteriori formula candidates of $L(\varphi)$ are depicted in Figure 4-2. 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 a 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 4-2b depicts the number of unique formulas in the posterior distributions. The CB likelihood function resulted in posteriors being peakier, with fewer unique formulas as training set size increased; this effect was not observed with the CI function.

We also computed the posterior distributions 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 4-3a. Prior 3 aligned better with the ground-truth specification with fewer training examples. Prior 2 recovered the exact specification with a larger training



(a) Scenario 1

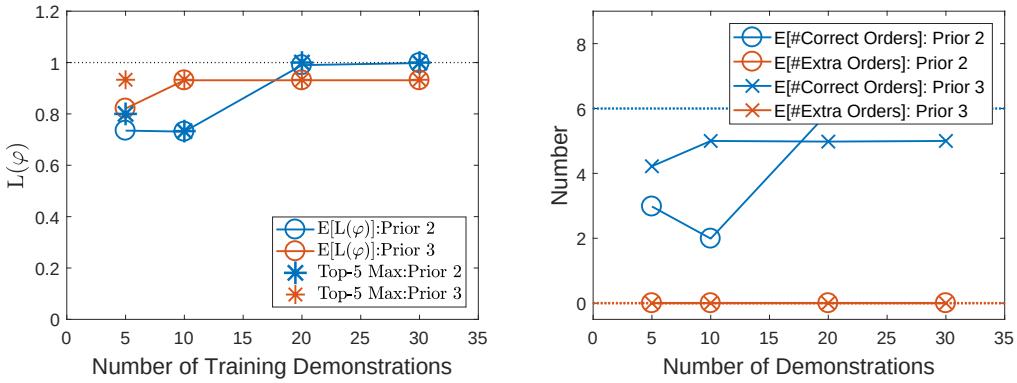
(b) Scenario 1

Figure 4-2: Comparison of CI and CB likelihood functions for Scenario 1 within the synthetic domain. Figure 4-2a depicts the results from Scenario 1, with the dotted line representing the maximum possible value of $L(\varphi)$. Figure 4-2b shows the number of unique formulas in the posterior distribution

set, while prior 3 failed to do so. Figure 4-3b 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 demonstrations compared 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, we sampled the threat positions uniformly for each demonstration. All the POIs, if accessible, had to be visited. We imposed a partial ordering constraint 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 4-4.

For Scenario 2, we computed the posterior distribution 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 4-5a. 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 4-5b 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, but both priors converged to the correct specification given



(a) Scenario 1 (b) Scenario 1

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

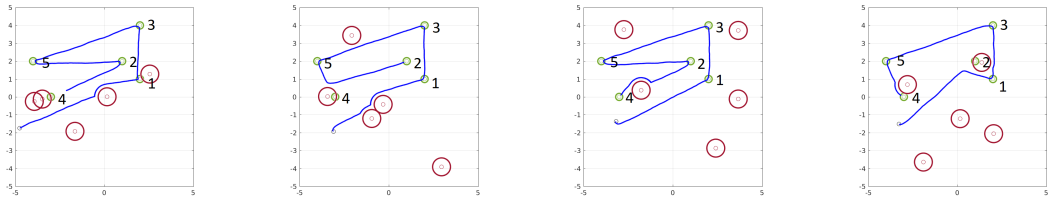
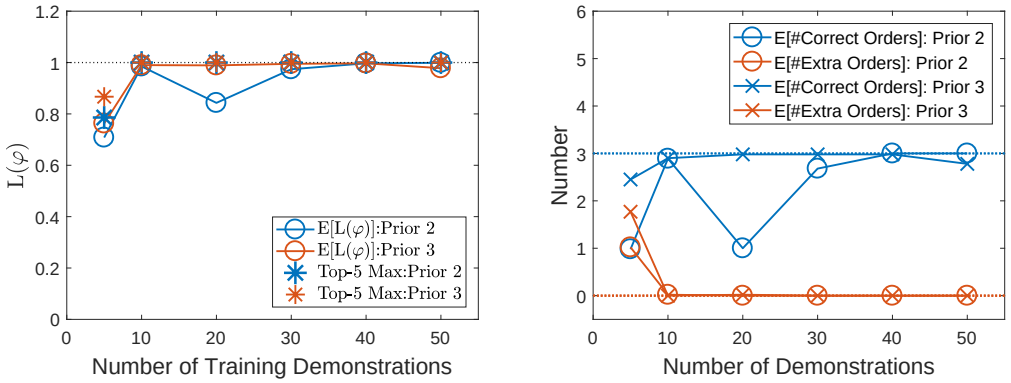


Figure 4-4: Example trajectories from Scenario 2. Green circles denote the POIs; red circles denote the avoidance zones of threats.

enough training examples.

Scenario 3: Scenario 3 included five threats, and five POIs labeled $\{1,2,3,4,5\}$, respectively. We uniformly sampled the threat positions 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 4-6 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 4-7a. 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 4-7b. Once again, the tendency



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

Figure 4-5: Specification inference results for Scenario 2 within the synthetic domain. Figure 4-5a indicates the $L(\varphi)$ values for Scenario 2, and Figure 4-5b depicts the correct and extra orderings inferred in Scenario 2. The dotted lines represent the number of orderings in the true specification.

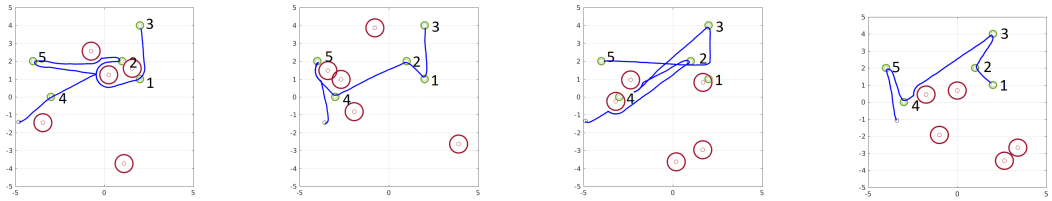
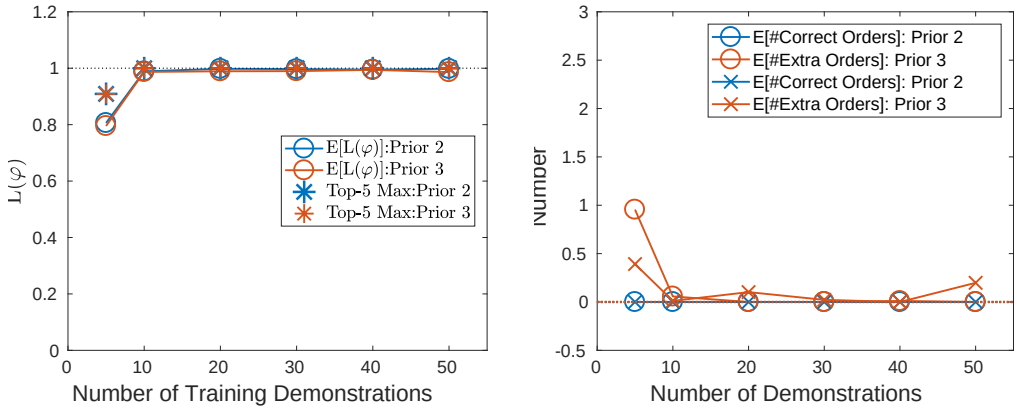


Figure 4-6: Example trajectories from Scenario 3. The green circles denote the POIs; the red circles denote the threat avoidance zones.

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–developed by Kong et al. [66] for inferring parameterized STL formulas from observations–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 4.1, 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. Scaling



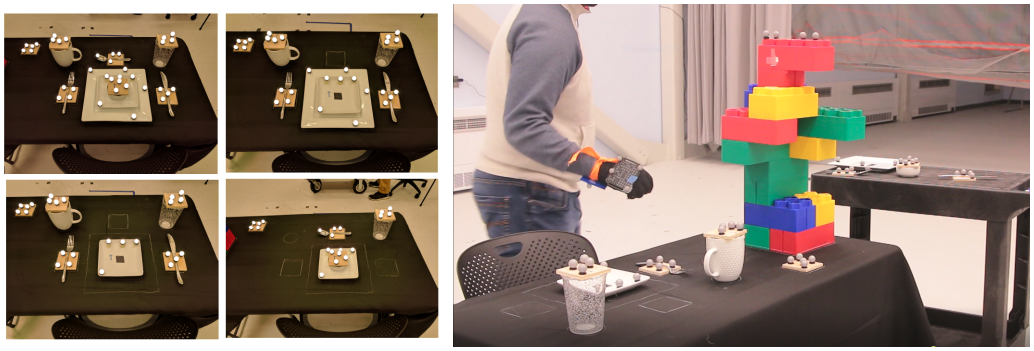
(a) Scenario 2 $L(\varphi)$ (b) Scenario 2 orders

Figure 4-7: Specification inference results for Scenario 3 within the synthetic domain. Figure 4-7a indicates the $L(\varphi)$ values for Scenario 3, and Figure 4-7b depicts the correct and extra orderings inferred in Scenario 3. The dotted lines represent the number of orderings in the true specification.

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 $n+1$.

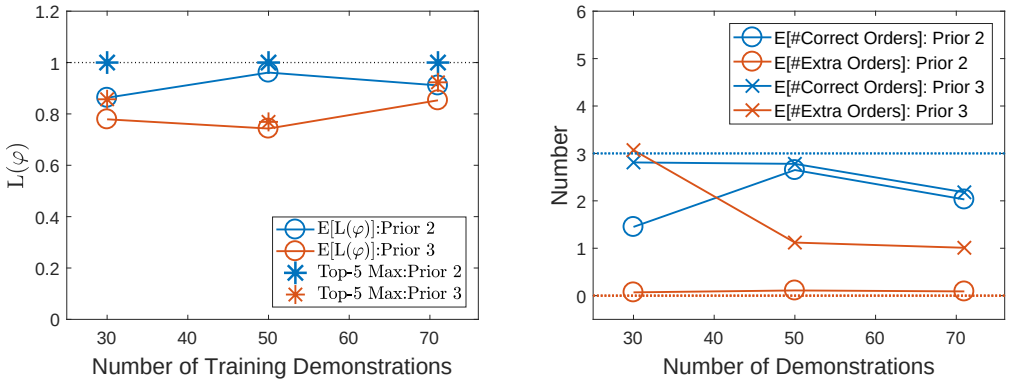
4.3.2 Table-Setting: A Decomposable Single-Agent Task

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



(a) (b)

Figure 4-8: Data collection setup for the dinner table task. Figure 4-8a depicts all the final configurations. Figure 4-8b depicts the demonstration setup. (Photographed by the authors in April 2017.)



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

(b) Dinner Table orders

Figure 4-9: Specification inference results for the table-setting task. Figure 4-9a depicts the $L(\varphi)$ values for the dinner table domain, with the dotted line representing the maximum possible value. Figure 4-9b depicts the correct and extra orderings inferred within this domain; the dotted lines represent the number of orderings in the true specification.

contact with a centerpiece. Figure 4-8a 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. We collected a total of 71 demonstrations, 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 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 $\boldsymbol{T}_i^O$. 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. [15].

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:

$$\omega_i = \begin{cases} \text{true}, & \boldsymbol{T}_i^{W_i} \in \boldsymbol{P}_i \\ \text{false}. & \text{otherwise}. \end{cases} \tag{4.14}$$

Thus, the proposition ω_i evaluates as true if the pose of the 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. 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:

$$\tau_c = \begin{cases} \text{false}, & \boldsymbol{T}_{h1}^c \in \boldsymbol{P}_c \vee \boldsymbol{T}_{h2}^c \in \boldsymbol{P}_c \\ \text{true}, & \text{otherwise} \end{cases} \tag{4.15}$$

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

Finally, condition propositions $p_i \; \forall \; i \in \Omega$ 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 that serving certain courses during a meal requires the 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 p_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, we observed that this condition was not satisfied in only four of the 71 demonstrations.

4.3.3 Evaluating Large Force Air-Combat Exercises

Large-force exercises (LFE) are combat flight training exercises that involve multiple aircraft groups, with each aircraft group playing a designated role in the completion of the mission by accomplishing individual objectives, while coordinating with other groups to achieve the mission objectives. Evaluating an 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 aircraft were included in the scenarios we simulated), along with configuration settings for each of those aircraft (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. They must first identify the transition points between predetermined scenario phases, then assess 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 aircraft 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 experts evaluate these missions 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 aircraft and a variable number of enemy aircraft and ground-based threats. Among the friendly aircraft, there are eight escort aircraft that are capable air-to-air fighters, eight SEAD (suppression of enemy air defenses) aircraft capable of attacking ground-based threats, and two strike aircraft that carry the ammunition that must be deployed in order to attack a designated ground target within a time-on-target (TOT) window. The aircraft' starting positions during a typical scenario are depicted in Figure 4-10. The role of the mission commander is to debrief the participants once an 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 to 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 aircraft 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 aircraft (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 concerning the mission objectives, based on 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 assess whether

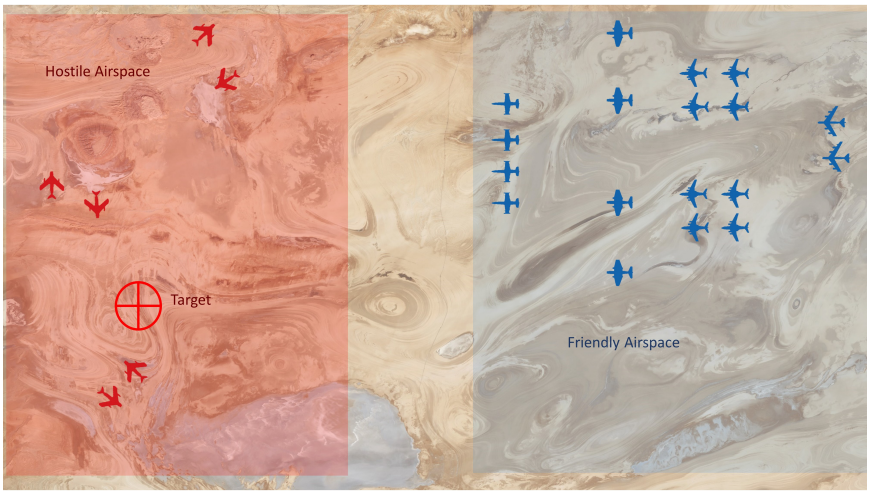


Figure 4-10: The starting configuration of a large-force exercise scenario. The red aircraft are the hostile forces, and the blue are friendly forces.

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 train a secondary classifier in one of the baselines simultaneously.

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 to classify 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 [55], along with their bidirectional variants [52]. Such models have shown state-of-the-art performance during time-series classification tasks [89]. 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 [121]. 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, 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 dataset was divided into four equal partitions. Three of the partitions were used for training (18 scenarios), and testing was 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 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 4.2, 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

Table 4.2: 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**

commonly occurring outcome in the training set for all values of inputs. Thus, the model could not 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 trained on the entire dataset. 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 aircraft 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 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 aircraft (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 aircraft 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 aircraft could be shot down, and also that the strikers were never shot down or fired upon.

4.4 Summary and Future Work

In this chapter, we presented a Bayesian model for inferring temporal logic specifications from labeled task executions observed by the learner. We proposed an expressive template as the hypothesis class for the specifications and developed structured priors over that expressed as probabilistic programs. We then proposed a domain-independent approximate likelihood function that adheres to the size principle. We demonstrated that our model allows for specification inference in both an inductive setting and in a setting where both acceptable and non-acceptable task executions are observed. We showed that a likelihood function that adheres to the size principle is key to successful specification inference in the inductive setting. We also demonstrated the efficacy of our model in learning specifications to set a dinner table in multiple valid configurations and assess the mission objectives of multi-aircraft air combat exercises.

While our model can infer a wide range of ground-truth specifications, adopting a template-based approach limits the class of learnable tasks. Expanding the library of templates and using disjunctive compositions is a natural first extension of our proposed model. However, leveraging grammatical inference first to discover useful templates automatically and then adding them to the inference library would be an important extension to expand the expressivity without hampering the relevance of the sampled formulas.

We also assumed that the labeling functions from the observed state x to the propositions α were pre-defined. However, learning the atomic propositions directly from observed data, or symbol learning [68], is crucial to adopt a specification inference approach to new domains. Hierarchical models for joint inference of relevant propositions and the formula

structure are important in future research.

Chapter 5

Planning with Uncertain Specifications

This chapter tackles the problem of planning with uncertain specifications as described in Section 3.2.2. We developed a novel planning formulation, PUnS, that allows an agent to generate its policy while reasoning about the epistemic uncertainty in the underlying task specifications. Within the scope of this thesis, the uncertainty arises from the ambiguous nature of natural interaction modalities like demonstrations and assessments. In these modalities, the information about the underlying specifications is provided implicitly. However, there may be other sources of uncertainty, such as specifications based on preferences elicited from multiple experts [63], or formal specifications derived from statements expressed in natural language [88, 51]. The PUnS formulation is generally compatible with scenarios where an agent maintains a belief over a finite set of logical formulas conforming to a well-defined fragment of LTL.

A key challenge for the PUnS problem is defining the semantics of satisfying a belief over logical formulas in contrast to satisfying a single logical formula; we discuss this in Section 5.1. The algorithms for expressing PUnS problems as equivalent reward machines [129, 128] are described in Section 5.2. We characterize the interactions between the nature of belief distributions and the choice of the evaluation criterion. We then demonstrate solving a multi-step robotic manipulation task through the PUnS formulation in Section 5.3.

5.1 Satisfying a Belief over Logical Formulas

Recall from Section 3.2 that the learner always maintains a belief over logical specifications, $P(\varphi)$. To be compatible with the PUnS formulation, the belief is restricted to LTL formulas belonging to the 'Obligations' class as defined by Manna and Pnueli [78]. Let, $\varphi_{obligation}$ represent the set of all LTL formulas belonging to the 'Obligations' class. The admissible beliefs are defined as a probability mass function $P : \varphi_{obligation} \to [0,1]$. We further require the discrete probability distribution to maintain support ($P(\varphi) > 0$) over a finite set of LTL formulas $\{\varphi\} \in \varphi_{obligation}$. This distribution represents the learner's degree of belief that a candidate formula $\varphi \in \{\varphi\}$ encodes the ground truth specification.

A single LTL formula can be satisfied, dissatisfied, or undecided; however, satisfaction semantics over a distribution of LTL formulas do not have a unique interpretation. We identify the following four evaluation criteria, which capture the semantics of satisfying a distribution over specifications, and formulate each as a non-Markovian reward function $R_{P(\varphi)} : [\boldsymbol{\alpha}] \to \mathbb{R}$. The evaluation criteria are as follows:

1. **Most Likely**: This criteria entails executions that satisfy the formula with the largest probability as per $P(\varphi)$. As a reward, this is represented as follows:

$$\begin{aligned} R_{\text{most likely}}([\boldsymbol{\alpha}]; P(\varphi)) &= \mathbb{1}([\boldsymbol{\alpha}] \models \varphi^*) \\ \text{where } \varphi^* &= \arg\max_{\varphi \in \{\varphi\}} P(\varphi). \end{aligned} \tag{5.1}$$

   Here

$$\mathbb{1}([\boldsymbol{\alpha}] \models \varphi) = \begin{cases} 1, & \text{if } [\boldsymbol{\alpha}] \models \varphi \\ -1, & \text{otherwise.} \end{cases} \tag{5.2}$$

   Here only the most likely specification in the belief is considered during planning. The planner will disregard whether the other formulas in the support of the belief are satisfied. This approach makes the PUnS formulation compatible with any

methodology that generates robot behavior from task specifications expressed as a single logical formula. In scenarios where the learner's belief distribution has a very low entropy with most of the probability mass concentrated on a single formula, this criterion can alleviate the computational overhead associated with reasoning about the entire belief distribution.

2. **Maximum coverage:** This criteria entails executions that satisfy the maximum number of formulas in support of the distribution $P(\varphi)$. As a reward function, it is represented as follows:

$$R_{\text{max coverage}}([\alpha];P(\varphi)) = \sum_{\varphi\in\{\varphi\}} \mathbb{1}([\alpha] \models \varphi). \tag{5.3}$$

This criterion is best suited for scenarios where the belief distribution results from preference elicitations from multiple experts, but with no preference over them, such as the approach followed by Kim et al. [63] where it is desirable to satisfy the largest common set of specifications.

3. **Minimum regret:** This criteria entails executions that maximize the hypothesis-averaged satisfaction of the formulas in support of $P(\varphi)$. As a reward function, this is represented as follows:

$$R_{\text{min}_\text{r}\text{egret}}([\alpha];P(\varphi)) = \sum_{\varphi\in\{\varphi\}} P(\varphi)\mathbb{1}([\alpha] \models \varphi) \tag{5.4}$$

Maximizing this criterion results in a task execution that has the largest probability of being acceptable as per the learner's belief of the true task specification. This makes the *minimum regret* criterion best suited to applications involving a Bayesian approach towards uncertainty in task specification, the setting considered in this dissertation.

4. **Chance constrained:** Suppose the maximum probability of failure is set to δ, with φ^δ defined as the set of formulas such that $\sum_{\varphi\in\varphi^\delta} P(\varphi) \geq 1-\delta$; and $P(\varphi') \leq P(\varphi)\ \forall\ \varphi' \notin \varphi^\delta, \varphi \in \varphi^\delta$. This is equivalent to selecting the most-likely formulas

until the cumulative probability density exceeds the risk threshold. As a reward, this is represented as follows:

$$R_{\text{cc}}([\boldsymbol{\alpha}];P(\varphi)) = \sum_{\varphi \in \varphi^{\delta}} P(\varphi)\mathbb{1}([\boldsymbol{\alpha}] \models \varphi). \tag{5.5}$$

The chance-constrained criterion offers a trade-off between risk aversion and computational overhead of reasoning about multiple LTL formulas within the support of $P(\varphi)$. For $\delta = 0$, this corresponds to the *minimum regret* criterion. As the value of δ approaches 1, the formulas in the support $\{\varphi\}$ are sequentially disregarded till only a single formula is relevant to computing the policy; thus coinciding with the *most likely* criterion.

Each of these four criteria represents a valid interpretation of satisfying a belief over LTL formulas, with the choice between the criteria dependent upon the relevant application. Broadly, the choice of the criterion represents a trade-off between flexibility in task execution by reducing constraints and risk aversion in terms of generating task executions that dissatisfy a subset of the support of the belief distribution. The impact of the choice of the evaluation criterion is studied in greater detail in Section 5.3.1.

5.2 The PUnS Formulation

An instance of a PUnS problem is concretely defined as follows:

Definition 1 *A PUnS problem is defined by the tuple* $\langle \mathcal{M}_{\mathcal{X}}, f, P(\varphi), R_{P(\varphi)} \rangle$*, where:* $\mathcal{M}_{\mathcal{X}} = \langle \mathcal{X}, \mathcal{A}, T_{\mathcal{X}} \rangle$ *represents the planning environment as an MDP sans a reward function;* $f : \mathcal{X} \to \{0,1\}^{n_{prop}}$ *is the labeling function that maps the state of the environment to the truth value of* n_{prop} *Boolean propositions;* $P(\varphi)$ *represents a belief distribution over LTL formulas with support over a finite set of of formulas,* $\{\varphi\}$ *constructed from the same propositions that are the output of* f*; and* $R_{P(\varphi)} : [\boldsymbol{\alpha}] \to \mathbb{R}$ *is the reward function that maps a sequence of truth values of the propositions,* $\boldsymbol{\alpha} \in \{0,1\}^{n_{prop}}$*, to a real-valued reward.*

Consider a planning domain representing the task of setting a dinner table, with multiple objects accessible to the robot, including a fork, a bowl, and a plate. The environment MDP, $\mathcal{M}_\mathcal{X}$, consists of a discrete state space, $\mathcal{X}$, that encodes whether each object is correctly placed; a discrete action space, $\mathcal{A}$, that encodes the selection of the object to be placed next; and the transition function, $T_\mathcal{X}$, which encodes how the action selection affects a change in the state. The acceptability of the task execution is evaluated using the vector of Boolean propositions, $\boldsymbol{\alpha} = [\mathit{Fork},\ \mathit{Bowl},\ \mathit{Plate}]$, where each proposition represents whether that object was correctly placed on the table. An example of a belief over the task specifications is represented by the distribution $P(\varphi)$, whose support, $\{\varphi\} = \{\varphi_1 = \mathbf{G}\ \neg \mathit{Fork} \wedge \mathbf{F}\ \mathit{Bowl} \wedge\ \neg \mathit{Bowl}\ \mathbf{U}\ \mathit{Plate},\ \varphi_2 = \mathbf{G}\ \neg \mathit{Fork} \wedge \mathbf{F}\ \mathit{Bowl}\}$, with probabilities as follows: $P(\varphi_1) = 0.3$, and $P(\varphi_2) = 0.7$. φ_1 encodes the specification that the fork must never be placed, the bowl must be placed eventually, and that the bowl must not be placed until the plate has been placed; φ_2 encodes that the fork must never be placed and that the bowl must be placed eventually. Thus, any task execution that satisfies φ_2 also satisfies φ_1; however, the converse is not true. To perform the task to best align with this belief over specifications, one must place the plate, then the bowl, and must not place the fork. The PUnS formulation formalizes this intuition.

Solving a given PUnS problem involves three steps: first, we construct a minimal state machine that is a Markov representation of all the formulas in the support $\{\varphi\}$; next, the reward function $R_{P(\varphi)}$ is transformed into an equivalent reward function over the states, thus yielding a PUnS reward machine (defined in Section 5.2.1); finally, the reward machine is composed with the environment MDP $\mathcal{M}_\mathcal{X}$ to yield an MDP equivalent of the original PUnS problem. The final policy for a PUnS problem can be computed by any solver that accepts an MDP as an input.

5.2.1 Reward Machines

Reward machines, proposed by Toro Icarte et al. [128, 56], are a form of state machine that can encode non-Markov specifications through states and transition tables. Following Toro Icarte et al. [128], we define a *simple* reward machine as follows:

Definition 2 *Given a set of n_{prop} Boolean propositions $\alpha \in \{0,1\}^{n_{prop}}$, a simple reward machine is defined as the tuple $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{F}, T, R \rangle$ where: $\mathcal{S}$ is a finite set of states; s_0 is the initial state; $\mathcal{F} \subset \mathcal{S}$ is the set of terminal states; $T : \mathcal{S} \times \{0,1\}^{n_{prop}} \times \mathcal{S} \to \{0,1\}$ is the transition function that determines the next state, given a current state and a truth value of the propositions at a given time step. $R : \mathcal{S} \times \{0,1\}^{n_{prop}}$ is the reward function that maps a transition to a real valued reward.*

Intuitively, a reward machine maintains a compact and sufficient representation of the execution history through its state-space $\mathcal{S}$; thus, it is a Markov representation of an equivalent non-Markov reward function. The reward machine can be composed with a Markov transition function of the environment to yield an equivalent Markov decision process (MDP), thus making reward machines compatible with any MDP or RL solvers. Toro Icarte et al. [127], and Camacho et al. [26] showed that reward machine representations can be constructed for a wide variety of specifications encoded as formal logic formulas. We demonstrate that every instance of a PUnS problem can be compiled into an equivalent reward machine.

5.2.2 Constructing a PUnS Reward Machine

Given a single LTL formula φ, a Büchi automaton can be constructed that accepts all traces that satisfy φ [132]. These automata are directed graphs where each node represents the LTL formula, φ' that the trace must satisfy from that point onwards in order to be accepted by the automaton. An edge labeled by the truth assignment α connects a node to its progression $\text{Prog}(\varphi', \alpha)$. Our decision to restrict the candidate LTL formulas to the obligation class ensures that the automaton representing φ is deterministic. In general, a formula of the obligation class, can be rewritten as a conjunction of *safe* and *co-safe* LTL formulas, $\varphi = \varphi_{safe} \wedge \varphi_{cosafe}$. Thus the automaton representing the formula φ will have terminal states that represent one of $\top$, $\bot$ or φ_{safe}. These terminal states are the reward generating states in the reward machine representing a single LTL formula of the obligations class defined according to Definition 2 as follows:

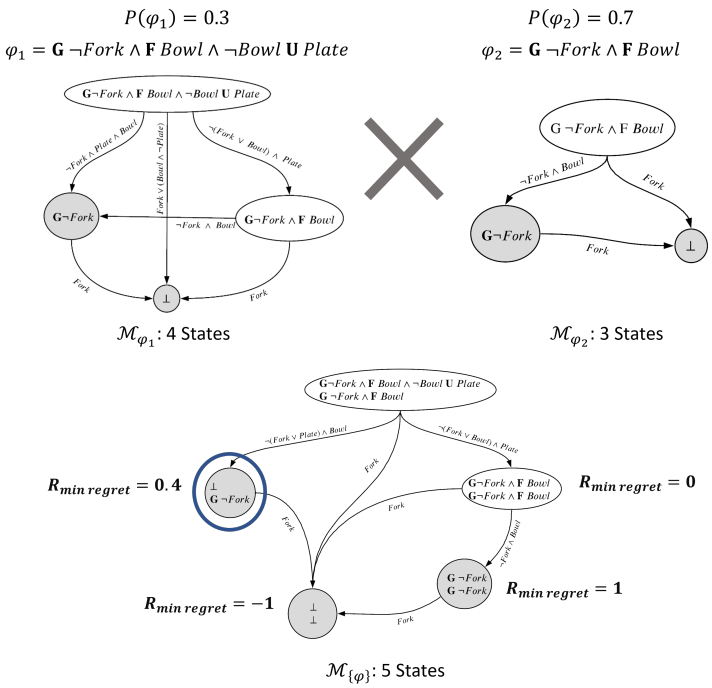


Figure 5-1: Example of a PUnS problem compilation with minimum regret rewards

$$\mathcal{M}_\varphi = \langle \{\varphi'\}, \varphi, \{\varphi'\}_{term}, T_\varphi, R_\varphi(\varphi') \rangle. \tag{5.6}$$

Here $\{\varphi'\}$ represents the set of enumerations of all progressions of the formula φ. The initial state of the reward machine corresponds to φ, the original formula. $\{\varphi'\}_{term}$ corresponds to the terminal state, i.e. states that have progressed to $\top$, $\bot$ or φ_{safe}. The transition function is defined as follows:

$$T_\varphi(\varphi_1', \alpha, \varphi_2') = \begin{cases} 1, & \text{if } \varphi_2' = \text{Prog}(\varphi_1', \alpha) \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

Finally, the reward function R_φ is defined as follows:

$$R_\varphi(\varphi') = \begin{cases} 1, & \text{if} \varphi' = \top or \varphi' = \varphi_{safe} \\ -1, & \text{if} \varphi' = \bot \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

Recall the table-setting example from Section 5.2. The reward machines encoding the satisfaction of φ_1 and φ_2 are depicted in Figure 5-1. Consider the reward machine $\mathcal{M}_{\varphi_1}$; $\mathcal{M}_{\varphi_1}$ is initially in the state labeled as $\mathbf{G}\neg \mathit{Fork} \wedge \mathbf{F}\ \mathit{Bowl} \wedge \neg \mathit{Bowl}\ \mathbf{U}\ \mathit{Plate}$. Once the *Plate* is placed on the table, the reward machine enters the state labeled by $\mathbf{G}\ \neg \mathit{Fork} \wedge \mathbf{F}\ \mathit{Bowl}$, which encodes the temporal property that in the future, the *Bowl* must eventually be placed on the table.

For a PUnS problem with the belief $P(\varphi)$ and support $\{\varphi\}$, the reward machine is constructed through a concatenation of the individual reward machines corresponding to the elements of $\{\varphi\}$. The reward machine corresponding to this PUnS problem is defined as follows:

$$\mathcal{M}_{\{\varphi\}} = \langle \{\langle \varphi' \rangle\}, \langle \varphi \rangle, \{\langle \varphi' \rangle\}_{term}, T_{\{\varphi\}}, R(\langle \varphi' \rangle) \rangle. \tag{5.9}$$

Here, $\{\langle \varphi' \rangle\}$ is the set of ordered tuples $\langle \varphi' \rangle$ that represent all progressions of the formulas contained in the support $\{\varphi\}$. In particular, φ'^i, represents the i^{th} formula in the tuple $\langle \varphi' \rangle$. The initial state of the reward machine is $\langle \varphi \rangle$ which is the ordered tuple of the unprogressed formulas in $\{\varphi\}$. $\{\langle \varphi' \rangle\}_{term}$ is the set of terminal states where each of the component formula has either been satisfied ($\top$), dissatisfied ($\bot$), or progressed to a safe-LTL formula. $T_{\{\varphi\}} : \{\varphi'\} \times \{0,1\}^{n_{prop}} \times \{\varphi'\} \to \{0,1\}$, is the transition function for state machine, and is defined as follows:

$$T_{\{\varphi\}}(\langle \varphi' \rangle_1, \alpha, \langle \varphi' \rangle_2) = \begin{cases} 1, & \text{if } \varphi_2'^i = \text{Prog}(\varphi_1^i, \alpha)\ \forall\ i \\ 0, & \text{otherwise.} \end{cases} \tag{5.10}$$

$R(\langle \varphi' \rangle) : \{\langle \varphi' \rangle\} \to \mathbb{R}$ is equivalent to $R([\alpha]; P(\varphi))$, but the domain of the function is now the set of the states of the reward machine. This transformation is straightforward, as any trace $[\alpha]$ advances the reward machine to a unique state $\langle \varphi' \rangle$ due to the deterministic nature of the transition function. Note that the underlying state machine depends only on the support $\{\varphi\}$ of the belief and not on the specific probability values.

5.2.3 Defining PUnS Reward Functions

The reward functions for the PUnS reward machines are constructed based on the evaluation criteria proposed in Section 5.1. These reward functions are defined as follows:

1. **Most likely:** Let $\varphi^i = \arg\max_{\varphi \in \{\varphi\}} P(\varphi)$. The reward function corresponding to the *most likely* evaluation criterion is defined as follows:

$$R_{most\ likely}(\langle \varphi' \rangle; P(\varphi)) = \begin{cases} 1, & \text{if } \varphi'^i = \top \text{ or } \varphi'^i \in \text{safe} - \text{LTL} \\ -1, & \text{if } \varphi'^i = \bot \\ 0, & \text{otherwise.} \end{cases} \tag{5.11}$$

2. **Max coverage:** The max coverage reward function is defined as follows:

$$R_{max\ cover}(\langle \varphi' \rangle; P(\varphi)) = \begin{cases} \sum_i r(\varphi'^i), & \text{if } \langle \varphi' \rangle \in \{\langle \varphi' \rangle\}_{term} \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

   Here $r(\varphi'^i)$ is defined as follows:

$$r(\varphi'^i) = \begin{cases} 1, & \text{if } \varphi'^i = \top \text{ or } \varphi'^i \in \text{safe} - \text{LTL} \\ -1, & \text{if } \varphi'^i = \bot. \end{cases} \tag{5.13}$$

3. **Minimum regret** The reward function based on the *minimum regret* criterion is defined as follows:

$$R_{min\ regret}(\langle \varphi' \rangle; P(\varphi)) = \begin{cases} \sum_i P(\varphi^i) r(\varphi'^i), & \text{if } \langle \varphi' \rangle \in \{\langle \varphi' \rangle\}_{term} \\ 0, & \text{otherwise.} \end{cases} \tag{5.14}$$

   Here $r(\varphi'^i)$ is defined as per Eq 5.13.

4. **Chance-constrained** Recall the definition of the chance-constrained set $\{\varphi\}^\delta$ from eq 5.5. The chance constrained reward function is defined as follows:

$$R_{CC}(\langle\varphi'\rangle;P(\varphi),\delta) = \begin{cases} \sum_i P(\varphi^i) r(\varphi'^i;\delta), & \text{if } \langle\varphi'\rangle \in \{\langle\varphi'\rangle\}_{term} \\ 0, & \text{otherwise.} \end{cases} \tag{5.15}$$

Here $r(\varphi'^i,\delta)$ is defined as follows:

$$r(\varphi'^i,\delta) = \begin{cases} 1, & \text{if } \varphi'^i = \top \text{ or } \varphi'^i \in \text{safe}-\text{LTL and } \varphi'^i \in \{\varphi\}^\delta \\ -1, & \text{if } \varphi'^i = \bot \text{ and } \varphi'^i \in \{\varphi\}^\delta \\ 0, & \text{otherwise.} \end{cases} \tag{5.16}$$

5. **Desired state reward**: This reward function is best suited to guide the robot to a particular intended state of the reward machine. Given a set of reward machine states $\{\langle\varphi'\rangle\}$, the transition function $T_{\{\varphi\}}$, and a desired final state $\langle\varphi'\rangle^* \in \{\langle\varphi'\rangle\}$; let $\{\langle\varphi'\rangle\}_{path}$ represent the set of states that lie on a simple path connecting $\langle\varphi\rangle$ to $\langle\varphi'\rangle^*$; then the desired state reward $R_{\langle\varphi'\rangle^*}$ is defined as follows:

$$R_{\langle\varphi'\rangle^*}(\langle\varphi'\rangle) = \begin{cases} 1, & \text{if } \langle\varphi'\rangle = \langle\varphi'\rangle^* \\ 0, & \text{if } \langle\varphi'\rangle \in \{\langle\varphi'\rangle\}_{path} \\ -1, & \text{otherwise.} \end{cases} \tag{5.17}$$

In addition to the reward function, the set of terminal states must be modified as follows:

$$\{\langle\varphi'\rangle\}^*_{term} = \{\langle\varphi'\rangle\}_{term} \cup \{\langle\varphi'\rangle^*\} - \{\langle\varphi'\rangle\}_{path}. \tag{5.18}$$

This ensures that the desired end state is a part of the terminal states of the modified reward machine, and none of the states that are on a transition path from the initial state to the desired state are in the modified set of terminal states. Thus a policy computed for the reward machine $\mathcal{M}_{\{\varphi\}} = \langle\{\langle\varphi'\rangle\},\langle\varphi\rangle,\{\langle\varphi'\rangle\}^*_{term},T_{\{\varphi\}},R_{\langle\varphi'\rangle^*}\rangle$ drives the robot towards the desired state.

5.2.4 Computing PUnS Policy

To compute the policy to solve a PUnS problem, the compiled reward machine $\mathcal{M}_{\{\varphi\}}$ is composed with the environment MDP $\mathcal{M}_{\mathcal{X}}$ to construct the MDP equivalent of the original PUnS problem $\mathcal{M}_{PUnS} = \langle \mathcal{X} \times \{\langle \varphi' \rangle\}, \mathcal{A}, T_{PUnS}, R(\langle \varphi' \rangle) \rangle$. Here the state-space is the Cartesian product of the states-spaces of the environment MDP $\mathcal{M}_{\mathcal{X}}$ and the reward machine $\mathcal{M}_{\{\varphi\}}$, the action space is identical to the action space of $\mathcal{M}_{\mathcal{X}}$, and the reward is only a function of the state of the reward machine. The transition function T_{PUnS} is defined as follows:

$$T_{PUnS}(\langle \langle \varphi' \rangle_1, x_1 \rangle, a, \langle \langle \varphi' \rangle_2, x_2 \rangle) = T_{\{\varphi\}}(\langle \varphi' \rangle_1, f(x_2), \langle \varphi' \rangle_2) \times T_{\mathcal{X}}(x_1, a, x_2). \tag{5.19}$$

Here f is the labeling function as defined in Definition 1. $\mathcal{M}_{PUnS}$ is compatible with any reinforcement learning or planning algorithm that accepts an MDP problem definition.

5.2.5 Counterfactual updates in a model-free setting

Toro Icarte et al. [56, 127] demonstrated that reward machines allow for off-policy updates for each state in the reward machine. Constructing $\mathcal{M}_{Spec}$ as a composition of $\mathcal{M}_{\mathcal{X}}$ and $\mathcal{M}_{\{\varphi\}}$ results in the following properties: the reward function is only dependent upon $\langle \varphi \rangle$, the state of $\mathcal{M}_{\{\varphi\}}$; the action availability only depends upon x, the state of $\mathcal{M}_{\mathcal{X}}$; and the stochasticity of transitions is only in $T_{\mathcal{X}}$, as $T_{\{\varphi\}}$ is deterministic. These properties allow us to exploit the underlying structure of $\mathcal{M}_{Spec}$ in a model-free learning setting. Let an action $a \in \boldsymbol{A}$ from state $x_1 \in \mathcal{X}$ result in a state $x_2 \in \mathcal{X}$. As $T_{\{\varphi\}}$ is deterministic, we can use this action update to apply a Q-function update (Equation 3.4) to all states described by $\langle \langle \boldsymbol{\varphi'} \rangle, x_1 \rangle \; \forall \; \langle \boldsymbol{\varphi'} \rangle \in \{\langle \boldsymbol{\varphi} \rangle\}$.

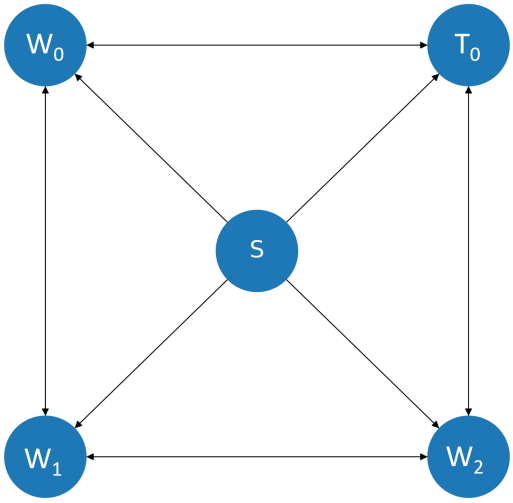


Figure 5-2: The transition diagram for the synthetic task with five states.

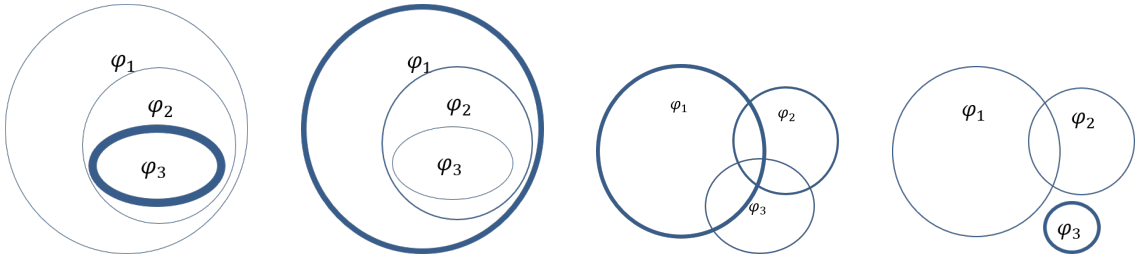
5.3 Experimental Evaluation

We first explored how the choice of criteria represented by Equations 5.1, 5.3, 5.4, and 5.5 results in qualitatively different performance by trained RL agents. This was conducted within the synthetic threats and waypoints domain used in Section 4.3.1, where it was easy to vary the problem size and the belief distribution and to compute the final policy for different problem variants rapidly. Then, we demonstrate how the PUnS compilation algorithm can serve to train an agent on a real-world task involving setting a dinner table with specifications inferred from human demonstrations, as per Shah et al. [111]. We also demonstrate the value of counterfactual Q-value updates for speeding up the agent's learning curve.

5.3.1 Interactions Between Beliefs and Evaluation Criteria

The choice of the evaluation criterion impacts the executions it entails based on the nature of the distribution $P(\varphi)$. Figure 5-3 depicts examples of different distribution types. Each figure is a Venn diagram where each formula φ_i represents a set of executions that satisfy φ_i. The size of the set represents the number of execution traces that satisfy the given formula, while the thickness of the set boundary represents its probability. Consider the simple discrete environment depicted in Figure 5-2: there are five states, with the start state

in the center labeled 'S' and the four corner states labeled "T_0", "W_0", "W_1", and "W_2". The agent can act to reach one of the four corner states from any other state, and that action is labeled according to the node it is attempting to reach.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

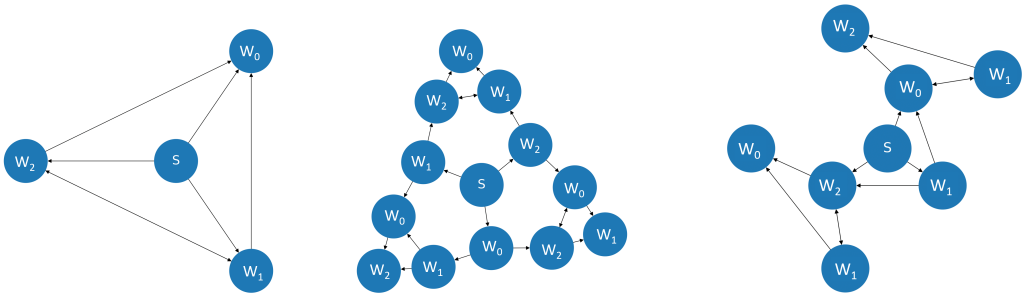
Figure 5-3: Comparisons between different types of distributions over specifications. In each case, the size of the set is proportional to the number of executions satisfying the specification, and the thickness of the boundary is proportional to the probability mass assigned to that specification.

Case 1: Figure 5-3a represents a distribution where the most restrictive formula of the three is also the most probable. All criteria will result in the agent attempting to perform executions that adhere to the most restrictive specification.

Case 2: Figure 5-3b represents a distribution where the most likely formula is the least restrictive. The *minimum regret* and *maximum coverage* rewards will result in the agent producing executions that satisfy φ_3, the most restrictive formula; however, using the *most likely* criteria will only generate executions that satisfy φ_1. With the chance-constrained policy, the agent begins by satisfying φ_3 and relaxes the satisfactions as risk tolerance is decreased, eventually satisfying φ_1 but not necessarily φ_2 or φ_3.

Case 3: Case 3 represents three specifications that share a common subset but also have subsets that satisfy neither of the other specifications. Let the scenario specification be $\{\varphi\} = \{\mathbf{G}\neg T_0 \wedge \mathbf{F}W_0, \mathbf{G}\neg T_0 \wedge \mathbf{F}W_1, \mathbf{G}\neg T_0 \wedge \mathbf{F}W_2\}$ with assigned probabilities to each of $0.4, 0.25,$ and 0.35, respectively. These specifications correspond to always avoiding "T_0" and visiting either "W_0", "W_1", or "W_2". For each evaluation criteria defined in Section 5.2.3, the Q-value function was estimated using $\gamma = 0.95$ and an ε-greedy exploration policy. A softmax policy with temperature parameter 0.02 was used to train the agent, and the resultant exploration graph of the agent was recorded. The *most likely* criterion requires only the first formula in $\{\varphi\}$ to be satisfied; thus, the agent will necessarily visit "W_0" but may or

may not visit "W_1" or "W_2", as depicted in Figure 5-4a. With either *maximum coverage* or *minimum regret* serving as the reward function, the agent tries to complete executions that satisfy all three specifications simultaneously. Therefore, each task execution ends with the agent visiting all three nodes in all possible orders, as depicted in Figure 5-4b. Finally, in the chance-constrained setting with risk level $\delta = 0.3$, the automaton compiler drops the second specification; the resulting task executions always visit "W_0" and "W_2" but not necessarily "W_1", as depicted in Figure 5-4c.

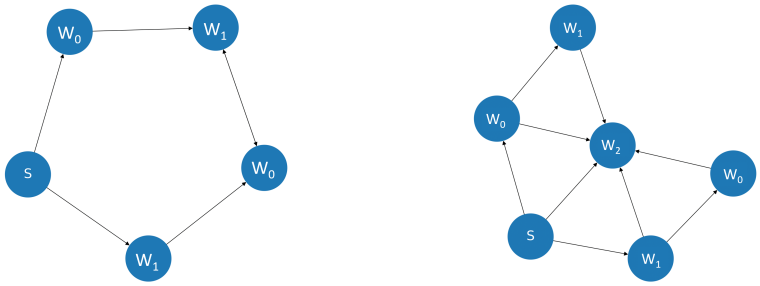


(a) Case 3: *most likely* (b) Case 3: *min regret* (c) Case 3: *chance constrained*

Figure 5-4: Exploration graphs for Case 3 within the synthetic domain. Figures 5-4a, 5-4b, and 5-5a depict the exploration graph of agents trained with different evaluation criteria for distributions with an intersecting set of satisfying executions.

Case 4: Case 4 depicts a distribution where an intersecting subset does not exist. Let the scenario specifications be $\{\varphi\} = \{\mathbf{G}\neg T0 \wedge \mathbf{G}\neg W_2 \wedge \mathbf{F}W_1, \mathbf{G}\neg T_0 \wedge \mathbf{G}\neg W_2 \wedge \mathbf{F}W_1, \mathbf{G}\neg T_0 \wedge \mathbf{F}W_2\}$, with probabilities assigned to each of $0.05, 0.15,$ and 0.8, respectively. The first two formulas correspond to the agent visiting either "W_1" or "W_0" but not "W_2". The third specification is satisfied when the agent visits "W_2"; thus, any execution that satisfies the third formula will not satisfy the first two. The first two formulas also have an intersecting set of satisfying executions when both "W_0" and "W_1" are visited, corresponding to Case 4 from Figure 5-3d. Optimizing for *max coverage* will result in the agent satisfying both the first and the second formula but ignoring the third, as depicted in Figure 5-5a. However, when using the *minimum regret* formulation, the probability of the third specification is higher than the combined probability of the first two formulas; thus, a policy learned to optimize *minimum regret* will ignore the first two formulas and always end an episode by visiting "$W2$", as depicted in Figure 5-5b. The specific examples and exploration graphs for

the agents in each of the scenarios in Figure 5-3 and for each reward formulation in Section 5.2.3 are provided in the supplemental materials.



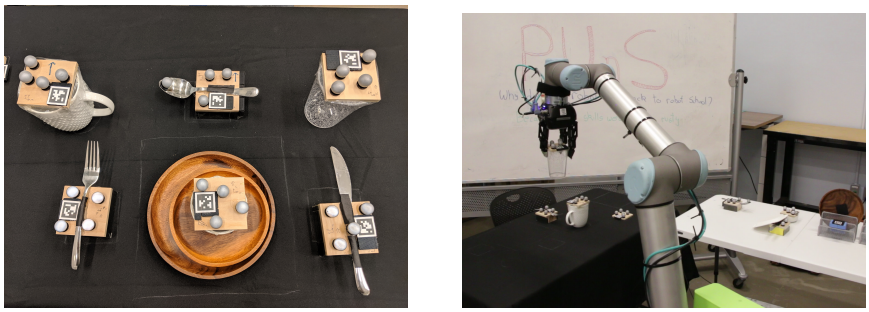
(a) Case 4: *max coverage* (b) Case 4: *min regret*

Figure 5-5: Exploration graphs for Case 4 within the synthetic domain. Figures 5-5a and 5-5b depict the exploration graph of agents trained with different evaluation criteria for distributions without an intersecting set of satisfying executions.

5.3.2 Demonstration: Multi-Step Manipulation Task

To demonstrate the utility of PUnS, we formulated the task of setting a dinner table as an instance of the PUnS problem. We used the posterior distribution inferred from 30 demonstrations (Section 4.3.1) as the belief distribution for this task. We used some newer dinner table objects that were better suited for robotic manipulation and shatter-resistant. The desired final configuration of the dinner table objects is depicted in Figure 5-6a. Recall that the successful completion of this task relied on placing all the dinner table objects in their correct final positions; and on stacking the dinner plate, small plate, and bowl on top of each other.

For the purpose of planning, the task environment MDP $\mathcal{M}_\chi$ was simulated. Its state was defined by the truth values of the eight propositions defined above; thus, it had 256 unique states. The action space of the robot was the choice of which object to place next. Once an action was selected, it had an 80% chance of success as per the simulated transitions. The posterior distribution inferred from 30 training demonstrations had the largest uncertainty in the true specification. This distribution $P(\varphi)$ had 25 unique formulas in its support $\{\varphi\}$. As per the expected value of the intersection over union metric, the belief was 85% similar to



(a) Desired final configuration (b) Task setup

Figure 5-6: Robot setup for the table setting task. Figure 5-6a depicts the desired final configuration of objects. Figure 5-6b presents the UR-10 arm performing the table-setting task.

the true specification. The true specification itself was part of the support but was only the fourth most likely formula, as per the distribution.

The reward machine $\mathcal{M}_{\{\varphi\}}$ compiled from $P(\varphi)$ had 3,025 distinct states; thus, the cross-product of $\mathcal{M}_{\{\varphi\}}$ and $\mathcal{M}_{\mathcal{X}}$ yielded $\mathcal{M}_{Spec}$ with $774,400$ unique states and the same action space as $\mathcal{M}_{\mathcal{X}}$. We chose the *minimum regret* criteria to construct the reward function and trained two learning agents using Q-learning with an ε-greedy policy ($\varepsilon = 0.3$): one with and one without off-policy updates. We evaluated the agent at the end of every training episode using an agent initialized with softmax policy (the temperature parameter was set to 0.01). The agent was allowed to execute 50 test episodes, and the terminal value of the reward function was recorded for each; this was replicated 10 times for each agent. We conducted all evaluations on a desktop with i7-7700K and 16 GB of RAM.

The statistics of the learning curve are depicted in Figure 5-7. The solid line represents the median value of terminal reward across evaluations collected from all training runs. The error bounds indicate interquartile range. The maximum value of the terminal reward is 1 when all formulas in the support $\{\varphi\}$ are satisfied, and the minimum value is -1 when all formulas are not satisfied. The learning curves indicate that the agent that performed counterfactual Q-value updates (Section 5.2.5) learned faster and had less variability in its task performance across training runs compared with the one that did not perform counterfactual updates. This provides additional empirical evidence that such counterfactual updates improve the sample complexity as first observed by Toro Icarte et al. [127, 56].

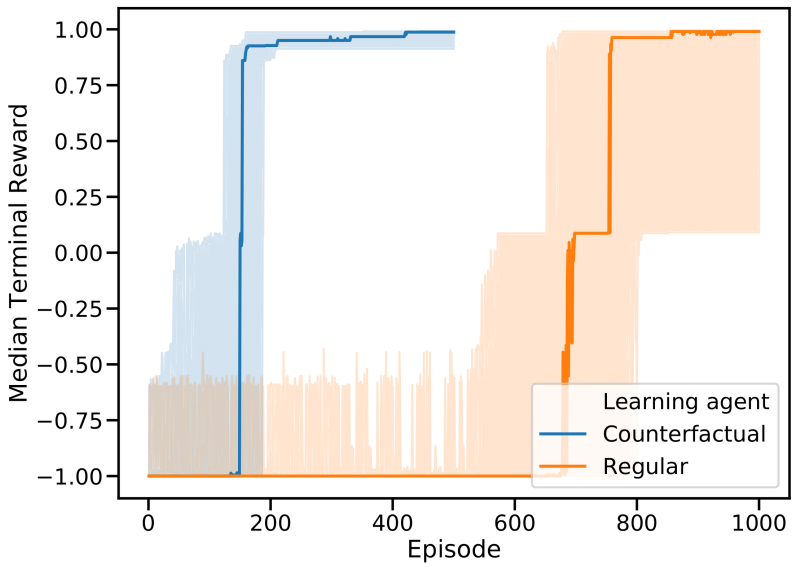


Figure 5-7: Learning curves for the computing the table-setting task policy. The curve depicts the median and the interquartile range of the terminal rewards.

We implemented the learned policy with predesigned motion primitives on a UR-10 robotic arm. We observed during evaluation runs that the robot never attempted to violate any temporal ordering constraint. The stochastic policy also made it robust to some environmental disturbances. For example, if one of the objects was occluded, the robot finished placing the other objects before waiting for the occluded object to become visible again.[1]

Next, to examine the trade-off between the creativity of performing the task and risk aversion, we repeated the training and testing with the $\mathcal{M}_{\{\varphi\}}$ compiled with the *most likely* and the *chance constrained* criteria with $\delta = \{0.1, 0.3\}$. For each of the trained agents, we recorded 20 physical executions by deploying the policy on the robot, and we also ran 20000 simulated test episodes for each instance of the PUnS MDP $\mathcal{M}_{Spec}$. We recorded the number of unique placement sequences and the number of specification violations during the simulated and physical test runs. The results are tabulated in Tables 5.2 and 5.3 respectively.

Assuming that the dinner plate, the small plate and the bowl must be placed in that partial order, there are 6720 unique, valid orderings for placing the eight objects. The policies trained as per *min regret* and both the *chance constrained* criteria generated 20 unique

[1]Example executions can be viewed at `https://youtu.be/LrIh_jbnfmo`

Reward Type	Formulas Included	$\mathcal{M}_{\{\varphi\}}$ States
Min Regret	25	3025
Most Likely	1	193
Chance constained (d = 0.1)	4	449
Change Constrained (d = 0.3)	3	353

Table 5.1: Relation between evaluation criteria choice, formulas considered in reward machine construction, and states of the reward machine.

orderings in the physical test executions. The simulated tests reveal that the policy trained in accordance with *minimum regret* criterion executes the task with fewer unique orders than the policy trained with both the *chance constrained* criteria. Both the physical executions and the simulations reveal that the policy trained with the *most likely* criterion performs the task with many constraint violations because the most likely formula does not include an ordering constraint existing in the ground truth specification. This demonstrates a three-way trade-off between computational complexity in terms of the additional states to consider while planning, the creativity displayed by the policy in terms of the unique executions discovered, and the risk of specification violation. The *min regret* policy is the most risk-averse but also the least flexible, while the *chance constrained* policies demonstrate higher creativity but with more constraint violations.

Reward Type	True Valid Orders	Successful Executions (On Robot)	Constraint Violations (On Robot)	Unique Orders (On Robot)
Min Regret	6720	20	0	20
Most Likely	6720	12	8	20
Chance constained (d = 0.1)	6720	20	0	20
Change Constrained (d = 0.3)	6720	20	0	20

Table 5.2: Successful task executions, and policy flexibility as demonstrated on a physical robot.

Reward Type	True Valid Orders	Successful Executions (Simulation)	Constraint Violations (Simulation)	Unique Orders (Simulation)
Min Regret	6720	19997	3	1962
Most Likely	6720	9920	10080	10215
Chance constained (d = 0.1)	6720	19995	5	4253
Change Constrained (d = 0.3)	6720	19987	13	4882

Table 5.3: Successful task executions and policy fliexibility as evaluated in simulations.

5.4 Summary and Future Directions

This chapter introduced planning with uncertain specifications (PUnS), a novel problem formulation that allows policy computation while reasoning about the epistemic uncertainty over what is the true task that the learner must perform. We introduced four evaluation criteria that capture the semantics of satisfying a belief over logical formulas. We further demonstrated that policies computed to optimize the reward functions derived from these evaluation criteria conform with the notion of hedging behaviors that satisfy multiple candidate formulas simultaneously. We also demonstrated the inherent trade-off between flexibility in task execution and risk-aversion that results from selecting the evaluation criteria. Finally, we demonstrated that every PUnS problem is equivalent to an MDP formed through the concatenation of the environment $\mathcal{M}_\mathcal{X}$, and the reward machine $\mathcal{M}_{\{\varphi\}}$ representing the belief over formulas. Finally, we proposed an algorithm to automatically construct the reward machine from the belief distribution $P(\varphi)$ and the choice of the evaluation criterion, based on breadth-first enumeration of the progression states of the formulas within the support $\{\varphi\}$ of the belief.

While we demonstrated that every PUnS problem can be framed as an equivalent MDP, computing sample-efficient policies for the resulting MDP remains an open problem. The learning agent acts within the environment $\mathcal{M}_\mathcal{X}$ while the rewards are obtained only when transition occurs within the reward machine $\mathcal{M}_{\{\varphi\}}$ that end in one of the terminal states; therefore, the reward structure for an MDP resulting from a PUnS problem is sparse. Toro-

Icarte et al. [128] have proposed approaches to shape rewards within the reward machines that allow for greater learnability. In addition to that, exploiting the compositional structure of the transitions of the reward machines presents a promising direction for future research.

Another key area of the extension is to relax the requirement for the candidate formulas to belong to the 'Obligation' class of temporal properties as defined by Manna and Pnueli [78]. The reward machine representation provides a sound realization of the underlying 'Obligation' temporal properties as a reward function; however, formulas that belong to classes higher in Manna and Pnueli's hierarchy demonstrate non-determinism in the underlying automata and the need for infinite planning horizons. Both these aspects make the construction of a reward machine that is a realization of the underlying properties challenging. The theoretical analysis of the expressivity of reward machines vis-a-vis the temporal properties is an important avenue of future research.

Chapter 6

Online Interactive Robot Training

Batch Bayesian specification inference (Chapter 4) deals with a learner that learns purely from observing labeled task executions provided by the teacher, whereas planning with uncertain specifications (PUnS) (Chapter 5) allows the learner to reason over the uncertainty in the underlying task specification and generate behavior that satisfies a maximal subset of the belief distribution. Together, these approaches allow a learner to infer task specifications passively from observational data, and then perform the task by planning with uncertain specifications. However, an ideal robot apprentice should be an active learner, i.e., it should be capable of guiding its learning by intelligently generating the data it learns from.

This chapter describes the development of the integrated framework formalized in Section 3.2.3. We start by demonstrating how the structure of the reward machine can be leveraged to identify a task execution whose acceptability label is most informative to the learner. We then demonstrate how to estimate the expected utility gain if the learner were to elicit a new demonstration from the teacher under the assumption of noisy rationality–i.e. taking the optimal decision with higher probability. The learner can compare these estimates of expected utility gain from an active query to eliciting a new demonstration from the teacher to identify the ideal modality for the next learning iteration. Finally, we instantiate utility functions based on principles of *uncertainty sampling*, *information gain*, and *model change* within the context of non-Markov tasks represented by PUnS reward machines. These technical contributions result in our proposed Bayesian framework where the learner learns iteratively, either from demonstrations provided by the teacher or from acceptability

assessments of its task execution. Further, the schedule of the learning modalities can either be fixed apriori or intelligently selected by the learner.

We then conduct simulation experiments within our framework to compare various learning modality schedules and assess the impact of the teacher's pedagogical selectivity. In particular, we demonstrate that a learner that can intelligently select its learning modality can compensate for a teacher who cannot provide a diverse set of teaching demonstrations.

Finally, we demonstrate our entire Bayesian framework on a real-world task of setting the table through a user study. Based on our study with 18 participants, all the participants could teach the robot to set the dinner table in the desired configuration without any errors within just five task executions, thus indicating the viability of deploying our framework towards real-world applications.

6.1 Multi-Modal Training Framework

In our framework (Figure, 6-1), the teacher intends to teach the desired task encoded by an LTL formula φ^* to a learner. Following a Bayesian approach, the learner maintains a belief over candidate LTL formulas $P(\varphi)$, defined by the probability mass function $P : \{\varphi\} \rightarrow [0,1]$. The support of the distribution $P(\varphi)$ is restricted to a finite set of LTL formulas, $\{\varphi\}$, where each formula φ belongs to the "Obligations" class of temporal properties [78].

The learner acts in and observes the environment encoded by the environment MDP $\mathcal{M}_X$ from as defined in Def 1. We assume that a set of n_{prop} propositional variables $\boldsymbol{\alpha}$ is sufficient to evaluate the truth values of φ^* and of all the formulas in the support of $P(\varphi)$ at all times. Thus task execution performed by either the teacher or the learner are represented by a sequence of environment states $[x]$ that is injectively mapped to a sequence of truth values of the propositions, $[\boldsymbol{\alpha}]$, through the labeling function $\boldsymbol{\alpha} = f(x)$.

The teacher can provide inputs to the learner by either providing a demonstration of successful task execution as per their intended task specification φ^* or assessing the acceptability of a task execution performed by the learner. Given the learner's current belief $P(\varphi)$, we model the teacher's demonstrations as a draw from the distribution $P_{teacher}([\boldsymbol{\alpha}] \mid \varphi^*; P(\varphi))$; i.e., we assume that the teacher's demonstrations are conditioned on the true

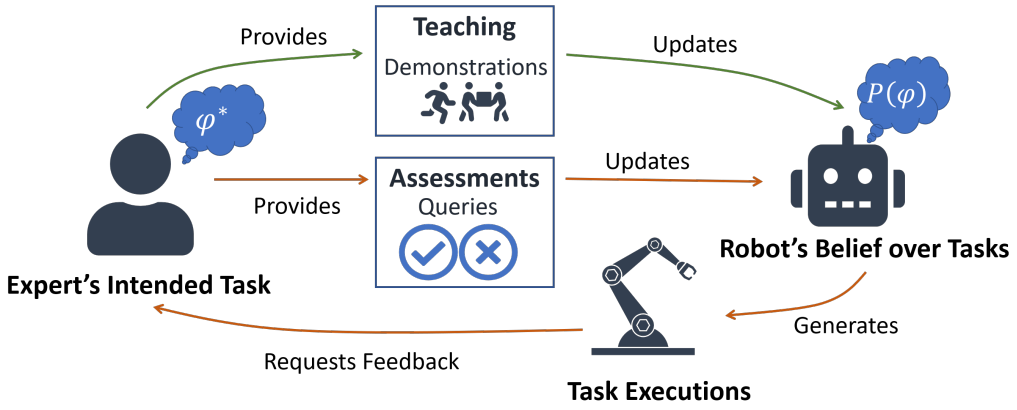


Figure 6-1: Our proposed Bayesian interactive learning framework that unifies learning from demonstrations provided by the teacher and using informative queries generated by the learner to refine the learner's belief. The green path depicts the teacher initiating training using task demonstrations. The orange path indicates the learner initiating training by demonstrating a task execution as a query requesting an assessment from the teacher.

task specification that they intend to teach the learner and the learner's current belief over task specifications. Further, we also assume that any demonstration provided by the teacher is acceptable with respect to the intended formula, φ^*. We also assume the teacher can assess the acceptability of task executions with respect to φ^* through the labeling function $\mathcal{L}_{\varphi^*}([\boldsymbol{\alpha}]) \in \{0,1\}$.

6.1.1 Modes of Learning

We adopt an iterative Bayesian formulation to update the belief of the learner. The prior belief $P(\varphi)_0$ of the learner is initialized to be one of the distributions over LTL formulas defined in Section 4.2.1. The teacher initializes the learner's belief $P(\varphi)_1 = P(\varphi \mid \mathcal{D}_0)$ over the intended task specifications by providing at least two acceptable task demonstrations, denoted by $\mathcal{D}_0 = \{\langle [\boldsymbol{\alpha}]_1, 1\rangle, \langle [\boldsymbol{\alpha}]_2, 1\rangle, \ldots\}$. All subsequent updates to the learner's belief occur iteratively, one task execution at a time. The data from each task execution, performed by either the teacher or the learner, is represented by the tuple $\langle [\boldsymbol{\alpha}], \mathcal{L}([\boldsymbol{\alpha}])\rangle$, representing the sequence of the truth values of the Boolean propositions, and the acceptability label. All task executions performed by the teacher are labeled as acceptable, and the executions performed by the learner are labeled by the teacher according to $\mathcal{L}_{\varphi^*}$. The posterior belief

conditioned on the data from each interaction is set as the prior belief for the subsequent Bayesian update:

$$P(\varphi)_{i+1} \leftarrow P(\varphi \mid \langle [\alpha], \mathcal{L}([\alpha]) \rangle)_i \tag{6.1}$$

The learner operates in two modes: *learning*, wherein the learner updates its belief conditioned on demonstrations provided by the teacher, or based on the teacher's assessments of the acceptability of its task executions; and *planning*, where it must compute the final policy based on the final belied $P(\varphi)_{final}$, once the teacher's input is not available to the learner. The computation of the final policy in the *planning* mode is an instance of the PUnS problem (Definition 1), $\langle \mathcal{M}_{\mathcal{X}}, f, P(\varphi)_{final}, R_{min\ regret} \rangle$.

In the *learning* mode, the learner can learn from either a task execution demonstrated by the teacher or from the teacher's assessment of the task execution performed by the robot. In Section 6.2.1, we define query selection strategies that allow the learner to identify a task execution, where the teacher's assessment would be the most beneficial in reducing the learner's uncertainty of the true task specification. However, the choice of the learning modality itself is a decision variable that impacts the rate at which the learner gains competency towards the desired task. We consider two approaches towards determining the learning modality at each iteration:

1. **Fixed modality schedule:** In this mode of learning, the number of iterations that the learner can use to update its beliefs are fixed, and so is the nature of the source of the task execution at each iteration. In this dissertation, we consider the two extremes of the fixed modality schedule once the learner's belief is initialized with a fixed number of demonstrations. First, the learner identifies and executes a query task execution following the active learning paradigm described in Section 6.2.1; we term this schedule as *Active*. Next, the learner only learns from acceptable demonstrations provided by the teacher; this schedule is called *Demonstrations*.

2. **Adaptive modality schedule:** In this mode of learning, the learner first decides what the ideal learning modality at each iteration would be based on a model of the teacher. The pedagogical selectivity of the teacher plays an important role in such a scenario.

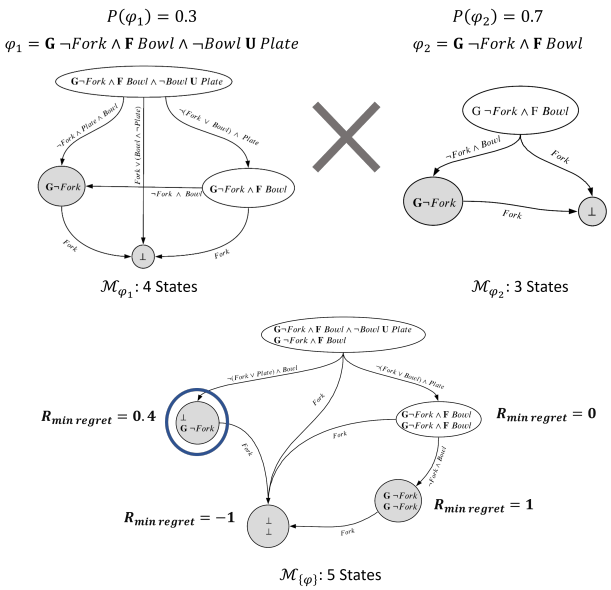


Figure 6-2: Example of selecting an informative query using a PUnS reward machine.The desired state reward function is indicated in blue.

We consider a noisy-pedagogical model of the teacher described in Section 6.3, that allows us to estimate the expected utility of a teacher's demonstration in contrast with that of an actively selected query execution. The learner leverages this to decide the learning modality to use during the next iteration as described in Section 6.2.3. We call this learning schedule, *Meta-Selection*.

6.2 Online Multi-Modal Learning

Consider the table setting task defined in Section 5.2. Uncertainty in whether φ_1 or φ_2 is the true formula would result in robot behavior that favors φ_1 under the PUnS formulation, as it is the more restrictive of the two. If the plate and the bowl are placed sequentially, that automatically satisfies the specification of placing just the bowl. However, if φ_2 were the ground truth specification, the learned policy overconstrains the robot's behavior.

Any task execution performed by the robot will progress the PUnS reward machine, $\mathcal{M}_{\{\varphi\}}$ (depicted in Figure 6-2), to one of the five states. An execution that ends in the states denoted by $\langle \perp, \perp \rangle$, or $\langle \mathbf{G}\ \neg\ Fork, \mathbf{G}\ \neg\ Fork \rangle$ has limited utility in refining the robot's belief as such an execution either satisfies or dissatisfies all the formulas in the support, $\{\varphi\}$.

However, the acceptability label for an execution that ends in the state $\langle \bot, \mathbf{G} \neg \mathit{Fork} \rangle$ is always informative, as it helps increase certainty towards one of the two formulas in the support $\{\varphi\}$.

If the learner had a choice to perform the following task execution, the learner would perform one that terminates in the state $\langle \bot, \mathbf{G} \neg \mathit{Fork} \rangle$, as the demonstrator's label is informative irrespective of whether this task execution was acceptable or not. Thus the learner must compute the utility of a query task execution over the possible *labels* that it might receive.

Conversely, a pedagogical demonstrator has the benefit of knowing the ground-truth formula and the learner's current belief, and the belief update model. In the scenario where φ_1 is the ground truth formula, the teacher can only provide demonstrations that terminate in the state $\langle \mathbf{G}\neg \mathit{Fork}, \mathbf{G} \neg \mathit{Fork} \rangle$; however, if φ_2 were the ground-truth formula, the teacher can provide demonstrations that terminate in either of the states $\langle \mathbf{G}\neg \mathit{Fork}, \mathbf{G} \neg \mathit{Fork} \rangle$ or $\langle \bot, \mathbf{G} \neg \mathit{Fork} \rangle$. However, with the knowledge of the learner's belief, a pedagogical teacher will choose to perform a demonstration that terminates in the more informative state that can help the learner refine its belief.

More generally, the states of the reward machine $\mathcal{M}_{\{\varphi\}}$ implicitly encode all possible truth values of the LTL specifications within the support, $\{\varphi\}$, of the learner's belief $P(\varphi)$. In the case of the learner eliciting an assessment of the query, the state is explicitly selected by the learner. To generate the query execution, once a state $\langle \varphi' \rangle^*$ is determined, the learner compiles a new PUnS reward machine, $\mathcal{M}_{\{\varphi\}}^{\langle \varphi' \rangle^*} = \langle \{\langle \varphi' \rangle\}, \langle \varphi \rangle, \{\langle \varphi' \rangle\}^*, T_{\{\varphi\}}, R_{\langle \varphi' \rangle^*} \rangle$. The optimal policy for $\mathcal{M}_{\{\varphi\}}^{\langle \varphi' \rangle^*}$ drives the learner to the desired informative state. A key benefit of using reward machines is that all functions of the trace, $[\boldsymbol{\alpha}]$, can be transformed to equivalent functions of the reward machine state $\langle \varphi' \rangle$, as the $[\boldsymbol{\alpha}]$ maps injectively to $\langle \varphi' \rangle$. Following this, we define the following shorthand for convenience. For any function f that accepts traces as domain $f([\boldsymbol{\alpha}])$, there will be an equivalent transformed function, $f([\boldsymbol{\alpha}]) \equiv f(\langle \varphi' \rangle)$ that accepts the set of states of the reward machine as its domain.

We developed models that allow the learner to estimate the expected utility of performing a query execution to seek specific refinements or allow the teacher to demonstrate diverse ways of acceptably performing the task execution. We first demonstrate how the learner can

leverage the structure of the reward machine to determine the expected utility of asking a query (Section 6.2.1) in comparison with that of seeking a demonstration from the teacher (Section 6.2.2). Next, in Section 6.2.4 we discuss the choices of utility functions that lead to various query selection strategies inspired by active learning approaches [106].

6.2.1 Determining Informative Queries

Let $J(\langle\varphi'\rangle, \mathcal{L}) : \{\langle\varphi'\rangle\} \times 0, 1 \to \mathbb{R}$ is a function of the reward machine state, $\langle\varphi'\rangle$, and the acceptability label, $\mathcal{L}$ of a task execution that ends in $\langle\varphi'\rangle$. We also assume that a trace and label with a larger value of $J(\langle\varphi'\rangle, \mathcal{L})$ is a more desirable source of data for learning.

If the learner performs a query task execution and elicits the acceptability label from the teacher, the expected utility of the query that ends in the state $\langle\varphi'\rangle$ over all possible labels that the teacher may provide is:

$$\mathbb{E}_{\mathcal{L}}[\![J]\!] = P(\mathcal{L}=1)J(\langle\varphi'\rangle, 1) + (1 - P(\mathcal{L}=1)J(\langle\varphi'\rangle, 0). \tag{6.2}$$

Here, given the learner's belief $P(\varphi)$, the probability of the acceptability label as estimated by the learner is $P(\mathcal{L}=1) = \mathbb{E}_{P(\varphi)}[\![\mathbb{1}([\alpha] \models \varphi)]\!]$. As the learner has complete control over what task execution to perform, the learner chooses to maximize the expected utility (Eq 6.2). Thus the state chosen, $\langle\varphi'\rangle^*$ by the learner, and the expected utility J_{query} from an active query are:

$$J_{query} = \max_{\langle\varphi'\rangle \in \{\langle\varphi'\rangle\}} \mathbb{E}_{\mathcal{L}}[\![J(\langle\varphi'\rangle, \mathcal{L})]\!], \text{ and} \tag{6.3}$$

$$\langle\varphi'\rangle^* = \arg\max_{\langle\varphi'\rangle \in \{\langle\varphi'\rangle\}} \mathbb{E}_{\mathcal{L}}[\![J(\langle\varphi'\rangle, \mathcal{L})]\!]. \tag{6.4}$$

6.2.2 Computing Expected Utility of Teacher's Demonstrations

Next, we must estimate the expected utility of allowing a teacher to provide a teaching demonstration, assuming that the teacher has access to the belief-updating algorithm of the learner and the learner's current belief. Note that the teacher will always provide an

acceptable task execution as a demonstration, therefore $\mathcal{L} = 1$, but the teacher can choose a trace that ends in an informative reward machine state $\langle \varphi' \rangle^*$. Thus the learner must compute the expected utility of a demonstration, given its belief over the true formula $P(\varphi)$, and a model of the teacher's pedagogical selection of the teaching state $\langle \varphi' \rangle^*$. The expected utility gain from a teacher's demonstrations is computed as follows:

$$J_{demo} = \mathbb{E}_{\langle \varphi' \rangle \in \{\langle \varphi' \rangle\}}[\![J(\langle \varphi' \rangle, 1)]\!] = \sum_{\langle \varphi' \rangle \in \{\langle \varphi' \rangle\}} J(\langle \varphi' \rangle, 1) P(\langle \varphi' \rangle) \tag{6.5}$$

Here the $P(\langle \varphi' \rangle)$ is the resulting distribution over the final state of the demonstration given the teacher's pedagogical model $P(\langle \varphi' \rangle \mid \varphi^*)$ and the learner's belief over the true task specification $P(\varphi)$, thus the expected utility of the teacher's demonstration is computed as follows:

$$J_{demo} = \sum_{\langle \varphi' \rangle \in \{\langle \varphi' \rangle\}} J(\langle \varphi' \rangle, 1) \sum_{\varphi \in \{\varphi\}} P(\langle \varphi' \rangle \mid \varphi) P(\varphi) \tag{6.6}$$

We discuss the teacher's pedagogical model in greater detail in Section 6.3.

6.2.3 Intelligent Learning Modality Selection (Meta-Choice)

An adaptive schedule of learning from teacher demonstrations and from acceptability assessments of the learner's performance can help the learner choose the 'right' modality for learning at each iteration. Similar to the active learning approach discussed in Section 6.2.1, the selection of the best learning modality requires the learner to estimate the relative utility of observing a demonstration and of eliciting an assessment of query execution.

During any given iteration, the learner computes expected utility from both eliciting a task demonstration from the teacher and seeking the teacher's assessment of a query execution performed by the learner. The learner selects the data source with the highest expected utility. As in the case of active learning, the learner can use uncertainty sampling, information gain, and model change heuristics to guide the selection of the learning modality, just as the learner can use them to determine an informative query execution.

6.2.4 Utility Functions

The query selection strategy is key to the performance of our interactive learning framework. We instantiated the utility functions inspired by approaches to active learning as surveyed by Settles [106]. We considered the following query selection strategies:

1. **Uncertainty Sampling:** The principle of uncertainty sampling [76] states that the ideal query for the active learner is the one where the current model is most ambivalent about the teacher's expected label. For binary acceptability labels, the probability that the query task execution is acceptable per the current belief should be closest to 0.5. Given the current belief of the learner $P(\varphi)_i$, the learners estimate of the acceptability of a given task execution $[\boldsymbol{\alpha}]$ is:

$$\mathbb{E}_{P(\varphi)_i}[\![[\boldsymbol{\alpha}] \models \varphi^*]\!] = \mathbb{E}_{P(\varphi)_i}[\![\mathbb{1}([\boldsymbol{\alpha}] \models \varphi)]\!] = 0.5 \times (1 + R_{\textit{min regret}}(\langle \varphi' \rangle)). \tag{6.7}$$

   Here $\langle \varphi' \rangle$ is the final state resulting from the progression of the reward machine $\mathcal{M}_{\{\varphi\}}$ with the trace $[\boldsymbol{\alpha}]$. Note that the *minimum regret* reward is linearly dependent on the model's evaluation of the probability of a trace satisfying the true specification. Thus, a probability of 0.5 corresponds to a minimum regret reward value of 0. Thus the utility function corresponding to uncertainty sampling is:

$$J_{US}(\langle \varphi' \rangle, \mathcal{L}) = - \mid R_{\textit{minregret}}(\langle \varphi' \rangle) \mid \tag{6.8}$$

2. **Information Gain:** The Shannon entropy [114], denoted by $H(\varphi)$ is a measure of the uncertainty in a random variable's distribution, and is:

$$H(\varphi) = \mathbb{E}_{P(\varphi)}[\![-P(\varphi)\log(P(\varphi))]\!]. \tag{6.9}$$

   Biyik et al. [19, 18] and Jeon et al. [57] have proposed active learning of reward functions in an MDP setting based on the principle of maximal information gain, i.e. selecting a query execution that would maximally reduce the conditional entropy in expectation over all possible expert labels. Note that any trace $[\boldsymbol{\alpha}]$ that ends in the state

$\langle \varphi' \rangle$, given the same label $\mathcal{L}([\boldsymbol{\alpha}])$ would result in an identical posterior distribution; i.e. $P(\varphi \mid [\boldsymbol{\alpha}], \mathcal{L}([\boldsymbol{\alpha}])) = P(\varphi \mid \langle \varphi' \rangle, \mathcal{L}(\langle \varphi' \rangle))$. The information gain for a trace ending in state, $\langle \varphi' \rangle$, with the teacher's acceptability label, $\mathcal{L}([\boldsymbol{\alpha}]) = \mathcal{L}(\langle \varphi' \rangle) = \mathcal{L}$ is:

$$J_H(\langle \varphi' \rangle, \mathcal{L}) = H(\varphi) - H(\varphi \mid \langle \varphi' \rangle, \mathcal{L}). \tag{6.10}$$

3. Maximum Model Change : The principle of maximum model change for active learning selects a data point that expects to have the largest impact on the learned model. Traditionally, this metric has been operationalized through the magnitude of the model gradient. However, we utilize the expected Jenson-Shannon distance between the prior and the posterior distribution as the metric for selecting the desired state for our purposes. The model change metric for a trace ending in the state, $\langle \varphi' \rangle$, with the teacher's acceptability label $\mathcal{L}$ is:

$$J_{JSD}(\langle \varphi' \rangle, \mathcal{L}) = JSD(P(\varphi) \parallel P(\varphi \mid \langle \varphi' \rangle, \mathcal{L}). \tag{6.11}$$

Each of the utility functions J_{US} (Eq: 6.8), J_H (Eq 6.10, or J_{JSD} (Eq 6.11) can be used as the utility functions to determine the most informative query that a learner should perform (Eq 6.4, or to compute the expected utility of observing a demonstration performed by the teacher (Eq 6.6).

6.3 Modeling Pedagogical Teaching Behavior

Finally, we discuss the learner's model of the teacher's pedagogical behavior. We assume that a teacher who has access to the learner's present belief and its mechanism for updating the belief can select training demonstrations that are more instructive than simply demonstrating task executions that satisfy the intended specification, independently of each other. We model the teacher as Boltzmann optimal [80, 81, 23], where the teacher's selection of the training example is proportional to the power of the posterior probability of the learner's belief over the true (or the most aligned) task specification. The teacher's demonstrations

are modeled as a draw from a distribution described as follows:

$$P(\langle \varphi' \rangle \mid \varphi^*) = \frac{P(\varphi^* \mid \langle \varphi' \rangle, 1; P(\varphi))^\gamma}{\sum_{\langle \varphi' \rangle \in \{\langle \varphi' \rangle\}_{sat}} P(\varphi^* \mid \langle \varphi' \rangle, 1; P(\varphi))^\gamma}. \tag{6.12}$$

Here $P(\varphi^* \mid \langle \varphi' \rangle, 1; P(\varphi))$ represents the posterior probability mass of the ground truth formula given a task execution demonstrated by the teacher (thus acceptable) that ends in the state $\langle \varphi' \rangle$ of the reward machine $\mathcal{M}_{\{\varphi\}}$, and the prior belief $P(\varphi)$. γ represents the pedagogical selectivity of the teacher; $\{\langle \varphi' \rangle\}_{sat}$ is the set of states of $\mathcal{M}_{\{\varphi\}}$ where the formula φ^* is satisfied. Note that the parameter γ determines the optimality of the teacher's selected state. Higher values of γ place a higher probability mass on the state that results in the largest posterior probability of the true formula; in the limit $\gamma \to \infty$, the teacher always selects the best possible state. $\gamma = 0$ represents the condition where the teacher independently selects any state where the ground truth formula is satisfied. $\gamma < 0$ models a teacher that shows acceptable demonstrations, but their demonstrations are very similar to the initial demonstrations shown, thus lacking in diversity.

We demonstrate the impact of the teacher's pedagogical selectivity on the learning curves and also the impact of an incorrect pedagogical model, where the learner either underestimates or overestimates the teacher's selectivity.

6.4 Simulation Experiments

Prior research has demonstrated the compatibility of reward machines with reinforcement learning approaches to accomplish tasks with a long planning horizon [127, 128, 26]. However, prior work in learning for non-Markov tasks has relied on pre-determined tasks. For our experiments, we evaluated the performance of our learning methods on a wide variety of procedurally sampled ground truth task specifications based on the priors defined in Section 4.2.1.

The environment, $\mathcal{M}_{\mathcal{X}}$ was based on a synthetic domain that allows for a variable number of threats (signifying constraints) and waypoints (signifying sub-tasks). This domain allows flexibility in terms of the number of propositions considered, the temporal structure of the

specifications and ensures that the generated specifications are satisfiable within the context of the environment. We believe that evaluating the learner's performance over a wider variety of ground truth formulas in the same domain is more valuable than evaluating the performance in different domains with a few ground truth formulas in each.

The first experiment intended to compare the performance of various query selection strategies within an active learning setting. Next, we evaluated the impact of the teacher's pedagogical selectivity when learning purely from demonstrations. We then evaluate the relative performance of learning purely from demonstrations compared to a setting where the learner is allowed to choose the learning modality at each iteration. Finally, we perform experiments to model a scenario where the learner overestimates or underestimates the teacher's pedagogical selectivity.

6.4.1 Experiment Setting and Evaluation Metrics

Each simulation experiment modeled a virtual learner and a teacher. The teacher was assumed to have access to the learner's belief over task specification at any given time and the belief update model, thus allowing it to predict what the updated belief would be conditioned on the input provided by the teacher.

Each simulation experiment was assigned a set of conditions that had to be run for a pre-determined number of replicates. In each of the replicates, a ground truth formula was sampled from the prior over LTL formulas, $P(\varphi)$. Additionally, a set of two (2) execution traces were generated, $\mathcal{D} = \{\langle [\boldsymbol{\alpha}]_1, 1\rangle, \langle [\boldsymbol{\alpha}]_2, 1\rangle\}$, that satisfied the ground truth formula. This was then used to initialize the task-specific prior $P(\varphi)_0 = P(\varphi \mid \mathcal{D}; P(\varphi))$ that was then used for all the online learning conditions for that replicate.

An online learning condition was defined by the learning modality schedule, the pedagogical selectivity of the teacher, the learner's estimate of the teacher's pedagogical selectivity, the criteria guiding the query selection strategy, and the modality selection. We considered one of three learning modality schedules: for the fixed modality schedule, we either used only active queries or only teacher-provided demonstrations, or we used an adaptive modality schedule as defined in Section 6.2.1. For each condition, we maintained a record of the belief distribution after each iteration; and the task execution trace along with

the label assigned to it.

To evaluate the learning performance, we used the entropy of the belief distribution to represent the (un)certainty of the learner and the similarity metric to the ground truth as a measure of the accuracy of the learned specification. Given two formulas, φ_1 and φ_2, that are conjunctive compositions of the clauses in sets $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ respectively, the similarity metric is defined using an intersection-over-union:

$$L(\varphi_1, \varphi_2) = \frac{|\boldsymbol{C}_1 \cap \boldsymbol{C}_2|}{|\boldsymbol{C}_1 \cup \boldsymbol{C}_2|}. \tag{6.13}$$

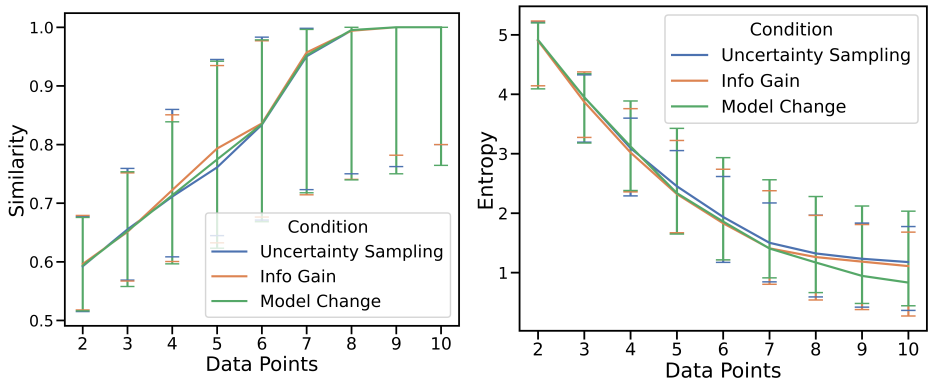
The similarity of a belief distribution $P(\varphi)_i$ with a ground truth formula φ^* is:

$$L(P(\varphi)_i) = \mathbb{E}_{P(\varphi)_i)}[\![L(\varphi, \varphi^*)]\!]. \tag{6.14}$$

6.4.2 Comparison of Query Strategies for Active Learning

In the first experiment, we compared the relative performance of a fixed learning modality schedule that only relies on assessments of queries generated by the learner. We utilized the three query selection strategies described in Section 6.2.1. We first compare the relative performance of each query selection strategy by comparing the similarity with the ground truth specifications as additional trajectories are provided to the learner. The learning curves comparing the similarity of the posterior distribution to the ground-truth value are depicted in Figure 6-3a, and the entropy of the posterior distribution is depicted in Figure 6-3b. Note that the variability within each query selection strategy (measured by the interquartile range) is much larger than the variability between the median performance. Further, this indicates that an active learning strategy can accurately infer the ground truth task specification as the learner is allowed to generate a greater number of query executions whose acceptability is labeled by the teacher.

As each replicate run used the same underlying ground truth formula and initial demonstrations, we can directly compare the best and the worst performing condition. Table 6.1 records when the similarity of each condition was equal to the best performance for a given experimental run. Note that each condition was the best and the worst-performing



(a) (b)

Figure 6-3: Statistics from the simulation experiments comparing a fixed learning modality schedule with only active queries for three different query selection strategies. Figure 6-3a depicts the median and the interquartile range of the similarity of the inferred distribution with respect to the ground truth formula as a box plot. Figure 6-3b depicts the same for the entropy of the inferred distribution. The underlying samples for these plots were generated through 500 simulated runs with a shared ground truth formula across all conditions.

	Uncertain Sampling	Information Gain	Model Change
Best performing	313	313	318
Worst performing	133	126	138

Table 6.1: Frequency as the best and the worst performing active learning query selection strategy over the course of 500 simulated experiment runs.

for a similar number of experimental runs. These results indicate that all the query selection strategies perform comparably.

Next, we investigate how the query selection strategies differ qualitatively. Recall from Section 6.2.1, that each query selection strategy determines a state $\langle \varphi' \rangle^*$ of the reward machine that would be most informative towards refining the learner's belief $P(\varphi)$. To determine if the similar performance results from the query selection strategies choosing the same state, we identify the number of times the choice of the evaluation criteria would have resulted in a different reward machine state. Thus, we identified the state based on each of the query selection strategies for each experimental run at each iteration. With 500 simulated runs and nine distributions per condition for three conditions, this resulted in 27,000 scenarios. Of these, we ignored the distributions that had already effectively converged to a single formula. The results are tabulated in Table 6.2. Note that while

	Uncertain Sampling	Information Gain	Model Change
Uncertainty Sampling	0	1181	242
Information Gain		0	1108
Model Change			0

Table 6.2: Mismatches between state selection based on the different query selection strategies

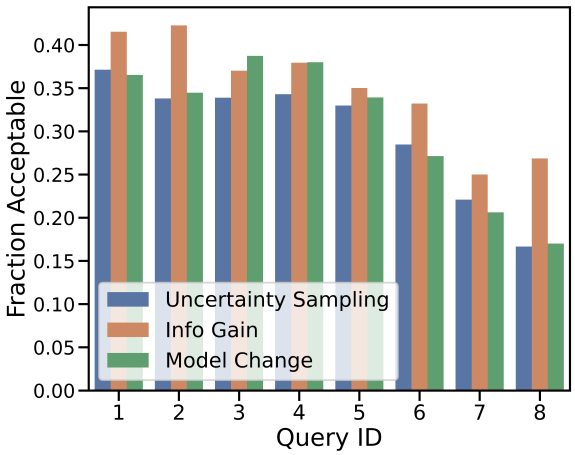


Figure 6-4: The fraction of acceptable query executions generated by each of the query selection strategies as a function of the number queries asked to the teacher.

different, *Uncertainty Sampling* and *Model Change* selection strategies are better aligned with each other compared to *Information Gain*.

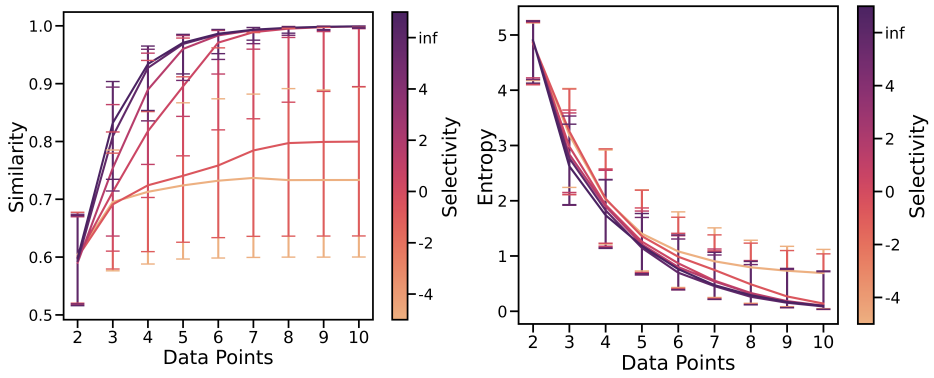
Finally, we evaluate whether the mismatches result in qualitatively different query executions. For this, we record the number of acceptable and unacceptable task executions generated as queries by the learner. This is indicative of whether a given query selection strategy prefers confirming queries or falsifying queries. The fraction of acceptable queries generated by each query selection strategy is depicted in Figure 6-4. We note that towards the end of the learning curve, the *Information Gain* query selection strategy selects states that result in a greater fraction of confirming demonstrations as compared to falsifying demonstrations preferred by both *Uncertainty Sampling* and *Model Change* strategies. This indicates that the *Information Gain* query selection strategy might be susceptible to confirmation biases when faced with low entropy belief distributions.

The similar quantitative performance with a diverging qualitative behavior indicates

that additional experiments are required to characterize the nature of the query executions selected by each strategy and its impact on the asymptotic learning performance. Further characterizing the confirming and falsifying nature of actively selected queries in other active learning paradigms for robot learning is also an interesting area of future investigations.

6.4.3 Effect of Pedagogical Demonstrations

In the second experiment, we investigate the effect of providing pedagogical examples on the learner's learning curve. This experiment was also run with 500 randomly sampled ground truth formulas and shared initial demonstrations between all the conditions. Each condition utilized a pre-determined learning modality schedule; each had a different value of the demonstrator's pedagogical selectivity γ (Eq 6.12). We varied the selectivity from generating demonstrations anchored to the initial demonstrations ($\gamma < 0$), all the way to a perfectly optimal demonstrator ($\gamma \to \infty$). The learning curves for the similarity and the entropy of the belief distribution at each iteration are depicted in Figure 6-5.



(a) (b)

Figure 6-5: Statistics from the simulation experiments comparing a fixed learning modality schedule with only demonstrations with varying levels of demonstrator pedagogical selectivity. Figure 6-5a depicts the median and the interquartile range of the similarity of the inferred distribution with respect to the ground truth formula as a box plot. Figure 6-5b depicts the same for the entropy of the inferred distribution. The underlying samples for these plots were generated through 500 simulated runs with a shared ground truth formula across all conditions. Note the poor learning performance when the teacher is not pedagogical ($\gamma < 0$).

The similarity learning curve (Figure 6-5a) indicates that as the pedagogical selectivity

of the demonstrator increases, the learner's belief converges to the true distribution more rapidly. Further, the smaller interquartile range suggests that pedagogical demonstrators can teach a wide variety of ground truth specifications reliably. A final aspect to note is that the median similarity values are lower than 1, the maximum possible value of the similarity metric. This is explained by the fact that the formulas that encode an underconstrained version of the ground-truth task specification cannot be fully eliminated from the belief in the absence of falsifying demonstrations. The likelihood function that conforms to the size principle, discussed in Chapter 4, increases the probability of the more constrained formulas with more confirming examples, but falsifying demonstrations generated by active queries can achieve a better convergence to the ground truth specification.

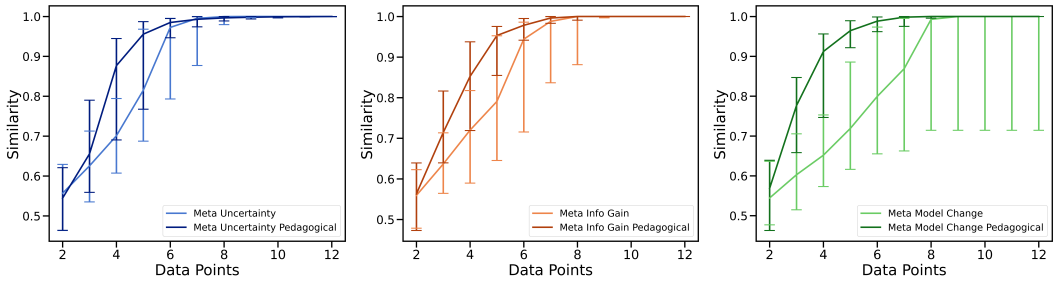
The learning curve of the entropy of the belief distribution at each iteration (Figure 6-5b) shows that allowing the learner to observe a greater number of demonstrations reduces the uncertainty of the belief as indicated by the reducing entropy for any value of the pedagogical selectivity. Specifically, the rapidly reducing uncertainty for an anchored demonstrator indicates the issue with confirmation biases for an inductive learner. Here the learner is very certain about an incorrect (overconstrained) task specification.

6.4.4 Comparison of Query Selection Strategies with Meta-Choice

In the third simulation experiment, we compare the effect of using different utility functions described in Section 6.2.4. Here The learner first used the utility functions to assert a meta-choice, i.e., the intelligent selection of the learning modality for the next iteration; then, if the modality was acceptability assessment of the query execution, the learner used the same utility function to determine the ideal query execution as described in Section 6.2.3. We tested each utility function with two different models for the teacher's pedagogical selectivity, first where the teacher is optimally pedagogical ($\gamma \to \infty$), and second where the teacher generates independent demonstrations $\gamma = 0$.

The similarity learning curves for each of the utility functions are depicted in Figure 6-6. We first note that each utility function except *uncertainty sampling* with an optimally pedagogical teacher demonstrates convergence to the ground-truth specification in greater than half the trials. Next, we note that the median value for uncertainty sampling with an

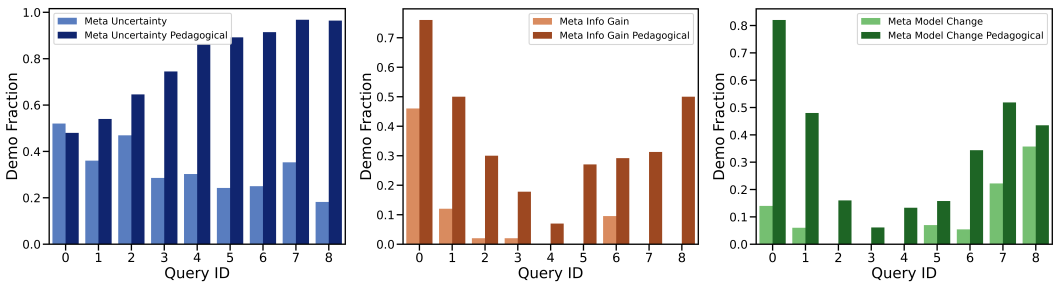
optimally pedagogical teacher is just less than 1, the maximum possible similarity score. Thus the learner's belief places a high probability mass on the ground-truth formula but still assigns a non-zero probability to competing candidate formulas. We also note that the *model change* utility function demonstrates the largest deterioration in performance between a pedagogical teacher and a non-pedagogical teacher. In contrast, *uncertainty sampling* demonstrates the smallest performance deterioration. Finally, we note that using *model change* as the utility function results in the fastest initial learning curve with an optimally pedagogical teacher.



(a) Uncertainty Sampling (b) Information Gain (c) Model Change

Figure 6-6: Similarity learning curves for the different utility functions. Note that each curve depicts the median and the error bars represent the inter-quartile range.

In order to explain the differences in the observed performance, we next examine the meta-choices made by the learning agent. Figure 6-7 depicts the fraction of times the learner chose to elicit a new demonstration from the teacher as a function of the iteration count. We note that when learning from an optimally pedagogical teacher, the learner universally elicits demonstrations more frequently than learning from a non-pedagogical teacher. Further, we note that with a pedagogical teacher, the choice of *information gain* or *model change* as the utility function results in the learner seeking more demonstrations in the early phase of learning as compared to the case where *uncertainty sampling* is used as the utility function. We note that this is also correlated with a steeper initial learning curve for *model change* as *information gain* utility functions as compared to *uncertainty sampling*. Finally, we note that *uncertainty sampling* with a pedagogical teacher resorts to a greater reliance on demonstrations towards the latter phase of learning. Thus the learning curve with *uncertainty sampling* resembles a learner that learns purely from demonstrations (Figure 6-5a).



(a) Uncertainty Sampling (b) Information Gain (c) Model Change

Figure 6-7: Fraction of trials with demonstrations requested for different utility functions.

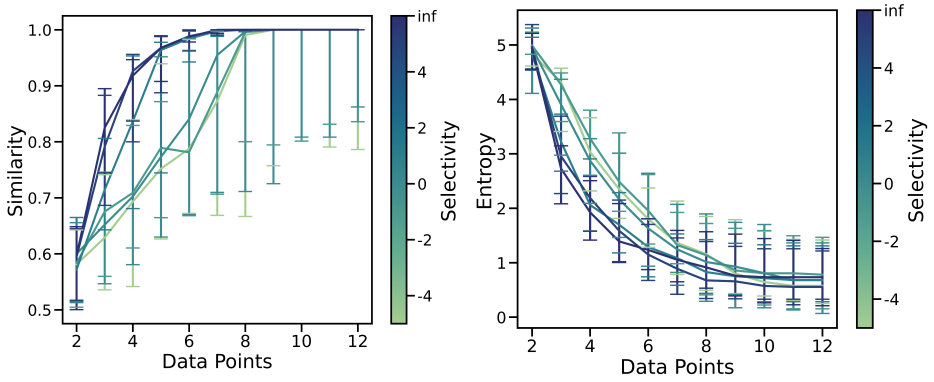
These experiments support the idea that pedagogical demonstrations are most useful in the early learning phase to quickly collect a representative sample of acceptable task executions. In contrast, actively generated queries are most useful to refine the specifications by eliciting acceptability labels for executions that falsify a targeted subset of the support of the belief distribution. These experiments also support the idea that while acceptable demonstrations allow for faster learning initially, falsifying observations are necessary to converge to the ground-truth task specification.

6.4.5 Effect of Pedagogical Selectivity with Meta-Choice

In the final set of simulation experiments, we examined the impact of noisy pedagogical selectivity when the learner could assert a meta-choice. For this set of experiments, we selected *model change* as the utility function to guide both query selection and the meta-choice. Just as in the experiment on learning only from demonstrations described in Section 6.4.3, we varied the pedagogical selectivity to represent a teacher generating anchored demonstrations $\gamma < 0$ to a teacher providing the optimal demonstrations $\gamma \to \infty$.

The learning curves are depicted in Figure 6-8, just as in the case of learning purely from demonstrations, decreasing the pedagogical selectivity results in deterioration of learning performance. However, we note the contrast with Figure 6-5a–where the learning curves converge to a value lower than 1–the ability to select the learning modality at each iteration allows the learner to infer the correct specification in a majority of trials even from a teacher providing anchored demonstrations ($\gamma < 0$).

To examine the cause for this, we also plot the fraction of trials in which the learner



(a) Similarity (b) Entropy

Figure 6-8: Learning curves for intelligent modality selection with varying pedagogical selectivity of the teacher. The plots represent the mean and the errorbars represent the inter-quartile range from 500 runs.

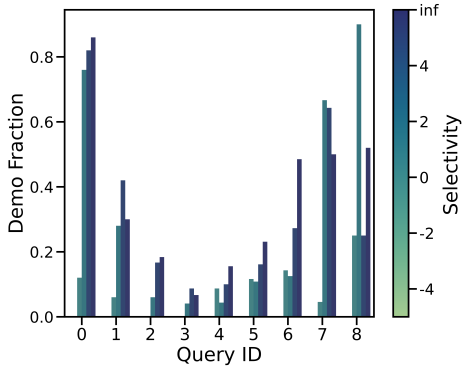


Figure 6-9: Fraction of trials with demonstration requested with varying pedagogical selectivity.

asked the teacher to provide a new demonstration at each iteration. These results are depicted in Figure 6-9. As expected, the fraction of trials where the learner requested a demonstration at each iteration increased as the pedagogical selectivity increased. We also note that the qualitative trend of how frequently the learner requested demonstration is similar for all values of pedagogical selectivity.

6.4.6 Key Findings

The key findings from the simulation experiments are:

- Active learning, i.e., learning from acceptability assessments of task executions performed by the learner, can infer the ground-truth specification accurately given a sufficient number of learning iterations.
- Learning purely from demonstration is sensitive to the pedagogical selectivity of the teacher. The learner can leverage the diversity of the demonstrations to learn very rapidly, even from a noisily pedagogical teacher; however, the learning performance suffers when the teacher's demonstrations are anchored ($\gamma < 0$).
- Allowing the learner to assert a meta-choice over the learning modality for the next iteration can potentially overcome the limitations of a teacher who provides anchored demonstrations.
- All the utility functions, namely *uncertainty sampling*, *information gain*, and *model change* show very similar performance across a wide range of ground-truth formulas. However we discovered that query executions entailed by *uncertainty sampling*, and *model change* utility function are better aligned with each other than those generated by *information gain*.
- When the learner is allowed to assert a meta-choice over the learning modality for the next iteration, *uncertainty sampling* favors seeking demonstrations in the latter phase of learning. In contrast, *model change* and *information gain* function favor demonstrations in the earlier phase of learning. We observe an impact of that on the slope of the learning curve and the asymptotic convergence of the learning protocols.

(a) Setup (b) Task 1 (c) Task 2

Figure 6-10: Experimental Setup and desired configurations for the table-setting user study.

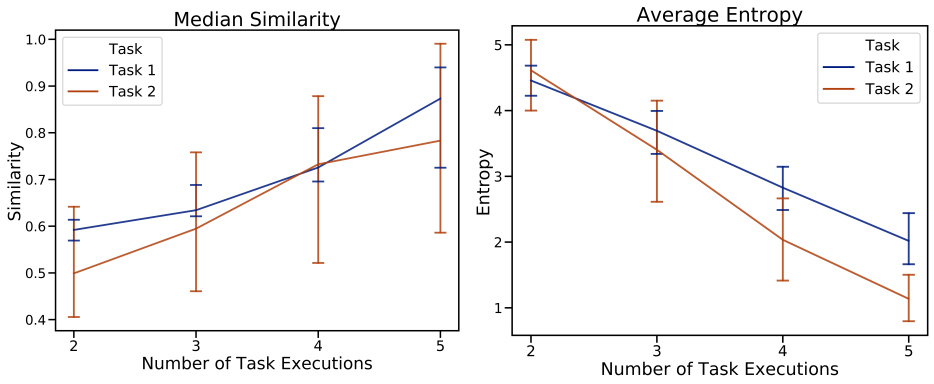
6.5 User Study

In order to evaluate the real-world performance of the active learning approach, we conducted a user study that involved participants teaching a robot to set a dinner table in various reference configurations.

Figure 6-10a depicts the experiment setup. During each task execution (whether demonstrated by the participant or performed by the robot), the five objects were initially placed on Table A, and subsequently arranged on Table B. In the first phase of the study, the task (*Task 1*) involved arranging the objects into the configuration depicted in Figure 6-10b, with demonstrations provided directly by participants. In the second phase, we modified the study to be conducted online due to the restrictions on in-person studies imposed in light of the COVID-19 pandemic, and participants remotely commanded the robot to provide the demonstrations. The robot's task execution and the experiment instructions were displayed to the participants via video conferencing. In addition to *Task 1*, we also added a second task (*Task 2*), wherein the dinner plate and bowl were to be placed on the table in the configuration depicted in Figure 6-10c. Placing the fork and knife were optional, and placing the small plate was not permitted.[1]

We instructed participants to move only a single object at a time while providing demonstrations, and informed them that objects could not be picked up again once placed on Table B. Participants were also instructed to provide an assessment after observing the robot while it executed the task; a participant's label was only recorded once the entire task had been completed. For both the in-person and remote study protocols, the participant

[1] example video: `ral2021.ajshah.info`



(a) Similarity (b) Entropy

Figure 6-11: Results from active learning user study.

initiated the robot's belief with two demonstrations; beliefs were then refined using three queries generated via our active learning models. Finally, the robot demonstrated the results of its learning by performing three task executions observed by the participant.

The state space of the robot, $\mathcal{X}$, was identical to the set of propositions required for evaluating the task, $\boldsymbol{\alpha}$, and contained five Boolean propositions, each of which encoded whether a particular object was successfully placed on the table. The robot's action space, $\mathcal{A}$, comprised five actions (one for each object). Initiating an action triggered a sequence of parameterized primitives programmed into the robot to locate, pick up, and place the object on Table B. Based on the constraints provided to the participants and the robot's action space, the only way to successfully complete *Task 1* was to ensure that the dinner plate, small plate, and bowl were placed in that specific order (the fork and the knife could be placed at any instant). There were multiple final acceptable configurations for *Task 2*; in each, the dinner plate and bowl were placed in that partial order, while the fork and the knife may or may not have been placed.

6.5.1 Results

We recruited 18 participants for the in-person phase of the study, but had to terminate the protocol with three participants due to robot hardware failure. The results include data collected from 15 participants (10 male, 5 female, median age: 26 years), seven of whom reported prior experience with robots or automated systems. All participants were instructed

to teach the robot to perform *Task 1*. For the remote phase, we recruited 12 participants (8 male, 4 female, median age: 28 years); four participants reported prior experience with robotics. We assigned six participants each to *Task 1* and *Task 2*.

All participants were successfully able to teach the assigned task to the robot–i.e., the policies learned by the robot did not result in an incorrect table setting during any of the test executions. The learning curves for the robot are depicted in Figure 6-11.

Overall, the median similarity of the final belief with respect to the ground truth formula was 0.83 95% $CI: [0.73, 0.94]$. The median similarity was 0.87 95% $CI: [0.73, 0.94]$ for Task 1; for Task 2, it was 0.78 95% $CI: [0.59, 0.99]$. For *Task 1*, the posterior belief distribution recovered the ground truth formula as the most likely LTL specification for 14 out of 21 participants, while the most likely specification differed from the ground truth by a single conjunctive clause in four cases. Similarly, for *Task 2*, the posterior belief distribution for two out of six participants recovered the ground truth formula as the most likely LTL specification, while the most likely specification for two of them differed from the ground truth formula by a single conjunctive clause. Our demonstration of the entire learning pipeline on an embodied robot indicates the viability of deploying our active learning framework for real-world applications.

6.6 Summary and Future Directions

In this chapter, we developed a Bayesian framework that integrates specification inference and planning to allow the learner to actively refine its belief over the teacher's intended task. We developed decision-making models that allow the learner to first assert a meta-choice over its next learning modality and determine the most informative task execution. We further conducted a battery of simulation experiments over a wide range of ground-truth task specifications to compare different learning modality schedules, utility functions for active learning, and assessing the impact of the teacher's pedagogical selectivity. In particular, we demonstrated that an active learner could overcome the limitation of a teacher in providing demonstrations that are not adequately diverse. Finally, we demonstrated our Bayesian framework on the real-world task of setting a dinner table, where 18 participants successfully

taught the robot to set the dinner table in the desired configuration.

Our work has demonstrated the viability of active learning approaches in learning non-Markov task specifications; however, it also opens up an exciting set of questions involved in learning from human teachers. A key question for asserting the meta-choice is the estimate of the teacher's pedagogical selectivity. The variance of selectivity across humans and between different task or task environments for the same person is unknown. Prior research has indicated that while the teacher's pedagogical selectivity can be leveraged [23, 80], it is also challenging to predict [81]. Further studies into human-robot pedagogy are necessary to develop best practices for deploying our framework.

This work models programming the robot as manipulating the belief distribution over tasks through interactions with the teacher. In our work, we have considered demonstrations and acceptability assessments as modalities for interacting with the robot; many intuitive modalities can be used. Developing computational models for grounding interactions through various intuitive modalities into a belief over LTL formulas is an interesting area of future research.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Conclusion

This dissertation lays the groundwork for a robotic apprentice that is an active learner. However, an ideal robotic apprentice should improve continually after every task execution or interaction with a human expert, recognize novel avenues for improvement, and succinctly explain its understanding of the task to a human expert. This will require many future algorithmic and empirical developments. Here, we highlight a few promising areas for future research.

1. **Aligning Mental Models of Abstraction**:
   Robots that develop hierarchical abstract task representations can later compose them through simple rules to perform increasingly complex tasks. The ability to develop useful abstractions is key to deploying robots in unseen, partially defined domains. However, to ensure that the robots can learn from human teachers in these domains, it is key that the robot's abstractions are aligned with those used by human domain experts.

   Cognitive science research has a vast body of work characterizing human cognitive biases and learning heuristics [130, 117]. Developing learning models for robots that leverage human cognitive biases to achieve data efficiency is a promising avenue for future work. Our work on learning temporal specifications from demonstrations is an example of successfully leveraging temporal abstractions along with cognitively inspired Bayesian learning. Leveraging tools such as natural language instructions

[88, 51], augmented reality [85] are other examples of leveraging human-interpretable abstractions to facilitate robot learning through expressive human feedback.

Closely related to this is the challenge of allowing a robot to accurately and compactly summarizes its own representation for the user. More concretely, the robot should be capable of answering questions like: '*when do you consider the bowl to be placed correctly?*' in the table-setting context. Our initial approach resulted in Bayes-TrEx [20], a sampling-based approach to model transparency by example. Zhou and Booth et al. [142] have also extended Bayes-TrEx to analyze robot behavior through the development of RoCUS, a sampling-based approach to discover environments in which motion planners generate trajectories that exhibit a pre-defined behavior.

2. **Algorithms for Pedagogical Learning and Teaching**:

   Empowering task experts to deploy automation by programming the task rather than the robot necessitates developing a diverse toolkit to train robotic apprentices. We can only assess the utility of these approaches through well-designed user studies informed by prior research into human-factors in addition to the technical contributions of this dissertation. Future algorithmic developments towards sophisticated human-robot interaction models will also open up new challenges in human factors research.

   Our recent experiments on interactive robot training [113] have indicated the influence of ground truth specifications, pedagogical selectivity of the teacher, and the specific training modality on the relative performance different training protocols. Prior research into cognitive models for representative sampling and pedagogical behavior [124, 107, 108, 109] have indicated the existence of different behaviors while teaching a task compared to simple performing it. Modeling these pedagogical cognitive priors remains an open problem. Appropriate pedagogical priors will inform the design of cognitive support systems for domain experts to better shape their teaching strategy and lead to algorithms that learn faster from human teachers. Subsequently, robots can also leverage these pedagogical priors to better explain their policies to human teammates.

3. **Dynamic Multi-Agent Environments**:

   The ability to learn novel tasks through intuitive modalities, jointly construct useful abstractions along with a human teammate, and then convey novel information back to the human teammate are critical enabling technologies for agile robots of the future. However, many real-world environments are dynamic, including multiple decision-making actors, and have pre-defined regulations for operations, such as public roads or airspace systems worldwide. There are two key challenges in dynamic environments: first, the expert demonstrator cannot always guarantee the correct execution of tasks due to the non-deterministic nature of the environment and the presence of other decision-making agents; second, the changes in the task state may not be causally linked to the actions of the robot. The ability to rapidly learn and execute novel tasks in multi-agent dynamic environments will be crucial to widespread robot deployment and adoption.

7.1 Summary of Contributions

In this dissertation, we developed a novel Bayesian framework for learning complex non-Markov tasks through interactions with a human teacher through intuitive modalities. We demonstrated the utility of this framework in the automated evaluation of task executions, decision-making under epistemic uncertainty concerning the task specification, and active learning setting for rapidly learning new tasks through interactions. The contributions of this dissertation are:

1. We developed a Bayesian model for inferring template-based LTL specifications from labeled task executions. This involved developing structured priors and approximate likelihood models to perform sampling-based inference over a hypothesis space of LTL formulas. Our approach can infer LTL specifications inductively, i.e., from solely positive examples and from both positive and negative examples.

2. We proposed planning with uncertain specifications (PUnS), a novel formulation that admits a belief distribution over candidate task specifications, and yields a policy

that attempts to satisfy a maximal subset of the belief distribution simultaneously. We formalize this by proposing four evaluation criteria that define the semantics of satisfying a belief over logical formulas. We further demonstrate the trade-off between the flexibility of task execution and risk-aversion.

3. We demonstrated that there exists an equivalent MDP encoding of a PUnS problem. We propose computational models to compile any PUnS problem into an equivalent MDP by first constructing a reward machine that captures the structure of the belief distribution and then concatenating it with the environment MDP.

4. We propose an interactive online Bayesian framework that combines specification inference and planning to enable an active learning agent to refine its beliefs by selecting the best learning modality and identify an informative task execution to learn from.

5. We demonstrate that allowing the learner to assert a meta-choice over the learning modality allows it to overcome the limitation of a teacher in providing diverse and representative training demonstrations.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[2] S. Reza Ahmadzadeh, Muhammad Asif Rana, and Sonia Chernova. Generalized cylinders for learning, reproduction, generalization, and refinement of robot skills. In *Proceedings of Robotics: Science and Systems*, 2017.

[3] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016.

[4] David J. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII*, 1983.

[5] Brandon Araki, Kiran Vodrahalli, Thomas Leech, Cristian-Ioan Vasile, Mark Donahue, and Daniela Rus. Deep Bayesian nonparametric learning of rules and plans from demonstrations with a learned automaton prior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

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

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

[8] Christopher G. Atkeson, P. W. Babu Benzun, Nandan Banerjee, Dmitry Berenson, Christoper P. Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin, M. Gennert, Joshua P. Graff, Peng He, Aaron Jaeger, Joohyung Kim, Kevin Knoedler, Lening Li, Chenggang Liu, Xianchao Long, T. Padir, Felipe Polido, G. G. Tighe, and X. Xinjilefu. What happened at the DARPA robotics challenge finals. In Matthew Spenko, Stephen Buerger, and Karl Iagnemma, editors, *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, pages 667–684. Springer International Publishing, Cham, 2018.

[9] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1160–1167, 1996.

[10] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial intelligence*, 116(1-2):123–191, 2000.

[11] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning robot objectives from physical human interaction. In *Proceedings of the Conference on Robot Learning*, pages 217–226. PMLR, 2017.

[12] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. Learning from physical human corrections, one feature at a time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, page 141–149. ACM, 2018.

[13] Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[14] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

[15] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

[16] Aude Billard, Sina Mirrazavi, and Nadia Figueroa. *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. MIT Press, 2022.

[17] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *arXiv preprint arXiv:2006.14091*, 2020.

[18] Erdem Biyik, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Proceedings of the Conference on Robot Learning*, volume 100, pages 1177–1190. PMLR, 30 Oct–01 Nov 2020.

[19] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Proceedings of the Conference on Robot Learning*, pages 519–528, 2018.

[20] Serena Booth, Yilun Zhou, Ankit Shah, and Julie Shah. Bayes-TrEx: a Bayesian sampling approach to model transparency by example. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11423–11432, 2021.

[21] Cynthia Breazeal, Andrew Brooks, Jesse Gray, Guy Hoffman, Cory Kidd, Hans Lee, Jeff Lieberman, Andrea Lockerd, and David Chilongo. Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robotics*, 1(02):315–348, 2004.

[22] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the 36th International Conference on Machine Learning*, pages 783–792. PMLR, 2019.

[23] Daniel S Brown and Scott Niekum. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7749–7758, 2019.

[24] Maya Cakmak, Crystal Chao, and Andrea L Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118, 2010.

[25] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[26] Alberto Camacho, R Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6065–6073, 2019.

[27] Alberto Camacho and Sheila A. McIlraith. Learning interpretable models in linear temporal logic. In *Proceedings of the Twenty-Nineth International Conference on Automated Planning and Scheduling*, pages 621–630, 2019.

[28] Alberto Camacho, Eleni Triantafillou, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-deterministic planning with temporally extended goals: Completing the story for finite and infinite LTL. In *Proceedings of the 8th Workshop on Heuristics and Search for Domain-independent Planning*, pages 68–76, 2016.

[29] Ivana Černá and Radek Pelánek. Relating hierarchy of temporal properties to model checking. In *International Symposium on Mathematical Foundations of Computer Science*, pages 318–327. Springer, 2003.

[30] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. *Proceedings of Conference on Robot Learning*, 2020.

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

[32] Daniil Chivilikhin, Ilya Ivanov, and Anatoly Shalyto. 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, pages 1185–1188, New York, NY, USA, 2015. ACM.

[33] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[34] Patrick Craven, Kevin Oden, Kevin Landers, Ankit Shah, and Julie Shah. Man-machine interoperation in training for offensive counter air missions. In *Interservice/Industry Training, Simulation and Education Conference*, 2018.

[35] Patrick Craven, Kevin Oden, Kevin Landers, Ankit Shah, and Julie Shah. Man-machine interoperation in training for large force exercise air missions. In *Interservice/Industry Training, Simulation and Education Conference*, 2019.

[36] Yuchen Cui and Scott Niekum. Active reward learning from critiques. In *2018 IEEE International Conference on Robotics and Automation*, pages 6907–6914. IEEE, 2018.

[37] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 221–236. ACM, 2019.

[38] Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[39] Matthew B Dwyer, George S Avrunin, and James C Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, pages 411–420. ACM, 1999.

[40] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[41] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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

[43] E Allen Emerson. Temporal and modal logic. In *Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[44] Nadia Figueroa and Aude Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In *Proceedings of the Conference on Robot Learning*, pages 927–946, 2018.

[45] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[46] Cameron E. Freer, Daniel M. Roy, and Joshua B. Tenenbaum. *Towards common-sense reasoning via conditional simulation: Legacies of Turing in Artificial Intelligence*, page 195–252. Lecture Notes in Logic. Cambridge University Press, 2014.

[47] Mark Gabel and Zhendong Su. Symbolic mining of temporal specifications. In *Proceedings of the 30th International Conference on Software Engineering*, pages 51–60. ACM, 2008.

[48] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David Smith, Ying Sun, and Daniel Weld. Pddl - the planning domain definition language. 08 1998.

[49] Noah D. Goodman, Vikash K. Mansinghka, Daniel Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: A language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, page 220–229. AUAI Press, 2008.

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

[51] Nakul Gopalan, Dilip Arumugam, Lawson Wong, and Stefanie Tellex. Sequence-to-Sequence Language Grounding of Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, 2018.

[52] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005.

[53] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, pages 6765–6774, 2017.

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

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

[56] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 2112–2121, 2018.

[57] Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4415–4426, 2020.

[58] Susmit Jha and Sanjit A Seshia. A theory of formal synthesis via inductive learning. *Acta Informatica*, 54(7):693–726, 2017.

[59] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. Team IHMC's lessons learned from the DARPA robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[60] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[61] Sertac Karaman and Emilio Frazzoli. Linear temporal logic vehicle routing with applications to multi-uav mission planning. *International Journal of Robust and Nonlinear Control*, 21(12):1372–1395, 2011.

[62] Daniel Kasenberg and Matthias Scheutz. Interpretable apprenticeship learning with temporal logic specifications. In *IEEE Conference on Decision and Control*, 2017.

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

[64] Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[65] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction*, pages 259–266. IEEE, 2010.

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

[67] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 273–282. ACM, 2014.

[68] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

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

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

[71] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):211–236, 2018.

[72] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[73] John E Laird, Kevin Gluck, John Anderson, Kenneth D Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, et al. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

[74] Pierre Simon Laplace. A philosophical essay on probabilities, 1951.

[75] Caroline Lemieux, Dennis Park, and Ivan Beschastnikh. General LTL specification mining. In *30th IEEE/ACM International Conference on Automated Software Engineering*, pages 81–92. IEEE, 2015.

[76] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

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

[78] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties (invited paper, 1989). In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, page 377–410. ACM, 1990.

[79] Bernard Michini and Jonathan P. How. Bayesian nonparametric inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer Berlin Heidelberg, 2012.

[80] Smitha Milli, Pieter Abbeel, and Igor Mordatch. Interpretable and pedagogical examples. *arXiv preprint arXiv:1711.00694*, 2017.

[81] Smitha Milli and Anca D. Dragan. Literal or pedagogic human? analyzing human model misspecification in objective learning. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115, pages 925–934. PMLR, 22–25 Jul 2020.

[82] Smitha Milli, Dylan Hadfield-Menell, Anca Dragan, and Stuart Russell. Should robots be obedient? In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 4754–4760, 2017.

[83] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[84] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[85] Carl Mueller, Jeff Venicx, and Bradley Hayes. Robust robot learning from demonstration and skill repair using conceptual constraints. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6029–6036. IEEE, 2018.

[86] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, 2000.

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

[88] Yoonseon Oh, Roma Patel, Thao Nguyen, Baichuan Huang, Ellie Pavlick, and Stefanie Tellex. Planning with state abstractions for non-Markovian task specifications. In *Proceedings of Robotics: Science and Systems*, June 2019.

[89] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016.

[90] Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

[91] Claudia Pérez-D'Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *2017 IEEE International Conference on Robotics and Automation*, pages 4058–4065. IEEE, 2017.

[92] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

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

[94] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *2014 IEEE 53rd Annual Conference on Decision and Control*, pages 81–87. IEEE, 2014.

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

[96] Preeti Ramaraj, Matt Klenk, and Shiwali Mohan. Understanding intentions in human teaching to design interactive task learning robots. In *RSS 2020 Workshop: AI & Its Alternatives in Assistive & Collaborative Robotics: Decoding Intent*, 2020.

[97] M Asif Rana, Mustafa Mukadam, S Reza Ahmadzadeh, Sonia Chernova, and Byron Boots. Learning generalizable robot skills from demonstrations in cluttered environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4655–4660. IEEE, 2018.

[98] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 471–477, 2015.

[99] E Ratner, D Hadfield-Mennell, and A Dragan. Simplifying reward design through divide-and-conquer. In *Proceedings of Robotics: Science and Systems*, 2018.

[100] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:297–330, 2020.

[101] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[102] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems*, 2017.

[103] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Proceedings of Robotics: Science and Systems*, 2016.

[104] Lindsay Sanneman, Christopher Fourie, and Julie A. Shah. The state of industrial robotics: Emerging technologies, challenges, and key research directions. *Foundations and Trends® in Robotics*, 8(3):225–306, 2021.

[105] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[106] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[107] Patrick Shafto and Noah Goodman. Teaching games: Statistical sampling assumptions for learning in pedagogical situations. In *Proceedings of the 30th annual conference of the Cognitive Science Society*, pages 1632–1637. Cognitive Science Society, 2008.

[108] Patrick Shafto, Noah D Goodman, and Michael C Frank. Learning from others: The consequences of psychological reasoning for human learning. *Perspectives on Psychological Science*, 7(4):341–351, 2012.

[109] Patrick Shafto, Noah D Goodman, and Thomas L Griffiths. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology*, 71:55–89, 2014.

[110] Ankit Shah, Pritish Kamath, Shen Li, Patrick Craven, Kevin Landers, Kevin Oden, and Julie Shah. Supervised Bayesian specification inference from demonstrations. *arXiv preprint arXiv:2107.02912*, 2021.

[111] Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31*, pages 3804–3813. 2018.

[112] Ankit Shah, Shen Li, and Julie Shah. Planning with uncertain specifications (PUnS). *IEEE Robotics and Automation Letters*, 2020.

[113] Ankit Shah, Samir Wadhwania, and Julie Shah. Interactive robot training for non-Markov tasks. *arXiv preprint arXiv:2003.02232*, 2020.

[114] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[115] RN Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[116] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[117] Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[118] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606. Cognitive Science Society, 2009.

[119] A Prasad Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–511, 1994.

[120] David Smith. Planning as an iterative process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[121] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*, pages 3483–3491. 2015.

[122] R. Tedrake, M. Fallon, S. Karumanchi, S. Kuindersma, M. Antone, T. Schneider, T. Howard, M. Walter, H. Dai, R. Deits, M. Fleder, D. Fourie, R. Hammoud, S. Hemachandra, P. Ilardi, C. Pérez-D'Arpino, S. Pillai, A. Valenzuela, C. Cantu, C. Dolan, I. Evans, S. Jorgensen, J. Kristeller, J. A. Shah, K. Iagnemma, and S. Teller. A summary of team MIT's approach to the virtual robotics challenge. In *2014 IEEE International Conference on Robotics and Automation*, pages 2087–2087, 2014.

[123] Joshua B Tenenbaum. Rules and similarity in concept learning. In *Advances in Neural Information Processing Systems*, pages 59–65, 2000.

[124] Joshua B Tenenbaum, Thomas L Griffiths, et al. The rational basis of representativeness. In *Proceedings of the 23rd annual conference of the Cognitive Science Society*, pages 1036–1041, 2001.

[125] Joshua Brett Tenenbaum. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

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

[127] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 452–461. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[128] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *arXiv e-prints*, pages arXiv–2010, 2020.

[129] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in Neural Information Processing Systems*, 32:15523–15534, 2019.

[130] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[131] Vaibhav V Unhelkar and Julie A Shah. Learning models of sequential decision-making with partial specification of agent behavior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2522–2530, 2019.

[132] Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency*, pages 238–266. Springer, 1996.

[133] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K Ho, and Sanjit Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems*, pages 5372–5382, 2018.

[134] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K Ho, and Sanjit Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31*, pages 5368–5378. 2018.

[135] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[136] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[137] Victoria Xia, Zi Wang, Kelsey Allen, Tom Silver, and Leslie Pack Kaelbling. Learning sparse relational transition models. In *International Conference on Learning Representations*, 2018.

[138] Zhe Xu, Yuxin Chen, and Ufuk Topcu. Adaptive teaching of temporal logic formulas to preference-based learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5061–5068, 2021.

[139] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 590–598, 2020.

[140] Zhe Xu, Melkior Ornik, A Agung Julius, and Ufuk Topcu. Information-guided temporal logic inference with prior knowledge. In *2019 American control conference*, pages 1891–1897. IEEE, 2019.

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

[142] Yilun Zhou, Serena Booth, Nadia Figueroa, and Julie Shah. Rocus: Robot controller understanding via sampling. *arXiv preprint arXiv:2012.13615*, 2020.

[143] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438, 2008.