Effective Information Sharing for Human-Robot Collaboration

by

Vaibhav V. Unhelkar

B.Tech., Indian Institute of Technology (2012)
M.Tech., Indian Institute of Technology (2012)
S.M., Massachusetts Institute of Technology (2015)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Autonomous Systems
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2020

Signature redacted

Author .....
Department of Aeronautics and Astronautics
November 20, 2019

Certified by ..... Signature redacted
Prof. Julie A. Shah
Thesis Supervisor

Certified by ..... Signature redacted
Prof. Manuela M. Veloso
Thesis Committee Member

Certified by ..... Signature redacted
Prof. Nicholas Roy
Thesis Committee Member

Accepted by ..... Signature redacted
Prof. Sertac Karaman
Associate Professor, Aeronautics and Astronautics
Chair, Graduate Program Committee

Effective Information Sharing for Human-Robot Collaboration

by

Vaibhav V. Unhelkar

Submitted to the Department of Aeronautics and Astronautics
on November 20, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Autonomous Systems

Abstract

Humans and machines often possess complementary skills. The recognition of this fact is leading to a steadily growing interest in collaborative robots. Despite the growing interest, however, a fundamental question remains to be answered: "How does one develop effective collaborative robots?"

Three entities need to be considered while answering this question — namely, the *collaborative robot* itself, the *human teammate* whom the robot interacts with, and, equally importantly, the *robot developer* who is tasked with designing the machine. Each of these entities possesses different information. Effective sharing of this information is essential for developing collaborative robots and achieving fluent collaboration. In this dissertation, I present models and algorithms to enable effective information sharing between the robot, the human, and the developer.

I begin by presenting the Agent Markov Model (AMM), a Bayesian model of sequential decision-making behavior, and Constrained Variational Inference (CVI), a hybrid learning algorithm that can learn generative models both from data and domain expertise. By utilizing AMM and CVI, the developer can specify decision-making models both for the human teammate and the collaborative robot with reduced labeling effort.

Next, I present ADACORL, a framework to generate the collaborative robot's policy for interaction. By leveraging algorithms for planning under uncertainty, ADACORL can generate fluent robot behavior for human-robot collaborative tasks with state spaces significantly larger than prior art (> 1 million states) and short planning times (< 1 s). Finally, I provide an approach for deciding if, when, and what to communicate during human-robot collaboration. Through human-robot interaction studies, I demonstrate that the proposed decision-making approaches result in the effective use of the robot's action and communication capabilities during collaboration with a human teammate.

Thesis Supervisor: Julie A. Shah
Title: Associate Professor, Massachusetts Institute of Technology

Thesis Committee Member: Manuela M. Veloso
Title: Herbert A. Simon University Professor, Carnegie Mellon University

Thesis Committee Member: Nicholas Roy
Title: R.L. Bisplinghoff Professor, Massachusetts Institute of Technology

Dissertation Reader: Christopher Amato
Title: Assistant Professor, Northeastern University

Dissertation Reader: Xi Jessie Yang
Title: Assistant Professor, University of Michigan

Acknowledgments

I convey my most sincere gratitude to Professor Julie Shah for her continued support, guidance, feedback, and insights. She introduced me to *seemingly simple but truly profound* concepts such as 'interaction, collaboration, and communication,' which have been fundamental to spawning this thesis research. I also thank Julie for championing my research efforts and her infectious positivity throughout the graduate experience. Simply put, I could not have asked for a better thesis advisor.

I express my deepest gratitude to Professors Manuela Veloso and Nicholas Roy. It was Manuela's work (on communication decision-making) that informed the formalisms that I use for modeling teamwork. In a similar vein, it was Nick's work (on Bayesian learning and POMDPs) that guided my choice of algorithmic techniques. I genuinely feel fortunate to have received their direct guidance and insightful suggestions, especially on questions that required a holistic perspective.

I am incredibly grateful to Professor Christopher Amato for his detailed feedback on my thesis proposal, dissertation, and research statements. I sincerely thank Professor Xi Jessie Yang for familiarizing me with human factors research. I must also thank all the members of my thesis defense committee – Julie, Nick, Manuela, Chris, and Jessie – for their continued mentorship, which has had a lasting impact on not only my research but also my professional development.

I sincerely thank Chongjie Zhang for being extremely generous with his time and advice during the formative years of my graduate experience. Many thanks to my collaborators at BMW, LMCO, ARL, and Fraunhofer IPA for supporting my research. Interactions with the industry collaborators have been pivotal for understanding the real-world challenges that this dissertation seeks to address. I particularly acknowledge Quirin Tyroller and Johannes Bix for their hospitality during the intercontinental collaboration visits and continued friendship.

Robotics is a team sport, especially while designing systems that can work with humans. This thesis research would not have been possible without the fantastic and fruitful collaborations with Shen Li and Pem Lasota. I will fondly remember

our programming sessions, time synchronization protocol, and adventurous travel visits. I also thank Ardavan Saeedi for insightful discussions on inference.

I will always remain indebted to my nurturing teachers at Kendriya Vidyalaya, IIT Bombay, and MIT. Especially, I thank Professors H. Arya, H. Hablani, P. Mujumdar, and K. Sudhakar for believing in me when I was an undergraduate and encouraging me to pursue graduate research. At MIT, I wish to acknowledge Professors Gregory Wornell and Leia Sterling for their fantastic courses on inference and statistics. Thanks should also go to the Pratham team at IIT Bombay and, particularly, Shashank Tamaskar for introducing me to research.

The most significant perk of being a graduate student is being able to interact with extraordinary colleagues and develop lifelong friendships. I thank all the members of the Interactive Robotics Group for the invaluable technical discussions, improving my communication skills, and appreciating my humor. In particular, I acknowledge the second/third generation of IRG – Abhi, Ankit, Claudia, Joe, Pem, and Ramya – with whom I had the great fortune of sharing the nonlinearities of graduate work and life. I also thank my friends, roommates, and, in particular, Mukund for making this journey all the more fun and memorable through numerous board game nights, badminton evenings, and dinners.

At MIT, I also had the great pleasure of mentoring, working with, and learning from exceptional undergraduate researchers. I thank Jorge, Hosea, Rares, Eghosa, Skylar, Thee, Kendall, Parker, Joe, and Michael for enriching my graduate experience. I also very much appreciate the technical and administrative support of Elizabeth Zotos, Beth Marois, Britton Bradley, Julie Finn, Ron Wiken, Michael Keohane, Mirabella Daguerre, MIT ISO, UROP office, and CSAIL TIG.

Last but not least, I extend my gratitude to my family: Aai, Papa, Payal, Amma, Amma, Raviraj, and Divya. Despite the extremely long distance and large time difference, I could always count on their unwavering support and encouragement. I would like to especially acknowledge my parents for their immeasurable effort in making my education possible. Finally, I thank my wife, Divya, for being a constant pillar of support and taking pride in my work.

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

Chapter 1

Thesis

Effective information sharing between the robot developer, the human teammate, and the collaborative robot is essential for human-robot collaboration and is enabled by the modeling and algorithmic advances presented in this dissertation.

Outline of the Dissertation

- Chapter 2 motivates the thesis and defines the terms: robot developer, human teammate, collaborative robot, and effective information sharing.

  Motivated by the quest of developing collaborative robots, three characteristics of effective human teamwork are summarized. Through a review of related research and potential applications, several (both existing as well as previously unexplored) paradigms of realizing collaborative robots with these characteristics are presented. Guided by this review, three opportunities for effective information sharing for human-robot collaboration are identified.

- Chapter 3 formalizes the problems of effective information sharing for human-robot collaboration, which are addressed in the dissertation.

  The problems addressed in the dissertation are informed by the following work on developing collaborative robots by the author: "Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time," Vaibhav Unhelkar*, Przemyslaw Lasota*, Quirin Tyroller, Rares-

Darius Buhai, Laurie Marceau, Barbara Deml, and Julie Shah, *IEEE Robotics and Automation Letters*, 3(3):2394–2401, 2018 [159].

- Chapter 4 addresses information sharing between the robot developer and the collaborative robot for modeling the human teammate.

  A representation, named Agent Markov Model (AMM), is developed for describing the sequential decision-making behavior of other agents. In order to facilitate information transfer during model specification, a suite of algorithms for learning the AMM given different types of inputs is developed. Central to these algorithms is Constrained Variational Inference (CVI), a hybrid paradigm for incorporating both data and high-level domain expertise during model learning. The utility of hybrid model learning is validated using experiments with both synthetic and human subject data.

  The technical content of this chapter is based on the following work led by the author: "Learning Models of Sequential Decision-Making with Partial Specification of Agent Behavior," Vaibhav Unhelkar and Julie Shah, in the *AAAI Conference on Artificial Intelligence (AAAI)*, 2019 [155].

- Chapter 5 addresses information sharing between the robot developer and the collaborative robot for autonomously generating the robot behavior.

  A framework, named ADACORL, for accelerating the specification of a robot's collaborative behavior is proposed. ADACORL enables the generation of adaptive collaboration with reduced labeling effort on part of the robot developer. The utility of ADACORL for enabling fluent collaboration is demonstrated in two human-robot collaborative tasks, which have state spaces significantly larger than the problems considered in prior art.

  The technical content of this chapter is based on the following work led by the author: "Semi-Supervised Learning of Decision-Making Models for Human-Robot Collaboration," Vaibhav Unhelkar*, Shen Li*, and Julie Shah, in the *Conference on Robot Learning (CoRL)*, 2019 [160].

- Chapter 6 addresses information sharing between the collaborative robot and the human teammate during human-robot collaboration.

  A framework for enabling collaborative robots to decide *if, when, and what* to communicate during sequential human-robot collaborative tasks is developed. In order to enable effective information sharing, the framework models impact of robot's communication on human's behavior, includes models of communication cost, and algorithms for planning under uncertainty. The benefit of communication decision-making is demonstrated in collaborative tasks with multiple communication types and short planning times.

- Chapter 7 summarizes the modeling and algorithmic contributions presented in the dissertation, and highlights future direction for advancing the theory and practice of human-robot collaboration.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Motivation: Developing Collaborative Machines

Humans and machines often possess complementary skills. The realization of this fact by experts across domains is leading to a steadily growing interest in robots and software agents that can assist humans and collaborate with them. For instance, robots are being developed to enhance the capabilities of astronauts [28], doctors [46, 57, 80], and factory associates [159, 172] — i.e., humans performing specialized tasks. In the long term, machines will support humans in myriad tasks, thereby continually transforming the landscape of human life.

I refer to the robots and software agents that, instead of being a passive tool, proactively support humans as **collaborative machines**. In order to unlock the potential of humans and machines working together, we first need to answer

How does one develop these collaborative machines?

This critical question is too broad to have a simple answer. We can glean insights on the characteristics of an effective collaborator from studying human teams. However, developing collaborative machines additionally requires the development of novel technology that can realize these characteristics. This dissertation provides such technology, in the form of modeling and algorithmic advances, that enables the design and development of collaborative machines. The underlying

problems that these advances address are motivated by insights from studies of human teams, recent research in human-machine interaction, and my experience with designing and deploying collaborative robots among humans.

2.1 Insights from Human Teams

Study of collaboration among humans is instructive for developing effective collaborative machines [54, 139]. Humans successfully collaborate in a variety of contexts, both challenging and seemingly mundane. While the size and composition of human teams differ based on the context, research on human teaming points to necessary characteristics of a team (and its members) that are critical for successful collaboration [14, 48, 79, 98, 128]. This section presents three such characteristics of effective human teams that motivate my thesis.

Modeling the Teammate Members of effective teams maintain a model of their teammate's task, abilities, and behavior [23, 35]. The fidelity, granularity, and instantiation of this model depend on the teaming context. However, a model of the teammate is essential for fluent collaboration. Humans generate this model using prior knowledge, observation, and team training [129, 134].

Adapting to the Teammate Members of effective teams proactively adapt their plans in response to the uncertainty in the plans of their teammates and the outcome of the team's actions [34, 134]. To generate this adaptive behavior, effective teammates utilize (a) a model of the teammate for inferring their intent and anticipating their actions, and (b) a model of the world to anticipate action outcomes.

Effective Communication with the Teammate Members of effective teams effectively share information with their teammates to accomplish tasks in a partially known and uncertain world [17, 24, 87, 137]. Effective information sharing is achieved by (a) anticipating the information needs of the teammate, and (b) gauging the cost and benefit of each communication.

Figure 2-1: Robots interacting with humans during automotive final assembly.

2.2 Implications for Collaborative Machines

The characteristics of effective human teams are context-invariant and, thus, also necessary for effective human-machine teaming. However, equipping collaborative machines with these characteristics is challenging; machines are inherently limited in their extent of prior knowledge as well as ability to train and share information with humans. Thus, in practice, a human expert (or a team thereof) is tasked with equipping collaborative machines with these characteristics. I refer to this human expert as the **robot developer**.

During the course of writing this dissertation, I have assumed the role of the robot developer and deployed a few of the first **collaborative robots** (i.e., collaborative machines with physical embodiment) that work with humans in the real world [158, 159]. Figure 2-1 (left) depicts one of these systems: Rob@Work 3, a mobile manipulator performing assembly tasks on the conveyor belts of automotive factories while sharing space with humans. Informed by this experience and recent research in human-machine interaction [21, 26, 30, 47, 73, 78, 103, 140, 150], in the next sections, I summarize the prior art for developing collaborative robots that exhibit the previously discussed characteristics of effective teammates.

The discussion presented next is general and applies across collaborative tasks (including shared workspace, shared manipulation, and handover tasks) and domains (such as disaster response, healthcare, and manufacturing). However, to ground the discussion in practice, I will utilize a shared workspace task from the domain of collaborative manufacturing.

Running Example: Human-Robot Collaborative Manufacturing There is an emerging desire across manufacturing industries to deploy robots that support people in their manual work. As a running example, I utilize one such opportunity, where a collaborative robot is tasked with delivering tools and materials to the human associates [156]. Such a collaborative robot would need to fetch parts from depots and deliver them to humans at their workstations while ensuring safe interactions. Figure 2-1 (right) shows CobotSAM, an example of such a collaborative robot, which was developed during this dissertation research in collaboration with the BMW automotive company [159].

CobotSAM is tasked with fetching parts from one end of the factory line, where the parts are housed in a depot, and delivering them on the other end, where human associates need this part. While completing this part delivery, the collaborative robot has to share its environment with its teammate and other human associates (e.g., associates responsible for cleaning the factory or for refilling the depot with parts). The use of a mobile collaborative robot to perform this task allows human associates to focus on dexterous, value-added work of car assembly, yielding significant time and cost savings [121].

Physically, CobotSAM is built by mounting a UR10 collaborative robot arm [161] on a linear axis unit (e.g., [1, 74]); both the arm and the linear unit are certified for industrial use. The arm enables the robot to perform manipulation tasks, while the linear axis unit provides the desired mobility. The mobile robot is equipped with an on-board 2-dimensional laser scanner and an off-board depth camera.

2.3 Modeling the Human Teammate

The first characteristic of effective teams suggests that, for fluent collaboration, collaborative machines need a model of their human teammates. For instance, in order to safely share space with the human, CobotSAM needs a predictive model of the human teammate's goal-directed motion. Several paradigms can be used to develop this teammate model. Figure 2-2 depicts a qualitative estimate of the devel-

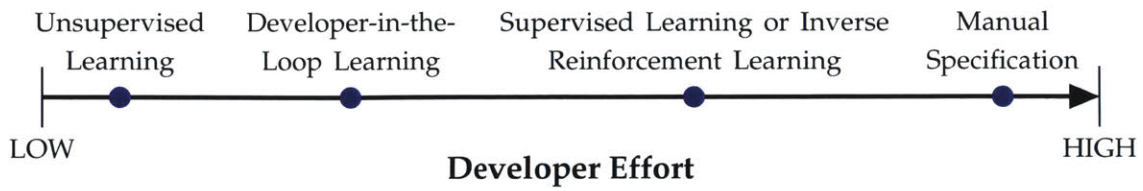

Figure 2-2: Paradigms for specifying models of the human teammate.

oper effort required to specify the teammate model under the different paradigms. For rapid prototyping and deployment of collaborative machines, model specification paradigms with lower developer effort are desired.

2.3.1 Unsupervised Learning

In theory, by utilizing unsupervised learning algorithms and observation of their teammate's behavior, machines can learn a teammate model without any intervention from the developer [86]. For instance, CobotSAM using its sensors can observe the motion of human associates and learn a predictive model of their motion. However, in practice, the sample complexity of unsupervised learning from observation is often prohibitively large for applications of human-machine interaction. Thus, a robot developer is tasked with developing this model.

2.3.2 Manual Specification

On the other end of the spectrum, instead of utilizing data or algorithms, the developer can manually specify a model of the human teammate. For instance, while developing a collaborative robot for manufacturing, such as CobotSAM, the developer may have domain expertise regarding the human associate's task and waypoints of interest which can help her specify a human model.

However, relying solely upon manual specification to develop the model is time-intensive and typically leads to incomplete models, as the developer needs to specify human teammate's behavior in all possible scenarios. This difficulty in manually specifying the models has motivated the development of algorithmic approaches that utilize labeled behavioral data [5, 51, 108].

2.3.3 Prior Art: Learning with Labels

Theses algorithmic approaches, such as supervised or inverse reinforcement learning, require the developer to follow the following steps,

1. specify decision factors that impact the human teammate's decisions;
2. collect unsupervised data of human behavior; and
3. annotate the data to obtain a labeled (supervised) dataset of behavior.

First, the developer specifies the decision factors (features) that impact the human's behavior. For instance, the motion of human associates in a factory primarily depends on two decision factors: where they want to go (i.e., their task/goal) and where they currently are (i.e., their position). Typically, a subset of the decision factors is observable (e.g., position), while the rest are latent and challenging to measure (e.g., goal). The latent decision factors often correspond to the human teammate's mental states, such as goals, preferences, attention, and workload.

Next, using sensors, the developer creates an unsupervised dataset of human behavior (i.e., observable decision factors and decisions). For instance, while developing CobotSAM, the developers utilized the off-board depth camera to record trajectories of the human teammate's motion. Finally, to generate the teammate model using supervised or inverse reinforcement learning techniques [77, 118, 173], a dataset of all features (including labels of the latent decision factors) is required. These labels are obtained manually: either by querying the human teammate or by utilizing domain expertise available to the robot developer.

Such approaches that rely on labeled data of behavior have been utilized for several proof-of-concept collaborative tasks [31, 69, 104, 125, 132, 159]. However, in practice, the process of manually annotating the dataset can be time-intensive, thereby increasing the time required for prototyping and deploying collaborative machines. Moreover, approaches relying on data alone may also fail to recover the true teammate model (i.e., the model is non-identifiable given data alone). This phenomena occurs as it is often difficult to specify and quantify human's mental states, and multiple models can explain the observed behavior equally well.

2.3.4 Opportunity: Effectively Utilizing Domain Expertise

While developing the teammate model, domain knowledge pertaining to the teammate's behavior is often available to the robot developer. In addition to the specification of decision factors, the domain experts also have access to partial knowledge of the dynamics of decision factors, change-points of decision factors, and impact of decision factors on decisions. For instance, in a factory, the developer may have access to the order in which the human teammate accomplishes her assembly task.

In order to address the limitations of approaches that learn with labeled data, a hybrid learning paradigm is essential, one where model learning is done both from data and effective utilization of the robot developer's domain expertise. This hybrid paradigm, which I denote as **developer-in-the-loop learning**, has the potential to reduce sample complexity of labeled data and alleviate the ambiguity between the true and the learned model that exists when learning with behavioral data alone. However, models and algorithms for utilizing the partial knowledge (i.e., domain expertise) are currently lacking.

2.4 Adapting to the Human Teammate

For fluent collaboration, collaborative machines need the ability to adapt to their human teammate. Mathematically, this ability can be encoded as a function termed as **interaction policy**, which specifies the action that the machine should choose in a given scenario. For instance, in order to safely and efficiently share space with the human teammate, CobotSAM needs an interaction policy that allows it to decide whether to wait, move forward, or move backward while navigating in the factory.

Similar to the specification of the teammate model, multiple paradigms can be utilized to specify the interaction policy (see Fig. 2-3). Each paradigm requires a different level of manual effort from the robot developer and the human teammate. In this section, I discuss these paradigms and present an opportunity of *effective information sharing* for specifying the interaction policy.

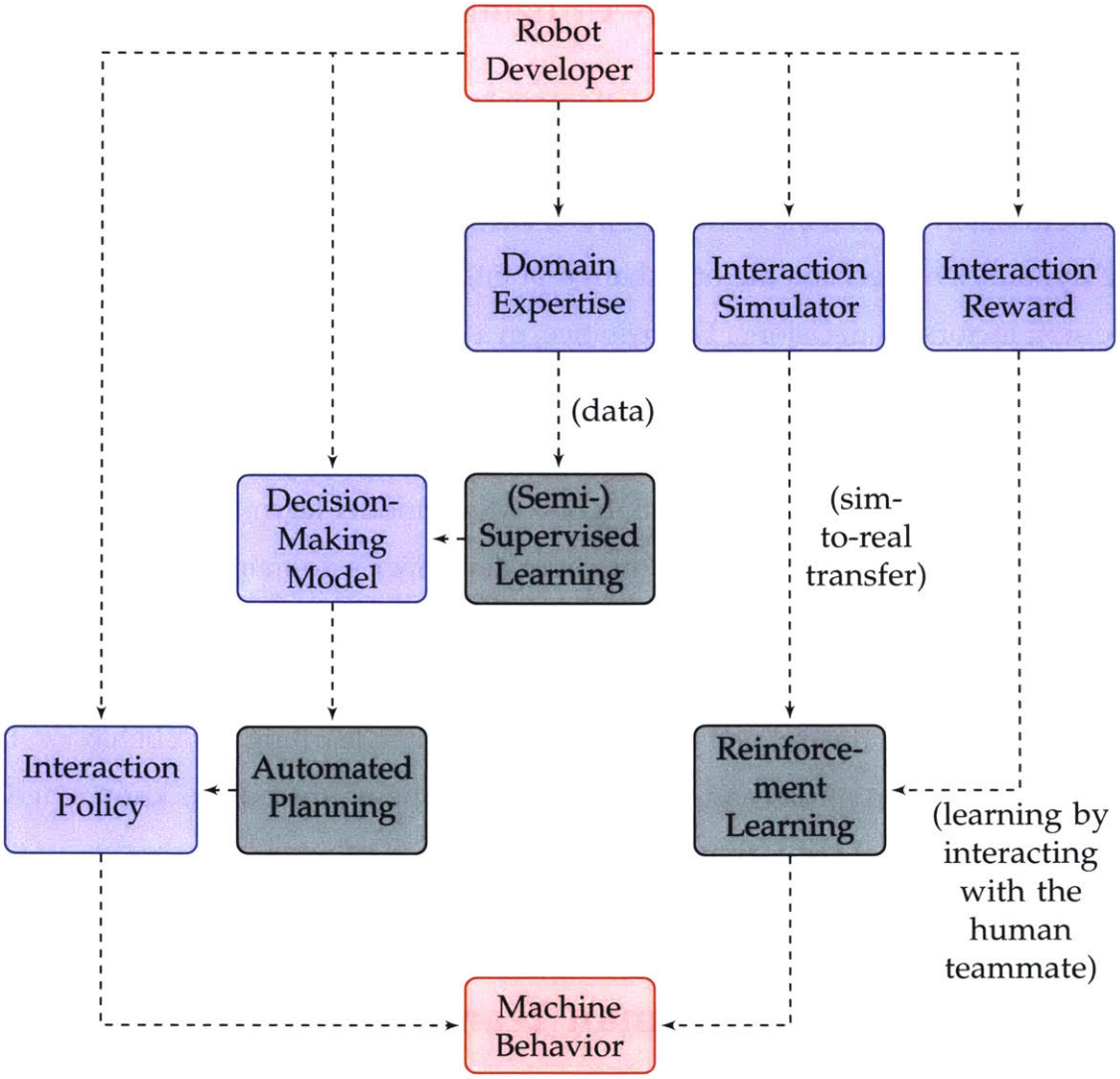

Figure 2-3: Paradigms for specifying a collaborative machine's adaptive behavior.

Based on the specification paradigm, the inputs from the robot developer (denoted in blue in Fig. 2-3) either directly specify the machine behavior (e.g., in the case when interaction policy is directly specified); or are used in conjunction with suitable learning/planning algorithms (denoted in gray in Fig. 2-3) for generating the machine's behavior. In addition to the developer's specification effort, in the learning-based paradigms, the manual effort also includes the time spent by the human teammate for interacting with the machine while it is training (i.e., while the machine is learning the interaction policy).

2.4.1 Reinforcement Learning

Under the paradigm of reinforcement learning (RL) [39, 99, 145], in theory, the machine can learn its interaction policy solely by training with the human teammate and without any intervention from the robot developer. For instance, using this paradigm, CobotSAM can learn how to accomplish its task (safe and efficient part delivery) by repeatedly trying to deliver parts to the human associates and learning from its successes and failures. However, in practice, utilizing this paradigm for human-machine interaction is challenging for a variety of reasons.

Safe interaction with humans is essential; however, guaranteeing safe exploration during reinforcement learning is an open problem [8, 42]. Further, application of RL to human-machine collaboration requires a large number of episodic interactions (typically, of the order of tens of thousands) [22]. Training with the human teammate for this large number of episodes is prohibitive in most scenarios of human-machine interaction. Adding to these challenges, at the implementation level, RL approaches need an external mechanism (typically, a developer) to reset the human-machine team's environment for every episode of learning.

In order to address these challenges, learning in simulation and then transferring the policy learned in simulation to the real world is an oft-used strategy while utilizing reinforcement learning [91, 95, 119, 124, 151]. However, this strategy too is challenging for human-machine interaction, the primary reason being the difficulty of simulating the human teammate's behavior [144]. Further, irrespective of whether learning is performed in simulation or real-world, the design of a suitable reward signal is a challenging prerequisite.

Given the above challenges, it is not surprising that the existing approaches to specifying an interaction policy for human-machine collaboration rarely rely on reinforcement learning. However, advances in reinforcement learning — specifically, those focusing on guaranteeing safe exploration and reducing sample complexity — can make it a highly desirable paradigm for generating adaptive and interactive behavior of collaborative machines.

2.4.2 Manual Specification of the Interaction Policy

On the other end of the spectrum, in complete contrast to the learning-based approaches, the robot developer can manually specify the interaction policy of a collaborative machine. In the absence of principled algorithmic approaches, the interaction policy was typically manually specified at the inception of the field of human-robot interaction [25]. The manual specification was done by encoding the developers' domain expertise either in the form of rule-based systems or through Wizard-of-Oz control of the machine [122, 133].

Despite the domain expertise available to the robot developer, manually handcrafting the interaction policy often results in a sub-optimal human-machine collaboration. Among other factors, it requires the robot developer to reason about and specify the machine's response for all the possible interaction scenarios — a process that is not only challenging but, often, also immensely time-consuming. For instance, it would require CobotSAM's developer to specify whether the robot should stay put, move towards the workstation, or move towards the depot for each possible combination of human and robot position histories.

Despite its limitations, manual specification of the interaction policy has utility for rapid prototyping and deployment. For instance, the behavior policies of autonomous cars are typically manually specified [100, 107, 162].

2.4.3 Prior Art: Specification of Decision-Making Models

In order to address the challenges that arise from relying on either reinforcement learning or manual specification, in recent human-machine interaction research, model-based planning approaches have been developed for generating the machine's interaction policy [19, 53, 58, 101, 104, 159]. Broadly, these planning approaches involve two steps,

1. specification of a decision-making model; and
2. generation of the interaction policy through algorithms for decision-making.

The robot developer specifies the decision-making model for interaction by utilizing her domain expertise. The specification is done either wholly manually or through a hybrid approach, where the model structure is specified by the developer and model parameters are learned using labeled data. Motivated by different subclass of human-machine interaction scenarios, different paradigms of decision-making have been explored in recent research, including mixed-integer linear programs [45, 138], variants of Markov decision processes [16, 104], planning as inference [152], planning as optimization [141, 111], sampling-based planners [81], timed Petri nets [19], and underactuated control [36, 125].

For these approaches, typically, the manual effort is limited to the specification of the decision-making model by the robot developer and does not involve additional effort on the part of the human teammate. This is possible as the interaction policy is computed given the decision-making model and does not need to be learned via repeated interactions with the human teammate.

The structure and specification process of the decision-making model and the decision-making algorithm used for generating the interaction policy differs based on the paradigm. However, irrespective of the paradigm, the robot developer needs to specify models of the shared environment, collaborative task, the collaborative machine, and the human teammate. For instance, while designing CobotSAM the robot developers specified a map of the environment (i.e., the environment model), the parts that the robot had to deliver and the order in which they were to be delivered (i.e., the task specification), a model of the robot's motion dynamics (i.e., the model of collaboration machine), and a predictive model of human motion and goal (i.e., the human teammate model).

In comparison to the specification of the interaction policy, algorithmic computation of the interaction policy speeds up the generation of machine behavior. However, in practice, due to the presence of multiple model components, manual specification of the decision-making model still remains time-intensive. For instance, the development of the CobotSAM system was completed by a team of robot developers over a period of two years.

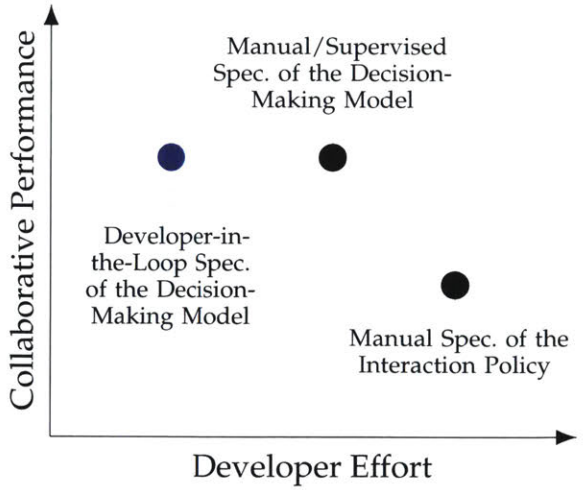

Figure 2-4: Opportunity (denoted in blue) for reducing developer effort while specifying a collaborative machine's decision-making model.

2.4.4 Opportunity: Reducing Effort in Model Specification

In order to further accelerate the deployment of collaborative machines, approaches that algorithmically generate not only the interaction policy but also the corresponding decision-making model will be important. Previously, in Sec. 2.3, we have discussed the opportunity available for developer-in-the-loop learning of the teammate model. This opportunity also presents itself for the specification of the decision-making model. Domain knowledge pertaining to the components of the machine's decision-making model (namely, task objective, environmental constraints, machine's dynamics, and human behavior) is often available to the robot developer. However, this domain knowledge is typically incomplete and needs to be augmented with novel learning and specification approaches.

I posit that through novel developer-in-the-loop approaches for specifying decision-making models, fluent human-machine collaboration can be achieved despite reduced developer effort (see Fig. 2-4). My belief in this hypothesis is further emboldened by recent investigations into approaches that learn the interaction reward through demonstrations and querying [11, 49, 126, 136]. However, developer-in-the-loop approaches for specifying the transition function (dynamics) of the collaborative machine's decision-making model are currently lacking.

2.5 Communication with the Human Teammate

For the design of collaborative machines, the first two insights from human teams (namely, modeling and adapting to the teammate) manifest themselves as opportunities for effective information sharing between the robot developer and the collaborative machine. In contrast, the third insight emphasizes the need for effective information sharing between the teammates (i.e., the human teammate and the collaborative machine) during the execution of the collaborative task.

Execution-time communication between the human and machine is essential in certain applications — such as a chatbot answering a user query [130], or a tutoring robot providing help/instructions [117]. In others, it aids in improving the interaction by allowing agents to share observations regarding the environment [163] and make inferences regarding teammates [166, 138]. For instance, CobotSAM could utilize speech to infer or influence the goal (latent state) of its human teammate. Further, information sharing in human-machine teams has the potential to influence the human teammate's subjective perception of the machine positively — e.g., by engendering trust among teammates [90, 131, 143]. Due to this potential of information sharing, in the last decade, significant research has been conducted to enable communication between humans and machines [93].

2.5.1 Communication Modality

Several modalities for human-machine communication are actively being researched — such as natural language [92, 149], visual signals [12], eye gaze [4], and the physical motion of a robot [32, 72, 146]. These modalities enable collaborative machines to either convey information to the human teammate, interpret the information received from the human teammate, or both. A challenge while developing a modality is that of symbol grounding, i.e., grounding the communication in the model of the world that the collaborative machine maintains [148]. In order to resolve this challenge, approaches that facilitate grounding through language models, gestures and interaction have been developed [9, 166].

In addition to communication modalities that enable explicit communication between teammates, the ability to communicate implicitly through social cues and joint action has also been identified [3, 68, 84]. Design of these modalities is an important prerequisite for achieving human-machine communication. However, in addition to being able to communicate, prior research indicates that effective use of this ability is necessary to realize its benefits for teaming [17].

2.5.2 Communication Policy

While communication provides several benefits for teamwork, it also incurs costs. Information sharing typically requires both the sender and the receiver to expend resources, which might otherwise be used for improving their performance on the collaborative task. For instance, if CobotSAM repeatedly asks the human associate information about her goal, the human associates workload may increase and attention to the collaborative task might degrade. Thus, in this case, unwarranted use of communication can result in deterioration of the team's task performance and the human associate's subjective perception of the robot.

While the development of a communication modality addresses the question,

How to communicate?

equally important questions that need to be addressed are

If, when and what to communicate?

During a human-machine collaborative task, too little communication may lead to poor situation awareness and miscoordination. On the other hand, communicating too often may lead to information overload [13, 114] or even humans ignoring communications from the machine [33, 110]. Thus, both low and high levels of communication are detrimental to team performance. As highlighted in Fig. 2-5, successful machine teammates will need the capability to effectively use their communication modality or, equivalently, a **communication policy**. Similar to the specification of the interaction policy, multiple paradigms can be explored for specifying a collaborative machine's communication policy.

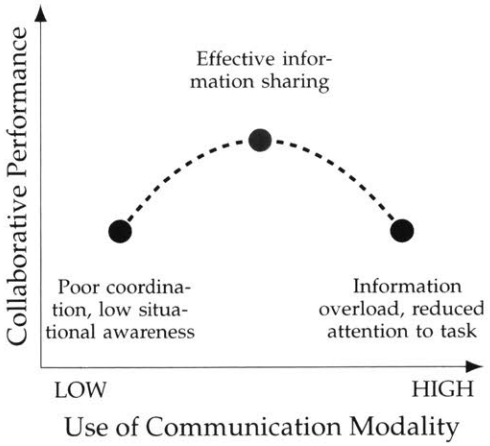

Figure 2-5: Opportunity (denoted in blue) for improving human-machine collaborative performance through effective information sharing.

Manual Specification For reasons identical to the specification of the interaction policy (discussed in Sec. 2.4.2), manual specification of the machine's communication policy remains effort-intensive and results in sub-optimal policies. While the importance of effective communication has been identified for teamwork, algorithmic approaches for generating the communication policy for human-machine collaboration have been relatively under-explored [88, 157].

Planning-Based Approaches In contrast to human-machine teams, however, there has been significant research and development towards algorithmically generating the communication policy for multi-robot teams. For multi-agent systems, several variants of the decentralized partially observable Markov decision process (dec-POMDP) model [106] that include communication have been proposed [44, 116, 142]. Further, multiple algorithms that *solve* these models have also been developed [7, 96, 123, 167, 168, 169]. With the help of developer-specified environment and agent models, these algorithms are capable of reasoning about action and observation certainty and making communication decisions. While instructive for developing solutions for human-machine teams, the agent models in multi-agent approaches do not include latent states (such as compliance and preference) that are important for capturing the effect of machine's communications on human behavior and critical for reasoning about human-machine communication.

Learning-Based Approaches In order to circumvent the need for specifying models (of environment as well as of the teammate's behavior) for generating the communication policy, approaches that learn a communication policy through multi-agent reinforcement learning have also been developed [37, 43, 66]. These learning-based approaches require the team to communicate at training time and generate a decentralized communication policy for execution time. However, similar to RL-based approaches to arrive at the interaction policy, the number of training episodes required for learning a communication policy remains prohibitively large for training human-machine teams. Thus, in summary, the discussion of related work highlights that novel approaches are needed for generating communication policy for human-machine collaboration.

2.5.3 Opportunity: Decision-Making for Communication

During human-machine collaboration, a machine may be able to share information, such as its intent or observations, which it cannot anticipate prior to execution. Sharing of this information, when it is beneficial for coordination, can improve the collaborative performance of the human-machine team. Achieving the ability to weigh the cost and benefit of a communication message necessitates the design of novel decision-making techniques for human-machine communication.

Decision-making for human-machine communication will need to be performed during interaction, especially in contexts where environment model is incomplete or the problem size (state space) is large. In order to enable communication decision-making during task execution, collaborative machines will need approaches that can jointly reason about the robot's actions and communications in the limited time available during interaction. Further, to communicate effectively, the collaborative machines will need to reason about the latent states that capture the effect of the communications on the human teammate's behavior. Modeling this effect would require the machine to maintain models of communication cost and human decision-making.

Of broad applicability and use will be development of general decision-making approaches that work across collaborative tasks and communication modalities (similar to existing general decision-making paradigms in robotics, such as motion planning and task planning). Finally, to address challenges of deploying collaborative machines in the real world, approaches that do not require extensive training data or developer effort are especially desirable.

2.6 Summary

Motivated by the quest of finding a principled approach to developing collaborative machines, in this chapter, we focused on three characteristics of effective human teamwork and discussed their implication for human-machine collaboration. In these discussions, we came across three entities that play a role in the design and deployment of collaborative machines — namely, the *collaborative machine* itself, the *human teammate* whom the machine interacts with and, equally importantly, the *robot developer* who is tasked with designing the machine.

Irrespective of the collaboration scenario, each of these three entities possess different information. Through a discussion of recent and on-going research in human-machine interaction, we observed that encoding the three characteristics of effective teaming in collaborative machines corresponds to opportunities for effective information sharing between the three entities. In the following chapter, I mathematically formalize these opportunities of *effective information sharing for human-machine collaboration* and state the problems addressed in this dissertation.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Problem Statements

The insights from studies of human teams and deployments of collaborative robots point to the importance of effective information sharing for human-machine collaboration. The opportunities for enabling effective information sharing, as discussed in the previous chapter, apply across a variety of collaboration scenarios.

In this chapter, I formalize these opportunities by formulating three problems, corresponding to each of the three opportunities. The first two problems focus on the transfer of information from the *robot developer* to the *collaborative machine* for the purposes of specifying a model of the human teammate and the interaction policy, respectively. The last problem addresses the need for execution-time decision-making for information sharing between the *human teammate* and the *collaborative machine*. Before discussing the problem statements, however, I present the human-machine collaborative scenarios of interest and provide a model for representing these scenarios.

3.1 Scope: Collaboration Scenarios

In this dissertation, I focus on a subclass of human-machine collaboration scenarios — namely, human-robot dyads performing sequential tasks with a shared and known task objective (see Fig. 3-1). Each member of the team (i.e., the human teammate and the collaborative robot) potentially has different abilities.

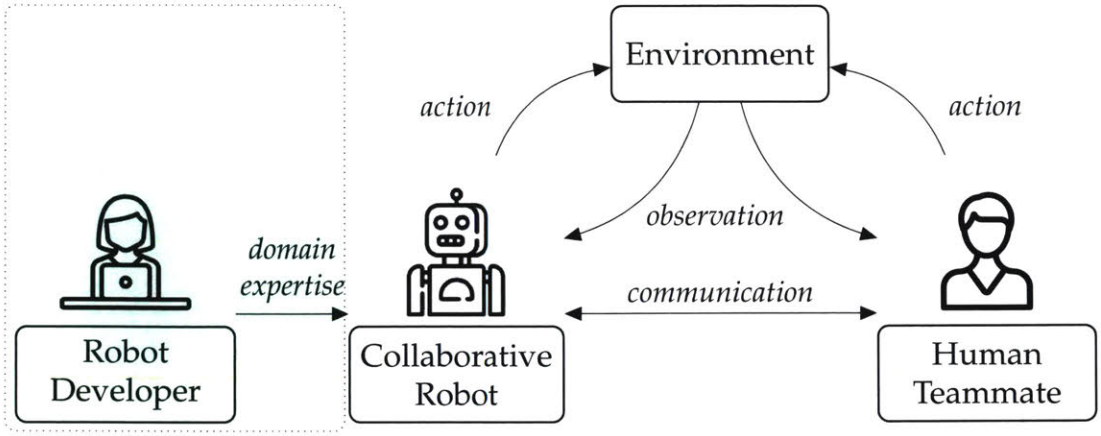

Figure 3-1: An abstraction for human-machine collaborative scenarios of interest.

Prior to task execution, the robot developer programs the machine by utilizing her domain expertise. During task execution, both the human and the robot have autonomy over their actions and receive observations from the environment. The information that each agent has might differ; this might happen due to decentralized nature of the multi-agent task, incomplete knowledge of the environment, or an inaccurate model of the teammate's behavior. Depending on the available communication modality, the teammates can share certain types of information with each other during task execution.

3.1.1 Motivating Domains and Applications

The focus on sequential tasks with a shared and known objective is motivated by their numerous instantiations, which are found across domains, including homes, offices, hospitals, and outer space. For instance, consider a robotic assistant (such as CobotSAM) tasked to support a human in the assembly of parts in a factory [159]. Assembly tasks typically require multiple steps and are, thus, sequential. Further, the human, the robot, and the robot developer know the objective of the assembly task a priori. Finally, by monitoring the assembly performance (e.g., the number of parts assembled), one can obtain an objective measure of the human-machine team's collaborative performance.

Other instantiations of human-machine collaboration that are a member of the chosen subclass include a robot supporting an astronaut [28], a robotic scrub nurse supporting a human surgeon [57], a service robot supporting a human user, and an unmanned aerial vehicle supporting a first responder [163].

3.1.2 Task Model

As discussed in Sec. 2.4.3, multiple paradigms can be utilized to represent human-machine collaboration. Due to the focus on tasks with a known and shared objective, I utilize a multi-agent variant of Markov decision processes (MDP) to model the collaboration scenarios of interest. Multi-agent variants of the MDP have a rich heritage in multi-agent systems and computational human-robot interaction research [16, 101, 106].

Specifically, I use a decentralized partially observable Markov decision process (dec-POMDP) with a factored transition and observation model to describe the collaborative tasks of interest. This factored variant of the dec-POMDP model describing human-robot dyadic tasks of interest is detailed as follows:

- The task includes two agents, namely, the human (denoted by H) and the robot (denoted by R);
- S denotes the set of states s of the collaborative task. I use a factored representation where $s \equiv (s_H, s_R, s_E)$, in which s_H and s_R correspond to human- and robot-specific features, respectively, and s_E denotes additional features (e.g., based on the task structure and the environment);
- $A_H \times A_R$ denotes the set of joint actions $a \equiv (a_H, a_R)$. A_H and A_R denote the action space of the human and the robot, respectively. The actions of agents can be temporally-extended (i.e., macro actions). While the actions are represented as joint actions it does not prevent the agents from making decisions and acting autonomously – a necessary feature for modeling human-robot teams;
- $\Omega \equiv \Omega_H \times \Omega_R$ denotes the finite set of joint observations o. Ω_H and Ω_R denote the set of the human and robot observations, o_H and o_R, respectively.

- The state dynamics are assumed to be Markovian and are governed by the transition function $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$. The transition function has the following factored structure,

$$T(s|s,a) = T_H(s'_H|s,a) \cdot T_R(s'_R|s,a) \cdot T_E(s'_E|s,a), \tag{3.1}$$

  where T_H, T_R, T_E are transition functions for the state factors;
- The team receives a shared reward at each step, $R(s,a) : S \times A \rightarrow \mathbb{R}$;
- The observation function is denoted as $O(o|s,a,s') : S \times A \times S \times \Omega \rightarrow [0,1]$. The observation function exhibits the following factored structure,

$$O(o|s,a,s') = O_H(o_H|s,a,s') \cdot O_R(o_R|s,a,s') \tag{3.2}$$

- $\gamma \in [0,1]$ denotes the discount factor; and
- (optionally) $t_f \in \mathbb{Z}^+$ denotes the time horizon of the problem.

The human-robot team's objective is to maximize the expected cumulative discounted reward. I reiterate that the task model is a specification of the the collaborative scenario and does not prescribe or proscribe how the teammates should act to accomplish the task. Consequently, the state of the task model does not model the mental states of either the human or robot.

3.1.3 Instantiations of the Task Model

In order to exemplify the varied types of collaborative scenarios that can be modeled using the task models, I present two instantiations of the model describing near-term applications of human-robot collaboration.

Shared Workspace Tasks As the first example, consider a human and a robot performing pre-allocated tasks in a static shared workspace. For instance, a robot and human navigating a narrow lane [112] or Rob@Work 3 sharing space with a human associate in the factory [158]. A simplified example from a grid-world is de-

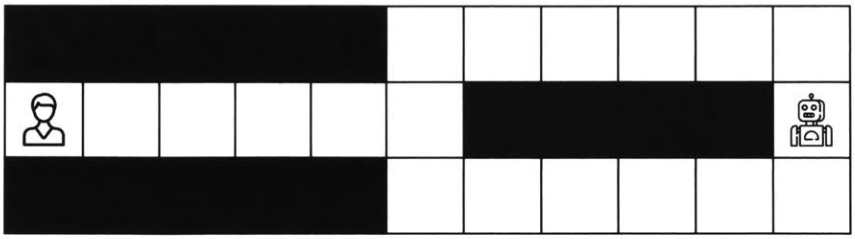

Figure 3-2: A simplified example of the shared workspace task.

picted in Fig. 3-2, where black grids denote static obstacles. The human and robot, who are working in the grid, need to switch places in order to complete their task. The teammates need to coordinate their motion to complete the task safely and efficiently. Real-world instantiations of the shared workspace task, from the domain of automotive final assembly, are depicted in Fig. 2-1. The shared workspace task can be readily represented using the task model.

For instance, for the simplified scenario, the important features include position of the human and robot and the map of the environment. These features correspond to the state components as follows: s_H (human position), s_R (robot position), and s_E (map). Consequently, the transition models T_H and T_R correspond to the motion dynamics of human and robot, respectively. The motion dynamics depend on the physical capabilities of the agents and the map of the shared workspace. The dynamics of the environment (e.g., motion of dynamic obstacles) is captured by T_E. For the static environment in the grid-world example, the transition model of s_E is given as follows: $T_E(s'_E|s,a) = \mathbb{1}(s'_E = s_E)$.

Depending on the teaming context, the agents may have complete or partial observability of the state s. For instance, in scenarios with partial observability, the teammates might have partial knowledge of the map or have a limited sensing radius. In contrast, for applications in known and structured environments (such as factories), the teammates may have complete knowledge of the map and each other's location. In order to succeed at the task, the human and the robot have to work alongside each other by coordinating their motion. Thus, the reward function specifies that the robot should reach its goal as soon as possible while maintaining a safe distance to the human.

Collaborative Assembly Tasks As the second example, consider the set of tasks where the human and the robot are collaborating to complete a multi-step assembly. For instance, a human-robot team assembling an automotive engine [159] or a robotic assistant supporting an astronaut in unpacking a shipment. As an intuitive example, consider a collaborative robot and human making sandwiches in a futuristic kitchen. Due to their complementary abilities, the human is performing the dexterous component of the task of making the sandwich, while the robot is handing over ingredients to the human.

For the example, the definition of s_H, s_R, and their corresponding transition models T_H, T_R remains identical to the first example. The task progress of the multi-step task (i.e., which steps/sub-tasks of the assembly have been completed) is encoded via s_E. The transition model T_E corresponds to the task recipe, e.g., a recipe tree [65] or an enumeration of task plans. The overall transition model exhibits the required factored structure, i.e., $T = T_H \cdot T_R \cdot T_E$. The reward specifies that the collaborative task should be completed as soon as possible while maintaining safety. Since safety can be quantified using the human-robot distance (a function of s_H and s_R) and the task completion can be measured using the task progress s_E, as required by the task model, the reward can be specified as a function of the task model's states and actions.

3.1.4 Assumptions

By formalizing the collaborative task as a factored variant of the dec-POMDP, I focus on a subset of possible human-robot collaboration scenarios. I discuss the implications of this modeling choice as follows:

- By representing exactly two agents, the task model is designed for dyadic human-robot interactions.
- By utilizing a discrete representation of the task, the task model does not capture tasks with continuous features (states). A workaround to this limitation is discretizing continuous features (if any) that are needed for describing the task.

- Due to its Markovian assumption, the task model is catered towards tasks whose dynamics can be described *tractably* using a Markov model. I emphasize that, in theory, any task can be modeled as Markovian by incorporating sufficient information in the state. Hence, the utilization of a Markovian representation does not further limit the tasks that the model can represent. However, in practice, the state space may have to be significantly large (as compared to a non-Markovian model) in order to satisfy the Markovian assumption.

Further, I also make certain assumptions about the prior knowledge available to the developer as well as the capabilities of the human and robot. I describe these assumptions and their implications for modeling collaboration as follows:

- I assume that during collaboration the human and robot have full observability of the task state. However, the task model does not model mental states of the agents. Thus, as motivated in Chapter 2, novel representations and techniques are needed that enable a robot to model and reason about these mental states. In general, the teammates cannot observe each other's mental states (such as intent). Hence, despite the assumption of full observability, the challenges and opportunities of *effective information sharing* are not only captured but emphasized in the considered problem settings.
- I assume that the developer has full knowledge of the parameters of the task model. The solutions presented in this dissertation are developed from the perspective of a robot developer, who is tasked to design a collaborative robot for a particular application. For instance, a developer deploying a collaborative robot to assist humans in a specific assembly task [159] or to assistant a surgeon in a particular procedure [57]. In such settings, the developer has knowledge of the task (e.g., map of the factory, task in the operating room). However, as detailed in Chapter 2, the challenge lies in facilitating the developer to effectively specify human models and robot policies to enable fluent human-robot collaboration. Hence, the problems considered in the dissertation aim to reduce the developer effort in specifying models and policies for interaction and communication.

- I assume that the the reward function is known not only to the developer but also the human teammate. Achieving common knowledge of the reward is an important problem and approaches, which are complementary to problems considered in this dissertation, are actively being developed [11, 49, 126, 136].

As the examples demonstrate, despite its structure and the Markovian assumption, the task model can represent a variety of applications of human-robot dyadic collaboration including sequential tasks. In this dissertation, however, I utilize instantiations of the task model that represent physical human-robot collaboration scenarios (specifically, the shared workspace and collaborative assembly tasks) to ground the proposed research problems and demonstrate their solutions. Further, as detailed next, the considered problem setting enables us to highlight and focus on the challenges and opportunities of effective information sharing between the developer, human teammate, and collaborative robot.

3.2 Modeling Other Agents with Partial Specifications of Behavior

Motivated by the need to model the human teammate, as the first problem (depicted in Fig. 3-3), I consider the problem of recovering another agent's decision-making model. I explicitly focus on the role of the robot developer in creating such models and seek to enable effective information transfer from the robot developer to the collaborative robot. The problem statement described next aims to address the gap in representations and learning algorithms for fusing both data and domain expertise while creating models of other agents.

3.2.1 Background

The behavior of the human teammate depends on a variety of decision factors. Some of these factors are known and observable to the developer, while others are unknown and latent. For instance, in human-robot teamwork and assisted

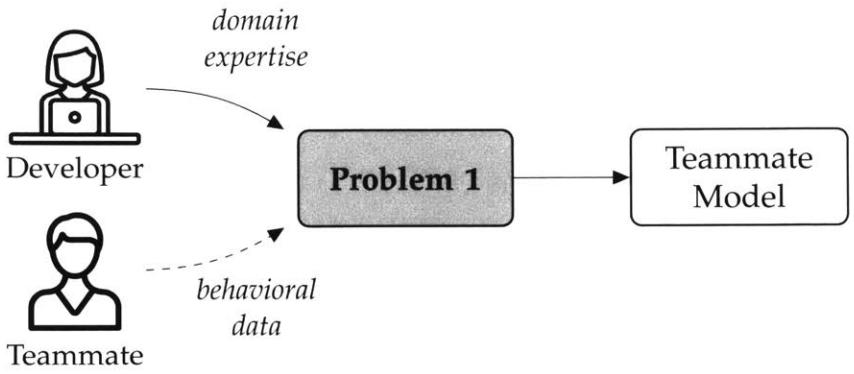

Figure 3-3: Learning model of another agent using data and domain expertise.

driving, the human's behavior depends on both observable features of the environment, progress of the task, and latent decision factors (such as human's mental states corresponding to trust, attention, intent). Hand-coded categories are typically used to quantify such latent states [41].

I denote the observable decision factors as $s_H \in S_H$ and the latent decision factors as $x_H \in X_H$. In general, both the types of decision factors are dynamic, i.e., they change during task execution. For a specific decision-making context, such as the collaborative task, the human teammate makes the decisions about his actions a_H based on the decision factors. The process of modeling the behavior of the human teammate (i.e., the other agent) is equivalent to specifying the dynamics of the agent's decision factors and the agent' *policy* π_H, which specifies a mapping from the decision factors to the agent's decisions a_H.

Behavioral Data By utilizing domain knowledge and observing the human, it is typically possible for the developer to identify the human's observable decision factors and collect behavioral data using sensors. However, typically, sensors can only measure the observable decision factors but not human's mental states. I denote the dataset of (s_H, a_H)-tuples as the *unsupervised behavioral dataset*, since it does not include data of human's latent decision factors or mental states x_H. Due to the focus on sequential tasks, this dataset comprises of execution traces (or, equivalently, trajectories) of the human's behavior.

Partial Specifications of Behavior While modeling other agents it is difficult to identify or quantify all of their decision factors. However, in specific applications, the robot developer can identify and obtain labels for the latent decision factors x_H by utilizing their domain knowledge or by querying the human teammate. In addition to the labels, the robot developer may be able to provide change-points of the latent decision factors. In general, in an execution trace, the change-points of decision factors are fewer but provide rich information for modeling the teammate.

As discussed in the previous chapter, multiple data-driven approaches are capable of learning predictive models from such labeled data. However, they may fail to recover the true model of another agent — e.g., the reward in inverse reinforcement learning (IRL) [2]. Further, the process of obtaining these labels or change-points is typically time-intensive and intrusive. Thus, approaches that can learn a teammate model with no or fewer labels are desirable.

While developing these models, domain knowledge pertaining to the true model of behavior is often available or can be acquired by the developer. This domain knowledge corresponds to the partial specification of that agent's decision factors (states), state dynamics, change-points of states, and policy (mapping from states to actions). I refer to this varied information available from the developer as the *partial specifications of behavior*.

For human-robot collaboration, ensuring that the learned model conforms to these partial specifications is necessary for maintaining model alignment. The partial specifications alone are insufficient for specifying a model, yet they can accelerate data-driven learning. Moreover, I posit that these partial specifications have the potential to alleviate the ambiguity between the true and learned models that exists when learning via behavioral data alone. The setup of the first problem is inspired by the the availability of these varied inputs for modeling the teammate, which are challenging to incorporate in existing model learning approaches.

Specifically, I assume that the dynamics of the observable decision factors is known to the robot developer, while the dynamics of the latent decision factors needs to be learned. The specifications include two types of auxiliary inputs,

namely, (a) global inputs, i.e., incomplete information about dynamics and policy, that are invariant to the behavioral dataset, and (b) local inputs, i.e., labels and change-points of the latent decision factors for the available behavioral data.

3.2.2 Problem Description

Inputs Thus, in summary, the inputs for learning a teammate model include:

- an unsupervised behavioral dataset comprising of the agent's observable decision factors (s_H) and decisions (a_H);
- dynamics of the observable decision factors;
- (optionally) specification of the latent decision factors (x_H);
- (optionally) labels and/or change-points of the latent decision factors (x_H) corresponding to a subset of the behavioral dataset;
- (optionally) partial information about the dynamics of the latent decision factors; and
- (optionally) partial information about the agent's policy.

Problem Statement Given the inputs (i.e., the unsupervised behavioral dataset and partial specifications of behavior), learn a model of the human teammate that is aligned to the true model of the teammate's behavior.

Challenges Section 2.3 motivates the importance of the problem for human-machine collaboration. The stated problem is also challenging for a variety of reasons. Prior art in human-robot collaboration typically focuses on learning the reward or policy of the human teammate; hence, a novel representation is needed that can represent not only the policy but also the dynamics of the other agent's decision factors. Next, the teammate model needs to be derived from the partial specification of behaviors, which include semi-supervised data and partial knowledge of model variables; hence, novel techniques for learning need to be developed. Finally, metrics to evaluate the alignment of the true and the learned model need to be identified.

Chapter 4 addresses these challenges by providing the Agent Markov Model (AMM), a generative representation for describing sequential decision-making behavior, and Constrained Variational Inference (CVI), a hybrid paradigm for learning generative models using semi-supervised data and model constraints.

3.3 Reducing Developer Effort for Specifying the Interaction Policy

As the second problem, depicted in Fig. 3-4, I focus on the challenge of specifying the collaborative robot's interaction policy. Similar to the first problem, the problem statement explicitly considers the role and domain expertise of the developer in this specification process. By identifying and quantifying the sources of developer's effort, the problem seeks to accelerate the process of prototyping and deploying collaborative machines.

3.3.1 Background

Section 2.4 describes the multiple paradigms available for specifying the interaction policy. Briefly, despite the knowledge of the task objective, manually handcrafting robot behavior (i.e., the interaction policy) or learning the policy directly through interaction is difficult. Given a decision-making model, algorithmic computation of the policy speeds up the generation of robot behavior.

Thus, in the current problem, I focus on the paradigm of (i) specifying the robot's decision-making model, followed by (ii) *interaction planning* (i.e., generating the interaction policy given the model). In practice, model specification still remains effort-intensive; my goal is to reduce this specification effort. For specifying the robot's decision-making model, the developer needs to specify models of the collaborative task and the human teammate's behavior. The task model corresponds to the specification of the task objective and dynamics (i.e., a model specifying the outcomes of the human and robot actions during the task).

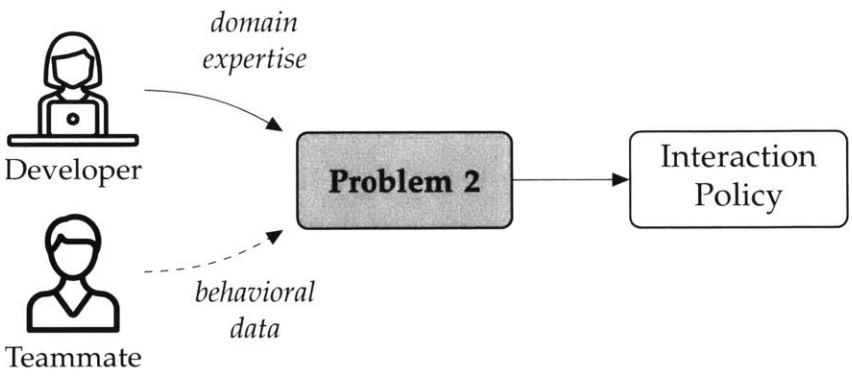

Figure 3-4: Reducing the developer effort in specifying the interaction policy.

In this problem, within the collaborative scenarios that can be described by the task model of Sec. 3.1.2, I limit the scope to scenarios where both the human and the robot have complete observability of the task state. I further assume that the task objective and dynamics are known *a priori* to the robot developer. However, note the difference between the team's *task model* and the robot's *decision-making model*. The task model by itself is insufficient for generating the robot's interaction policy, as it does not model the human teammate's behavior. A teammate model is thus necessary and allows for the prediction of human's actions during the task. While I assume necessary domain expertise is available for specifying the task model, as described in Problem 1, robot developers only have access to behavioral datasets and partial knowledge of human behavior while creating the teammate model.

3.3.2 Problem Description

Inputs Thus, in summary, the inputs for specifying the interaction policy include:

- a specification of the human-robot collaborative task provided using the task model described in Sec. 3.1.2;
- an unsupervised behavioral dataset comprising of the human teammate's observable decision factors (s_H) and decisions (a_H);
- dynamics of the observable decision factors; and
- (optionally) partial specifications of the teammate's behavior.

Task Characteristics Both the human and the robot have complete observability of the task state. Further, the dynamics of the task state and the shared reward are known to the teammate and the developer.

Problem Statement Given the inputs, arrive at the collaborative robot's interaction policy that accrues the maximum expected cumulative reward during human-robot collaborative task execution.

Challenges Similar to the challenges described for Problem 1, specification of the robot's decision-making model from varied inputs requires necessitates novel representations and learning algorithms. Despite the full observability of the task state, which reduces the collaborative task to a multi-agent Markov decision process (MMDP) [106], typically the robot will not be able to observe the human teammate's latent states. Thus, along with model specification, interaction planning in collaborative tasks is also challenging. The large problem size of real-world collaborative tasks limits the use of offline planning techniques, while the limited planning time available during interaction makes the online computation of robot decisions difficult. Resolution of this problem is presented in Chapter 5, which provides a novel framework for hybrid and semi-supervised specification of robot's interaction policy for human-robot collaboration.

3.4 Deciding to Communicate during Collaborative Task Execution

The first two problems focused on the information transfer from the developer to the collaborative robot for the purposes of specifying the human teammate model and the robot's interaction policy. This information transfer occurs prior to task execution. In contrast, in the third problem (depicted in Fig. 3-5), I consider the opportunity of effective information sharing between the teammates (i.e., the collaborative robot and the human teammate) during task execution.

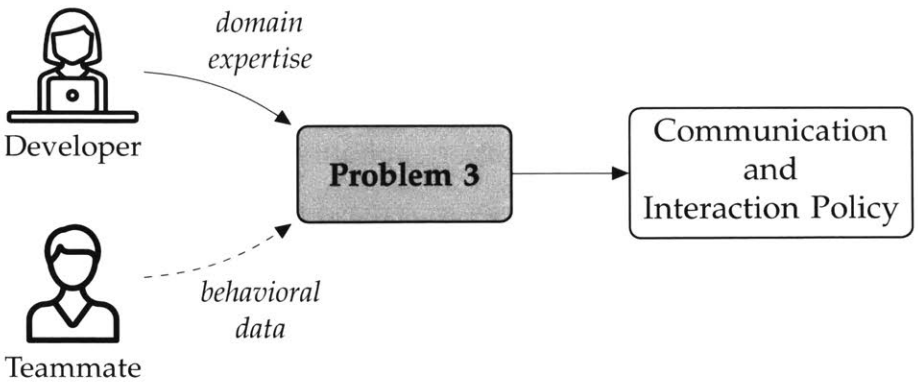

Figure 3-5: Developing an execution-time communication policy for fluent human-robot collaboration.

During task execution, communication between teammates can be used to improve coordination by sharing information about each other and the environment. However, as motivated in Sec. 2.5, too much or too little communication between the human and the robot is undesirable, and execution-time approaches for effective information sharing are needed.

3.4.1 Background

As both the actions and communications of the robot can influence the human teammate's behavior and task performance, in the third problem, I seek the development of approaches that can jointly generate the robot's interaction and communication policy. Such decision-making for the communication between a collaborative robot and a human teammate presents several modeling and algorithmic challenges [157], primarily arising due to

- challenges in quantifying the cost of communication,
- challenges in modeling the human teammate,
- difficulty in estimating the effect and benefits of communication,
- inherent decentralized-nature of multi-agent task, and
- the need for execution-time communication decisions.

In order to focus on the questions of *if, when, and what* to communicate, the problem assumes the presence of a communication modality (such as speech or text) with developer-specified symbol grounding. For achieving effective information sharing, in addition to the communication modality, novel models of human-robot communication cost and frameworks that reason about human's behavior during robot decision-making are required.

Similar to the first two problems, the challenge of modeling human behavior needs to be addressed by fusing behavioral data and partial specifications. However, in order to enable communication decision-making, the teammate model should not only capture human's task execution but also the human's communications (e.g., response to the robot's communications).

3.4.2 Problem Description

Inputs The inputs for making execution-time communication decisions include:

- a specification of the human-robot collaborative task provided using the task model described in Sec. 3.1.2;
- a communication modality with developer-specified symbol grounding;
- an unsupervised behavioral dataset comprising of the human teammate's observable decision factors (s_H) and decisions (a_H);
- dynamics of the observable decision factors; and
- (optionally) partial specifications of the teammate's behavior.

Task Characteristics Both the human and the robot have complete observability of the task state. Further, the dynamics of the task state and the shared reward are known to the teammate and the developer.

Problem Statement Given the inputs, arrive at the collaborative robot's interaction and communication policy that accrues the maximum expected cumulative reward during human-robot collaborative task execution.

I address the third problem in Chapter 6, which provides a novel model of communication cost and a hybrid framework to generate policies for effective information sharing during human-robot collaboration.

3.5 Summary

In this chapter, I formalized the three opportunities for effective information sharing as problems of learning with hybrid inputs, interaction planning, and communication decision-making, respectively. The presence of these opportunities and formulation of the corresponding problems highlight the need for and benefits of effective information sharing between the robot developer, collaborative robot, and human teammate for success of human-machine collaboration. The modeling and algorithmic solutions presented in the subsequent chapters demonstrate that the effective sharing of information is not only important but also feasible.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Modeling Other Agents with Partial Specifications of Behavior

In this chapter, I present novel representations and algorithms that address Problem 1, i.e., learning models of other agent's behavior by utilizing both unsupervised behavioral data and partial specifications of behavior (see Section 3.2.2). Both the model representation and learning algorithm have been designed to facilitate utilization of domain expertise (when available).

I begin with a description of a novel model, termed as the *Agent Markov Model* (AMM), to describe the decision-making behavior of an agent performing sequential tasks. The AMM incorporates a factored state representation to represent latent and dynamic decision factors. The model structure further facilitates encoding the domain knowledge of the other agent's known decision factors, their dynamics, and their impact on the policy.

I adopt a Bayesian approach and provide supervised, semi-supervised, and unsupervised algorithms for learning the AMM from behavioral data. In order to accelerate learning with partial specifications, I introduce Constrained Variational Inference (CVI), a learning paradigm that incorporates auxiliary inputs as constraints during the learning process. I evaluate the performance of the algorithms and their ability to incorporate partial specifications in experiments, and demonstrate improvements in alignment between the true and the learned model.

4.1 Models of Sequential Decision-Making

In order to formalize problem of learning models of other agents, I provide a model of decision-making informed by the presence of latent decision factors and bounded rationality. Several representations, originating from varied modeling and application requirements, have been proposed and analyzed in prior research [5]. Due to our focus on sequential decision-making, I adopt a representation inspired by controlled Markov chains, i.e., Markov chains with control inputs [75].

In order to minimize ambiguity, in this chapter, I refer to the agent who seeks to learn the decision-making model as the observer, while the other agent as the decision maker. While developing collaborative robots, the observer corresponds to the robot developer who is encoding the model of the human teammate (the decision maker) in the collaborative robot's mental model.

4.1.1 Controlled Markov Chain (CMC)

Briefly, a controlled Markov chain models the impact of an input on the sequential evolution of a random variable[1]. I denote the random variable of interest, also referred to as the state of the Markov chain, as $f \in F$. The input to the Markov chain, also referred to as the decision or action, is denoted as $a \in A$. Due to the Markov property, the distribution of the next state given the entire history depends only on the current state and action – i.e.,

$$T_f \equiv \Pr(f_{t+1}|f_t, a_t) = \Pr(f_{t+1}|f_{0:t}, a_{0:t}). \tag{4.1}$$

For a stationary CMC, both the state transition probabilities T_f and the initial state probability $b_f \equiv \Pr(f_0)$ remain constant over time. The model parameters (F, A, b_f, T_f) and the state-action pair (f_t, a_t), completely specify the distribution of the next state f_{t+1}.

[1] **A Note on Notation used in Chapter 4** The variables discussed in this chapter correspond to that of the decision maker (e.g., the human teammate). In Chapter 3, I use the subscript A to denote AMM-specific variables. However, for ease of exposition, the analysis presented in this chapter does not explicitly include this subscript. Thus, $a \equiv a_A$, $f \equiv f_A$, and so on.

4.1.2 Agent Markov Model (AMM)

In order to model sequential decision-making behavior, I build upon the CMC. The decision factors of an agent are modeled as the state of a factored CMC, $f \equiv [x, s]$. The decision-maker can observe both x and s. However, only some decision factors are known to the observer (denoted as known states $s \in S$), while others are unspecified (representing latent/mental states $x \in X$). This implies that, potentially, neither the variable x nor the set X may be known to the observer. Note that $F = X \times S$.

The dynamics of the known and latent decision factors are denoted as T_s and T_x, respectively. Since the mental states x can impact the known states s only via the agent's actions, the transition probabilities have the following factored structure:

$$T_f(f_{t+1}|f_t, a_t) = T_s(s_{t+1}|s_t, a_t)T_x(x_{t+1}|x_t, s_t, a_t). \tag{4.2}$$

In order to model an agent's decision-making, we additionally require a mapping from decision factors to actions (i.e., policy). I model the decision maker to follow a stationary Markov policy: $\pi \equiv \Pr(a_t|f_t) = \Pr(a_t|f_{0:t})$. Further, the initial probability distribution of the unknown decision factor is denoted as b_x, and the observable component of the initial state as s_0. I term this generative model of sequential decision-making behavior, parametrized by the tuple $(X, S, A, b_x, T_x, T_s, \pi)$, as the Agent Markov Model (AMM).

Example As an example of a behavior modeled via the AMM, consider an agent navigating a grid world (e.g., the human in Fig. 3-2). To an observer, the agent's position is observable and thus is the known state, s, while its goal is the unspecified latent state, x. Consequently, both the number of goals, $|X|$, and their dynamics (the way in which the next goal is chosen), T_x, are unknown. While both (x, s) impact the agent's choice of action (the direction in which the agent moves), the next position depends only on the current position and the action. Thus, the transition probabilities exhibit the factored structure of Eq. 4.2.

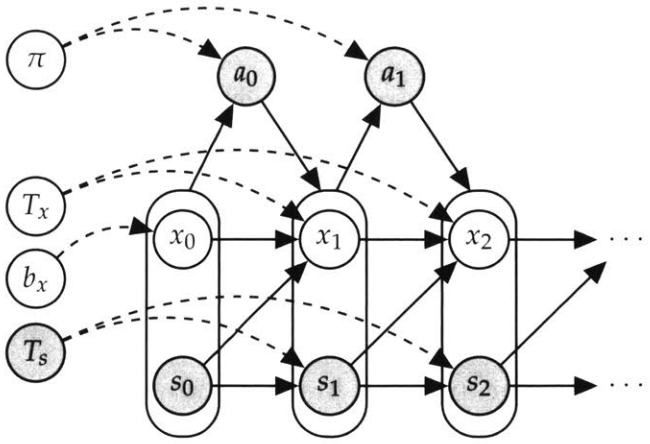

Figure 4-1: The Agent Markov Model (AMM)

Figure 4-1 provides a graphical representation of the AMM. Each node in the figure represents a random variable, and the observed variables are shown in gray. Decision factors s and x (included in the oval supernode) impact the decision choices a, based on the policy $\pi(a|s,x)$. The global variables T_x and T_s specify the probability for the next state given the previous state and the action, and b_x models the initial probability.

4.1.3 Related Models

The AMM shares properties with existing models for sequential decision-making. However, it also includes specific features for incorporating domain knowledge and facilitating model alignment.

Finite State Controllers Recently, Panella and Gmytrasiewicz used probabilistic deterministic finite state controllers (PDFCs) to model the behavior of another agent. A PDFC models the state transition as deterministic, and thus can be derived as a special case of the AMM with deterministic T_f. Further, in contrast with the PDFC, the AMM uses a factored representation for the state variable, which allows the model to incorporate known dynamics of the state (via T_s), and prior knowledge regarding the impact of the known state on the agent's policy (through probabilistic priors for π).

Markov Decision Processes Inclusion of a reward function R within the AMM describes an agent executing policy π in a factored Markov decision process (MDP) given by the tuple (F, A, T_f, R) [115]. Thus, the AMM subsumes the models utilized for inverse reinforcement learning. Since agents may not behave rationally [64], by directly representing the policy to describe decision-making, the AMM does not assume rationality or even goal-oriented behavior on the part of the other agent.

Partially Observable MDPs The AMM and partially observable Markov decision processes (POMDPs) are also related but serve different purposes. POMDPs are used by a decision-maker agent to arrive at her policy (π). In contrast, the AMM is designed for an observer agent seeking to infer and explain the decision-maker's policy (π) by observing her behavior. Further, the agent *solving* a POMDP to arrive at π may not observe the full state but has complete knowledge of the state space (S, X). In the AMM, however, a subset of state space (X) is not only unobservable but may also be unknown.

4.2 Problem Statement

Section 3.2.2 describes the problem of learning the behavior model of another agent performing a sequential task. In this section, I recast this problem in the context of learning an AMM. Based on the availability of the auxiliary inputs (partial specifications of behavior), a taxonomy for different versions of the problem statement is also discussed.

Consider an agent whose true behavior model is given by the AMM tuple $(X, S, A, b_x, T_x, T_s, \pi)$. An observer seeks to recover this true model, but has partial specification of the agent's behavior, described as follows:

- knowledge of the set of observable decision factors, $s \in S$;
- knowledge of the set of agent's actions, $a \in A$;
- dynamics of the observable decision factors, T_s;

- an unsupervised behavioral dataset – i.e., I execution traces of the agent's behavior, where the i-th trace refers to the sequences $(s^i_{0:N_i}, a^i_{0:N_i})$;
- (optionally) knowledge of the latent decision factors, $x \in X$;
- (optionally) labels of the latent decision factor for the complete or part of the unsupervised behavioral dataset;
- (optionally) noisy change-point information regarding the latent state – i.e., the indicator variable $c_t = \mathbb{I}(x_{t+1} = x_t)$ which is accurate with probability p_c;
- (optionally) linear equality or inequality constraints regarding some elements of the agent's policy (i.e., partial knowledge of π) and the transition function of latent states (i.e., partial knowledge of T_x).

Thus, formally, the problem of learning the decision-making model corresponds to learning the full AMM tuple given the partial tuple $(\cdot, S, A, \cdot, \cdot, T_s, \cdot)$, unsupervised behavioral data (i.e., the execution traces), local auxiliary input (i.e., the labels and change-point of latent decision factors), and global auxiliary input (i.e., partial knowledge of transition function and policy).

4.2.1 Availability of Auxiliary Inputs

The motivation behind and detailed description of the various types of inputs are presented in Section 3.2. Here, through example applications, I discuss the availability, or lack thereof, of the auxiliary input while modeling humans.

Knowledge of Decision Factors The behavior of humans depends on a variety of mental states, such as belief, attention, workload, arousal, and trust. Several fields of study, such as cognitive science, psychology, and human factors, are actively exploring research on recovering these latent states and understanding their impact on human behavior. Thus, in general, while modeling the human teammate, it is difficult to specify all the decision factors that affect her behavior. In parlance of machine learning, this challenge of identifying the decision factors is akin to that of feature discovery. However, depending on the modeling context, the robot de-

veloper may be able to specify the critical decision factors for a given task. For example, in the grid-based example introduced in Section 4.1.2, we have already discussed the specification of the known states (as the agent's position) and latent states (as the agent's goal).

Availability of Labeled Data Typically, the latent decision factors, when known, correspond to a human's mental states. Thus, even when the decision factors are known, it might be challenging to obtain its labels for multiple reasons. First, it is difficult to suitably quantify cognitive variables, such as attention or trust. In order to overcome this challenge, often a developer-defined scale is used for quantifying the latent state. Second, even when a scale exists, the process of obtaining the labels or change-points of the latent states is either intrusive, effort-intensive, or both. For instance, one approach is to ask users to answer questionnaires regarding their mental states during task execution. Another approach involves the user labeling a recording of their interaction post task execution. However, in certain modeling context, the robot developer may be able to label the critical decision factors for a given task using domain expertise or algorithms. Despite the option of acquiring labels of the latent states in certain domains, in order to reduce the developer's effort, approaches that require no or fewer labels are preferred.

Availability of Domain Expertise The developer typically does not have complete knowledge of a human's policy or evolution of the mental states. However, often partial knowledge of the human's behavior is available based on prior experience or domain expertise. For instance, in the grid-world example, the number of goals, $|X|$, or their possible locations may be known to the developer. Such information constrains the dynamics of the latent state and partially specifies the transition function T_x. Furthermore, the goal is but one example of the latent state; in other applications, where the latent state may correspond to other difficult-to-quantify variables, such as workload and attention, constraints regarding their dynamics can be obtained based on cognitive theories and models [114].

	Problem Variant					
	Parametric			Nonparametric		
Inputs	Sup.	Semi-Sup.	Unsup.	Sup.	Semi-Sup.	Unsup.
S, A, T_s	known					
X	known			unknown		
x-labels	full	partial	none	n/a	n/a	n/a
c-labels	n/a	partial	none	n/a	partial	none
partial π, T_x	if yes, then denoted as hybrid learning					

Table 4.1: A taxonomy of problem variants for learning the AMM

4.2.2 Taxonomy of Problem Variants

Depending on the availability of the auxiliary inputs, the problem of learning the model of other agent corresponds to different paradigms of learning. I discuss the variants of interest next, and summarize them in Table 4.1.

First, I classify the variants based on the specification of the decision factors:

- **Parametric AMM Learning:** The set of all decision factors (i.e., both observable S and latent X) is known.
- **Nonparametric AMM Learning:** The set of observable decision factors (S) is known, but the set of latent decision factors (X) is unknown.

Next, depending on the availability of local auxiliary inputs (i.e., labels and change-points of latent decision factors), the learning corresponds to either:

- **Supervised AMM Learning:** The labels of the latent state x are available for the entire behavioral dataset. Note that supervised learning is not possible for the case of nonparametric learning, since X (the state space of x) is unknown.
- **Semi-Supervised AMM Learning:** The labels or change-points of the latent state x are available, but only for a subset of the behavioral dataset. Semi-supervised learning is possible for both the parametric and the nonparametric case. However, in the nonparametric case, labels of x cannot be obtained (as X is unknown).

- **Unsupervised AMM Learning:** No labels or change-points are available for the latent state x. Unsupervised learning is possible for both the parametric and nonparametric case.

Finally, irrespective of availability of other inputs (i.e., labels, change-points, or knowledge of decision factors), **Hybrid AMM Learning** refers to the case when partial knowledge of the latent state transition or policy is available. Thus, hybrid learning of AMM can be parametric or nonparametric as well as supervised, unsupervised, or semi-supervised. Next, I provide solutions for these problem variants of learning the AMM. I begin by considering nonparametric, unsupervised learning, i.e., the problem variant with least input information. I augment this solution to consider other input types and arrive at solutions for other problem variants.

4.3 Unsupervised AMM Learning with Nonparametric Priors

I adopt a Bayesian approach to recover the AMM parameters given the partial AMM tuple $(\cdot, S, A, \cdot, \cdot, T_s, \cdot)$ and the unsupervised behavioral dataset. I view the unknown model parameters and state sequences as latent random variables, and seek to infer their posterior using the data. I limit the scope to those AMMs in which x is a scalar, i.e., only one of the decision factors is latent.

4.3.1 Prior Distributions

The AMM tuple provides a generative model for an agent's behavior; however, since the tuple is only partially known, I include priors for the unknown AMM parameters in order to compute their posterior. In the nonparametric case, as the set of unknown factors X is unknown a priori, the number of factors x is also unknown. This motivates the use of nonparametric priors for the number of unknown states $n_x = |X|$, initial distribution b_x and transition function T_x.

I use hierarchical Dirichlet process (HDP) priors inspired by the infinite HMM model [147]. Under this HDP prior, the popularity of the latent states is generated via a Dirichlet process (DP). The base distribution over possible latent states, given by β, can be obtained using a stick-breaking construction with hyper-parameter γ, i.e.,

$$\beta(\cdot) \sim \text{GEM}(\gamma) \tag{4.3}$$

$$T_x(\cdot|x,s,a) \sim \text{DP}(\alpha \cdot \beta), \quad \forall (s,a) \text{ and } x = 1,2,\cdots \tag{4.4}$$

The rows of the transition function T_x are also generated using a Dirichlet process (DP) with base distribution β and scaling parameter α. This allows the model to share parameters (in this case, the state space of latent states) between the rows of the transition function. The latent states are indexed as positive integers. A similar process is used for b_x.

In addition to the priors for the latent states and their dynamics, the model includes a prior for the policy $\pi(a|s,x)$. This prior is modeled as a probability distribution over the decision-maker's action space and is identical for each of the potentially countably infinite latent states x. This policy prior allows the developer to specify prior domain knowledge (if any) regarding the agent's policy. Further, the policy prior can vary based on the known state s. In absence of additional knowledge, a Dirichlet distribution can serve as the policy prior,

$$\pi(a|s,x) \sim \text{DIRICHLET}(\rho_{1:|A|}), \quad \forall (s) \text{ and } x = 1,2,\cdots \tag{4.5}$$

Infinite AMM (iAMM) The graphical model of Fig. 4-1, along with the priors specified in Eq.4.3-4.5, provides a generative model for the AMM. In line with other nonparametric models, I term this model as the infinite AMM (iAMM). The iAMM incorporates a factorization structure common to several statistical models (cf. Eq. 1 Hoffman et al.), which includes global variables (T_x, b_x, π, β), local hidden variables $(x_{0:N})$, observations ($s_{0:N}$ and $a_{0:N}$) and hyper-parameters (α, γ, ρ).

4.3.2 Variational Inference with Execution Traces

In this section, I derive a algorithm for the case of execution traces as the training data. In later sections, I augment the algorithm to incorporate other input types and arrive at solutions to other problem variants for learning the AMM.

In general, obtaining the exact posterior distribution of the iAMM is intractable. Hence, I explore approaches that approximate this distribution. Both optimization-based (e.g., variational inference) and sampling-based (e.g., Markov chain Monte Carlo methods) paradigms can be developed for learning the posterior [154, 155]. Guided by the recent success of variational inference (VI) for sequential models with factorization structures similar to the iAMM [61], here, I provide a mean-field VI algorithm in order to approximate the posterior distribution the iAMM. A sampling-based method for parametric models is presented in the next section.

In the current setting, the training data consists of (s, a) traces from I sequences: $\mathbf{x}=\{x^i_{0:N_i}\}^I$, $\mathbf{s}=\{s^i_{0:N_i}\}^I$, $\mathbf{a}=\{a^i_{0:N_i}\}^I$. The posterior distribution of the hidden variables of the iAMM is then denoted as $p(\mathbf{x}, T_x, b_x, \pi, \beta|\mathbf{s}, \mathbf{a})$. Mean-field inference approximates this posterior by the product of variational factors $q(\mathbf{x})q(T_x)q(b_x)q(\pi)q(\beta)$ – i.e., by assuming independence between the hidden variables. Each variational factor is a distribution with separate parameters. The posterior is obtained as the arg max of the evidence lower bound $\mathcal{L}$;

$$\mathcal{L}(q) \equiv \mathbb{E}_q\left[\frac{p(\mathbf{x}, T_x, b_x, \pi, \beta|\mathbf{s}, \mathbf{a})}{q(\mathbf{x})q(T_x)q(b_x)q(\pi)q(\beta)}\right] \tag{4.6}$$

Due to the mean-field assumption, this optimization problem can be solved by iteratively optimizing and updating the parameters of the local $q(\mathbf{x})$ and global $q(T_x), q(b_x), q(\pi), q(\beta)$ variational factors. Following Hoffman et al., I use the natural gradient ascent optimization for the global variational updates [6]. While the structure of the algorithm is similar to that of the SVI algorithm for the iHMM [61], the variational factors and update equations differ. Here, I include the key terms for inference of the iAMM; for an excellent introduction to variational inference, I refer the reader to [55].

Local Variational Factors

In order to efficiently estimate the latent state sequences in the presence of a non-parametric prior, I utilize the direct assignment truncation [61]. This inference approach requires a truncation parameter K as input and models the assumption that K latent states at most are present in the execution traces, i.e., $q(\mathbf{x})=0$ if any $x > K$. The mean-field update for the local variational factors is given as follows:

$$q(\mathbf{x}) \propto \exp(\mathbb{E}_q[\ln p(\mathbf{x}, \text{data}|T_x, b_x, \pi, \beta, T_s)]) = \prod_i q(x^i) \tag{4.7}$$

$$\begin{aligned} q(x^i) &\propto \exp\left(\mathbb{E}_q[\ln p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|T_x, b_x, \pi, \beta, T_s)]\right) \\ &\propto \exp\left(\mathbb{E}_q[\ln p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|T_x, b_x, \pi, T_s)]\right) \\ &= p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|\tilde{T}_x, \tilde{b}_x, \tilde{\pi}, T_s)/Z^i \\ &\propto \exp\Bigg(\ln \tilde{b}_x(x^i_0) + \sum_0^{N_i-1} \big(\ln \tilde{T}_x(x^i_{t+1}|x^i_t, s^i_t, a^i_t) + \\ &\qquad \ln T_s(s^i_{t+1}|s^i_t, a^i_t) + \ln \tilde{\pi}(a^i_t|x^i_t, s^i_t)\big)\Bigg) \end{aligned} \tag{4.8}$$

The tilde denotes the operator $\tilde{A}=\exp(\mathbb{E}_{q(A)}[\ln A])$ and is used to provide expectation with respect to the global variational actors. The variable Z^i denotes the normalization constant. In order to compute the normalization constant and given their utility in updating global factors, I compute the forward F and backward messages B for each sequence. The message computation takes $\mathcal{O}(NK^2)$ time.

$$F_{(t,j)} \equiv \Pr(x_t=j, s_{0:t}, a_{0:t}) \tag{4.9}$$

$$= \sum_k \Big(F_{(t-1,k)} \tilde{T}_x(j|k, s_{t-1}, a_{t-1}) T_s(s_t|s_{t-1}, a_{t-1}) \tilde{\pi}(a_t|j, s_t)\Big) \tag{4.10}$$

$$B_{(t,j)} \equiv \Pr(s_{t+1:N}, a_{t+1:N}|x_t=j, s_{0:t}, a_{0:t}) \tag{4.11}$$

$$= \sum_k \Big(B_{(t+1,k)} \tilde{T}_x(k|j, s_t, a_t) T_s(s_{t+1}|s_t, a_t) \tilde{\pi}(a_{t+1}|k, s_{t+1})\Big) \tag{4.12}$$

$$F_{(0,j)} = \tilde{b}_x(j)\tilde{\pi}(a_0|j, s_0) \qquad B_{(N,j)} = 1, \tag{4.13}$$

$$Z = \sum_j F_{(N,j)}. \tag{4.14}$$

Global Variational Factors

The direct assignment truncation allows us to represent the base distribution as $\beta = (\beta_{1:K}, \beta_{rest})$, where $\beta_{1:K}$ represents the probability of the first K states and $\beta_{rest} \equiv 1 - \sum_{k=1}^{K} \beta_k$ represents the probability of truncated states. Further, it results in a Dirichlet prior for the rows of T_x and b_x given as $\text{DIR}(\alpha \cdot (\beta_{1:K}, \beta_{rest}))$. Due to the conjugacy of the Dirichlet and Multinomial distributions, the variational factors (approximate posterior) for each row of T_x are given as follows:

$$q(T_x(\cdot|x = j, s = s, a = a)) = \text{DIRICHLET}(\lambda_{jsa}) \tag{4.15}$$

In order to update the global parameters, expected statistics are necessary with respect to the local factor $q(\mathbf{x})$. For the transition function this corresponds to the expected transition counts which can be efficiently computed using the forward and backward messages as follows:

$$\hat{u}^{T_x}_{kjsa} \equiv \mathbb{E}_{q(\mathbf{x})} \sum_t \mathbb{I}[x_{t+1}{=}k, x_t{=}j, s_t{=}s, a_t{=}a] \tag{4.16}$$

$$= \sum_t F_{(t,j)} T_s(s_{t+1}|s, a) \tilde{T}_x(k|j, s, a) \tilde{\pi}(a_{t+1}|k, s') B_{(t+1,k)} / Z \tag{4.17}$$

Given the expected transition counts and conjugacy, the global variational parameters are updated by maximizing the evidence lower bound (ELBO),

$$\lambda_{jsa} = \arg\max_{\lambda} \mathcal{L}(\lambda; \mathbb{E}_q[\eta_g]{=}(\alpha \cdot \beta + \hat{u}^{T_x}_{\cdot jsa})) \tag{4.18}$$

$$\mathcal{L}(\lambda) \equiv (\mathbb{E}_q[\eta_g] - \lambda) \cdot \nabla_\lambda \ln B(\lambda) + \ln B(\lambda) + \text{const.} \tag{4.19}$$

$B(\lambda)$ represents the multivariate Beta function, and $\ln B(\lambda)$ is the log normalizer of the Dirichlet distribution (cf. Eq. 13 Hoffman et al.). This optimization problem is solved by equating the natural gradient [6] of the objective function to zero, resulting in the update:

$$\lambda_{jsa} \leftarrow (\alpha \cdot \beta + \hat{u}^{T_x}_{\cdot jsa}) \tag{4.20}$$

This update in the variational parameter λ_{jsa} corresponds to a natural gradient ascent with step size, ν, of one. For large datasets, stochastic natural gradient ascent can be used by considering only a subset of the dataset (local variational factors) when computing the expected statistics, i.e.,

$$\lambda_{jsa} \leftarrow \nu_t(\alpha \cdot \beta + \hat{u}^{T_x}_{\cdot jsa}) + (1 - \nu_t)\lambda_{jsa} \tag{4.21}$$

with the sequence of step size ν_t satisfying the properties: $\sum_t \nu_t = \infty$, $\sum_t \nu_t^2 < \infty$.

Due to the presence of conjugate priors, analysis similar to $q(T_x)$ follows for updating $q(b_x)$ and $q(\pi)$. However, different expected statistics are required, which are computed as follows:

$$\hat{u}^{b_x}_j \equiv \mathbb{E}_{q(\mathbf{x})}\mathbb{I}[x_0{=}j] = B_{0,j}/Z \tag{4.22}$$

$$\hat{u}^{\pi}_{ajs} \equiv \mathbb{E}_{q(\mathbf{x})}\textstyle\sum_t \mathbb{I}[a_t{=}a, x_t{=}j, s_t{=}s] \tag{4.23}$$

$$= \textstyle\sum_t F_{t,j} B_{t,j} \mathbb{I}[a_t{=}a, s_t{=}s]/Z \tag{4.24}$$

In order to obtain a variational estimate for β, following [61], a point estimate β^* is obtained by minimizing the evidence lower bound with the constraint that all elements are non-negative.

Mean-field Variational Inference

Given the nonparametric priors and the variational factors, we arrive at a nonparametric approach to learning the AMM, summarized in Algorithm 4.1. Along with data, the inference requires as inputs specification of the prior distributions (i.e., hyper-parameters). The inference procedure iteratively updates the local and global variational factors, until the evidence lower bound converges.

The output of the inference algorithm is a distribution over AMMs. In applications, if need be, a single model can be arrived at by sampling from the distribution or by selecting its mode. We revisit nonparametric learning for the semi-supervised case in Sec. 4.6.2.

Algorithm 4.1: Mean-field variational inference for the infinite AMM

Data: N seqs. of (s, a)-tuples
Parameters: Partial AMM tuple (S, A, T_s), and hyper-parameters $(\alpha, \gamma, \rho, K)$
Result: Posterior for π, T_x, b_x, and N seqs. of x

```
Initialize the global variational parameters
repeat
    for each execution trace in the dataset do
        Update the local variational parameters
    end
    Update the global variational parameters
until the ELBO converges
```

4.4 Unsupervised AMM Learning with Parametric Priors

In this section, I provide an algorithm for parametric, unsupervised AMM learning. In contrast to nonparametric learning, the additional input available in this sections is the knowledge of the state space of the latent decision factors X. Thus, I update the prior distributions to encode the knowledge of X and arrive at *Bayesian AMM*, a parametric generative model for the AMM.

I provide both sampling-based and optimization-based algorithms for learning the Bayesian AMM. The variational inference algorithm is derived as a special case of the algorithm presented in Algorithm 4.1. Gibbs sampling is used to derive the sampling-based algorithm, which could be used to seed the optimization routine of variational inference.

4.4.1 Prior Distributions

Similar to nonparametric AMM learning, I pose the problem of recovering AMM parameters as one of Bayesian learning. I first specify the priors for unknown model parameters (π_A, T_x, b_x) and infer their posterior $Pr(\pi_A, T_x, b_x|\cdot)$ given the inputs for learning, namely, partial AMM tuple (S, X, A, T_s) and the unsupervised behavioral dataset.

In this case, as the set of latent state is a known parameter, I utilize the Dirichlet distribution (as opposed to the Dirichlet process) to arrive at the priors for the latent state distributions (T_x, b_x). Specifically, I use the following parametric priors for the latent state distributions,

$$b_x(\cdot) \sim \text{DIRICHLET}(\alpha) \tag{4.25}$$

$$T_x(\cdot|x,s,a) \sim \text{DIRICHLET}(\alpha), \quad \forall(s,a) \text{ and } x = 1,2,\cdots,|X| \tag{4.26}$$

where α is a hyper-parameter; in applications, different hyper-parameters can be used for different rows of the transition matrix and the initial distribution.

The choice of the distribution for policy priors depends on the domain. Similar to the nonparametric model, in absence of additional domain knowledge, I use the Dirichlet distribution as policy priors with hyper-parameter ρ,

$$\pi(a|s,x) \sim \text{DIRICHLET}(\rho_{1:|A|}), \quad \forall(s) \text{ and } x = 1,2,\cdots,|X| \tag{4.27}$$

However, in contrast to the iAMM, the number of policy distributions are finite.

Bayesian AMM (iAMM) The graphical model of Fig. 4-1, combined with the priors specified in Eq.4.25-4.27, provides a parametric generative model for the AMM. Analogous to other parametric Markov models for sequential data, I term this model as the Bayesian AMM.

4.4.2 Blocked Gibbs Sampler

In practice, the performance of variational inference can be improved by providing a suitable initial guess. Thus, first, I provide a blocked Gibbs sampler for the Bayesian AMM, summarized as Algorithm 4.2. The Gibbs sampler begins with an initial guess for the unknown latent state (x_A) sequences. In absence of any additional information, the guess is randomly generated. Next, using the initial guess and unsupervised data of behavior, the algorithm samples the unknown global

Algorithm 4.2: Blocked Gibbs sampler for the Bayesian AMM

Data: N seqs. of (s_A, a_A)-tuples
Parameters: Partial AMM tuple (S, X, A, T_s), and hyper-parameters (α, ρ)
Result: Samples for π_A, T_x, b_x, and N seqs. of x_A

Initialize π_A, T_x, b_x, and x_A randomly
repeat
  Sample initial distribution, $Pr(b_x|\mathbf{x_A}, \text{data}; \alpha)$
  Sample transition model, $Pr(T_x|\mathbf{x_A}, \text{data}; \alpha)$
  Sample policy, $Pr(\pi_H|\mathbf{x_A}, \text{data}; \rho)$
  Sample latent states, $Pr(\mathbf{x_A}|\text{data}, T_x, b_x, \pi_H)$
until the desired number of samples are generated

parameters of the AMM. The AMM parameters are then used to sample the latent state sequences. This procedure is repeated iteratively to generate successive samples, until the desired number of samples are generated.

Sampling of model parameters (π_A, T_x, b_x), utilizes the counts of inferred latent state sequences $\mathbf{x_A}$. For sampling T_x, I define n_{isaj} as the count of transition from latent state $x_A = i$ to latent state $x'_A = j$ for action a_A and observed state s_A. Both i and j range from 1 to $|X_A|$. The conditional distribution of T_x is given as: $T_x(\cdot|x_A, s_A, a_A) \sim \text{Dirichlet}(n_{xsa\cdot} + \alpha_t)$. The sampling of initial distribution follows similarly, and depends on the initial counts of latent states in the data. Conditional distributions for π_H depend upon the policy prior. Latent state sequences are sampled using a variant of forward filtering-backward sampling (FFBS) algorithm derived for the AMM.

4.4.3 Variational Inference with Execution Traces

In the variational inference algorithm for iAMM, I utilize direct assignment truncation to estimate latent state sequences with nonparametric priors. In this approximation, the algorithm requires as input K, i.e., an upper bound on the number of latent states $|X|$. However, for the Bayesian AMM the number of latent states $|X|$ is known as an input for model learning. Thus, the mean-field variational inference algorithm for parametric, unsupervised AMM learning is obtained as a special case of Algorithm 4.1, with $K = |X|$.

The computation of the local variational factors (Eq. 4.8-4.17) remains identical to that of learning with nonparametric priors. However, since the T_x-prior is a Dirichlet distribution (and not a Dirichlet process), the global update equations (Eq. 4.18) are modified as follows,

$$\lambda_{jsa} = \arg\max_{\lambda} \mathcal{L}\big(\lambda; \mathbb{E}_q[\eta_g] = (\alpha + \hat{u}^{T_x}_{\cdot jsa})\big) \tag{4.28}$$

$$\lambda_{jsa} \leftarrow (\alpha + \hat{u}^{T_x}_{\cdot jsa}) \tag{4.29}$$

Further, β is absent in the parametric model, alleviating any need to compute its posterior. Finally, the computation of policy posterior also remains identical to that of the unsupervised AMM learning with Bayesian nonparametric priors.

4.5 Supervised AMM Learning

For supervised learning of the Bayesian AMM, in addition to the inputs of unsupervised learning, labels of the latent state x are available for the entire behavioral dataset. The inference of the posterior, given the labels, is simplified and obtained trivially as a special case of the parametric, unsupervised AMM learning.

For the blocked Gibbs sampler (Algorithm 4.2), due to the availability of labels, the step required for sampling the latent states (line 6) is not required. The posterior inference of the latent model parameters (b_x, T_x, π), however, remains identical. Similarly, for mean-field variational inference, the computation of local variational factors is unnecessary given the state labels. Thus, a variational inference algorithm for supervised learning of AMM is obtained by skipping lines 3-5 in Algorithm 4.1, and by specifying the hyper-parameter $K = |X|$.

The problem setup of supervised AMM learning is closely related to that of inverse reinforcement learning [118, 173]; however, without any assumption of the agent's rationality. By augmenting the AMM with a generative process for the policy π that depends on a latent random variable corresponding to reward, algorithms for Bayesian inverse reinforcement learning can be derived analogously.

4.6 Semi-Supervised AMM Learning

The solutions presented so far either assume no or complete knowledge of the other agent's latent decision factors. However, as motivated in Sec. 2.3, often partial specifications of behavior are available while modeling other agents. In this section, I provide semi-supervised learning approaches that can incorporate local auxiliary inputs, i.e., specifications that are specific to a particular behavioral dataset. I first discuss the case where the developer specifies labels only for a subset of the behavioral dataset; such an approach is useful, as it can reduce the developer's labeling effort while specifying human teammate models. Next, I augment the AMM to enable learning with the change-points.

4.6.1 Learning with Partial Labels

Analogous to supervised AMM learning, an approach for learning with partial labels can be arrived as a special case of parametric, unsupervised AMM learning. Here, I consider the case where in addition to the inputs for unsupervised AMM learning, the developer provides labels for a subset of the execution traces. The sampling-based and variational inference algorithms, then, can be derived by computing the posterior only for the unlabeled subset of the execution traces.

4.6.2 Learning with Change-Points

Change-points are a useful source of information as they can help learn the dynamics of the latent state, and may be queried even when it is difficult to accurately quantify the latent state. For instance, while modeling the goal-directed motion of an agent, the set of all possible goals might be unknown to the developer (i.e., X); however, the developer may be able to specify change-points of goals for a specific execution trace. Thus, the change-point information may be available during both the parametric and nonparametric case of AMM learning.

Likelihood Model

In contrast to learning with partial labels, learning with change-points requires modification to the model. I consider the change-point information, when available, as an additional observation with two levels, 0 and 1, and the likelihood model ψ,

$$\begin{aligned} \psi(c_t{=}1|x_{t+1}{=}x_t) &= \psi(c_t{=}0|x_{t+1}{\neq}x_t) = p_c \\ \psi(c_t{=}1|x_{t+1}{=}x_t) &= \psi(c_t{=}0|x_{t+1}{\neq}x_t) = (1-p_c) \end{aligned} \tag{4.30}$$

where p_c represents the confidence of the domain expert in the provided input. As depicted in Fig. 4-2, the change-point observation depends on both the current and next value of the latent state.

Updates to Variational Factors

The overall structure of the semi-supervised algorithm remains identical to that of Algorithm 4.1; however, the update equations for variational factors are modified due to the availability of additional observations (i.e., change-points). The modified message computations and sufficient statistics u are derived as follows :

$$q(x^i) = p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}, c^i_{0:N}|\tilde{T}_x, \tilde{b}_x, \tilde{\pi}, T_s, \psi) \tag{4.31}$$

$$\begin{aligned} F_{(t,j)} &\equiv \Pr(x_t{=}j, s_{0:t}, a_{0:t}, c_{0:t-1}) \\ &= \textstyle\sum_k F_{(t-1,k)} \tilde{T}_x(\cdot) T_s(\cdot) \tilde{\pi}(\cdot) \psi(c_{t-1}|k,j) \end{aligned} \tag{4.32}$$

$$\begin{aligned} B_{(t,j)} &\equiv \Pr(s_{t+1:N}, a_{t+1:N}, c_{t:N-1}|x_t{=}j, s_{0:t}, a_{0:t}) \\ &= \textstyle\sum_k B_{(t+1,k)} \tilde{T}_x(\cdot) T_s(\cdot) \tilde{\pi}(\cdot) \psi(c_t|j,k) \end{aligned} \tag{4.33}$$

$$\hat{u}^{T_x}_{kjsa} = \textstyle\sum_t F_{(t,j)} T_s(\cdot) \tilde{T}_x(\cdot) \tilde{\pi}(\cdot) B_{(t+1,k)} \psi(c_t|j,k) / Z$$

By utilizing the mean-field approximation, given the modified sufficient statistics, no other changes to the global update equations are necessary.

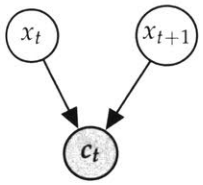

Figure 4-2: Change-points as an observation for the latent states of the AMM.

Differences between the Parametric and Nonparametric Case

The change-point information can be available both with or without knowledge of the latent state space, i.e., both in the the cases of parametric and nonparametric priors. The approach for incorporating change-point information as novel observations, discussed above, applies equally to both the cases, with differences only in the update of global variational factors. The analysis of global variational factors is discussed in Sec. 4.3.2 for the nonparametric case (i.e., when X is unknown or partially known) and in Sec. 4.4.3 for the parametric case (i.e., when X is known).

4.7 Hybrid AMM Learning

Thus far, we have discussed approaches for learning an Agent Markov Model with the partial AMM tuple (S, A, T_s), unsupervised behavioral dataset, and

- without any optional inputs (i.e., nonparametric unsupervised learning);
- with only one optional input, namely, the set of latent decision factors X (i.e., unsupervised learning with parametric priors);
- with the optional inputs: the latent state space X, partially labeled dataset, and change-point information (i.e., semi-supervised learning); and
- with labels of latent state for the entire dataset (i.e., supervised learning).

In addition to the optional inputs listed above, the developer might have domain knowledge which is not considered for either of these methods — namely, the final optional input listed in the problem statement, which corresponds to the partial knowledge of the policy π or transition function T_x. For instance, in an assembly task, not only is the set of a human associate's goals (X) known, often the

(complete or partial) sequence in which the associate accomplishes her goals are also known. Similarly, for modeling behavior of a human driver, policy in some states (such as at a traffic signal) may be reliably known but not at other parts of the road. Since this domain knowledge corresponds to the global parameters of the generative model, I refer to these partial specifications as global auxiliary inputs. In this section, it present a method to incorporate partial knowledge of global variables during model learning and apply it for learning the AMM.

4.7.1 Challenges

Priors and hyper-parameters provide one avenue for incorporating domain knowledge regarding global variables in a Bayesian approach. For instance, while learning the AMM, the knowledge about the other agent's policy and latent state transition function can be incorporated via the policy priors, ρ, and the scaling parameter, α, respectively.

However, the expressivity of the prior is limited, i.e., it is difficult to specify arbitrary knowledge of the random variable through its prior parameters. Further, changes to the prior distribution may require manually updating the inference procedure. This challenge of translating domain knowledge into updates for the inference procedure is further exacerbated for algorithm-users (such as developers) that may not be statisticians or researchers. Finally, while performing inference, these updates may lead to computational challenges (e.g., due to loss of conjugacy).

Example In order to exemplify these challenges, we revisit the case where a developer is modeling the goal-directed motion of a robot's human teammate. The human's goal corresponds to the latent state x and position corresponds to the known state s. Further, there are three possible goals $X = \{i, j, k\}$ and the human moves from one goal to the other to complete his task. Based on her domain knowledge of the human's task, the developer knows that after reaching goal k the human may choose to remain at goal k or with equal likelihood switch to i or j. However, the exact transition probabilities are unknown.

Thus, without loss of generality, we focus on one row of the transition function $\theta \equiv T_x(\cdot|k, s, a)$ for which the developer knows that two elements i and j are equal. Note that each row of the transition function is a Multinomial distribution, with a conjugate Dirichlet prior. The domain knowledge corresponds to $\theta_i = \theta_j$, a constraint for the parameters of the Multinomial distribution θ. In this example, given the domain knowledge, specifying a prior (or even a sampling procedure) for the row of the transition function T_x is non-trivial for the developer.

Thus, while the local auxiliary input can be included by augmenting the model, incorporating global auxiliary input as additional observations or by updating the prior distribution is not straightforward. Further, if the prior distribution of T_x is modified, update to the inference process are necessary. Therefore, a novel inference procedure is necessary that can ideally (a) automatically translate the developer's domain knowledge to an input appropriate for learning, and (b) does not require manual updates to the inference procedure.

4.7.2 Constrained Variational Inference

Guided by these requirements, I develop constrained variational inference (CVI), an approach that facilitates translation of the domain knowledge to inputs for learning and requires minimal updates to the inference procedure. Constrained variational inference consists of the following steps,

- conversion of domain knowledge to constraints on variational parameters; and
- constrained optimization of variational parameters to infer the posterior.

Converting domain knowledge to constraints

First, the available domain knowledge about a global variable θ is converted into a set of constraints for its variational parameter λ. Note that the variational parameter parameterizes the posterior of the corresponding random variable, i.e., $\theta \sim \lambda$. The moments of the posterior distribution are used to derive the constraints, denoted as $g(\lambda) \geq 0$.

Constrained optimization for inference

In order to compute the posterior, CVI solves a constrained optimization problem in which the objective is identical to the evidence lower bound, $\mathcal{L}$, and the constraints are derived from the domain knowledge (i.e., $g(\lambda) \geq 0$). The constrained optimization problem is defined as follows,

$$\lambda = \arg\max_{\lambda} \mathcal{L}(\lambda; e) \quad \text{subject to} \quad g(\lambda) \geq 0 \tag{4.34}$$
$$\mathcal{L}(\lambda; e) \equiv (e - \lambda) \cdot \nabla_{\lambda} \ln B(\lambda) + \ln B(\lambda) + \text{const.}$$

where, e is a shorthand for $\mathbb{E}_q[\eta_g]$, the expected natural parameter of the complete conditional distribution of θ. Due to the presence of constraints, the optimization of the variational parameter is no longer a single-step update (e.g., Eq. 4.18 and Eq. 4.28); instead a constrained optimizer (such as sequential quadratic programming or augmented Lagrangian method) is required to infer λ.

In addition to the constraints and the optimization function described in Eq. 4.34, algorithms for constrained optimization require the optimization function, its gradient, and (optionally) its Hessian. While employing CVI for random variables with posterior distributions in the exponential family, these terms can be obtained analytically. These analytical expressions for the natural gradient and natural Hessian are given as follows,

$$\hat{\nabla}_{\lambda} \mathcal{L}(\lambda; e) = (e - \lambda) \tag{4.35}$$
$$\hat{\nabla}^2_{\lambda} \mathcal{L}(\lambda; e) = -\{\nabla^2_{\lambda} \ln B(\lambda)\}^{-1}, \tag{4.36}$$

where, $\hat{\nabla}_{\lambda}$ denotes the natural gradient, $\hat{\nabla}^2_{\lambda}$ denotes the natural Hessian, and $\ln B(\lambda)$ is the log normalizer of the variational distribution.

By incorporating the auxiliary information as constraints over the variational parameters, the support of the posterior distribution is limited to those regions of variational parameters λ that satisfy the constraints. Thus, CVI functions, in essence, as a weighted prior that is algorithmically generated given the devel-

oper's partial specifications. Furthermore, CVI provides an approximate posterior by limiting the posterior to be in the same family as the prior.

Consequently, a computationally expensive approach that modifies the prior but sacrifices conjugacy can lead to better estimates. However, such an approach would required the developer to manually modify the prior and re-design the inference scheme. Instead, by utilizing constrained optimization, CVI leverages the benefits of conjugacy and allows for approximating the posterior with its conjugate distribution. More importantly, it enables developers to utilize their domain knowledge (available as partial specifications of latent variables) in the learning process, without requiring them to modify the prior or the learning algorithm.

4.7.3 Application of CVI for AMM Learning

For hybrid AMM learning, I focus on incorporating domain knowledge regarding the variables $\{\pi, T_x\}$. In general, these global variables of the AMM are described by a set of Multinomial distributions θ over action and latent state, respectively. The conjugate prior and variational distribution for θ is the Dirichlet distribution, which is parameterized by variational parameters denoted as λ. I provide the approach for converting domain knowledge to variational constraints for the following abstract types of domain knowledge regarding θ,

C1. the ratio of the i-th and j-th elements of θ is known, i.e., $\theta_i = b\theta_j$, resulting in the constraint $g(\lambda)$ for the variational parameter, $\lambda_i = b\lambda_j$;

C2. the value of the i-th element of θ is known, i.e., $\theta_i = b$, resulting in the constraint $g(\lambda)$ for the variational parameter, $\lambda_i = b\sum_j \lambda_j$;

C3. the maximum value of the i-th element of θ is known, i.e., $\theta_i \leq b$, resulting in the constraint $g(\lambda)$ for the variational parameter, $\lambda_i \leq b\sum_j \lambda_j$; and

C4. the minimum value of the i-th element of θ is known, i.e., $\theta_i \geq b$, resulting in the constraint $g(\lambda)$ for the variational parameter, $\lambda_i \geq b\sum_j \lambda_j$.

The four abstract types correspond to linear constraints regarding the global variables of the AMM. Yet, the four abstract constraint types span a variety of domain knowledge that the developer might have regarding the agent's behavior. For instance, if the developer wants to specify that the agent will not to choose an action a_i in a given decision state (s_i, x_i), she can utilize $C2$ to encode $\pi(a_i|s_i, x_i) = 0$. Further, if the developer is not certain but confident to the degree p, the constraint $C3$ can be used to specify the degree of uncertainty in the domain knowledge $\pi(a_i|s_i, x_i) \leq (1 - p)$. Similarly, for the transition function T_x, if the developer wants to specify that the transition to two states x_i and x_j are equally likely then the constraint type $C1$ is useful. For cases where one of the alternatives is highly likely than the others, then the developers can indicate the likelihood of that alternative via $C4$; for instance, while modeling behavior of a human driver, policy at the traffic signal may be reliably known and can be specified via $C4$.

Note that the Multinomial distribution is a member of the exponential family of probability distributions. Thus, the analytical expressions for the objective criterion and its derivatives (Eq. 4.34-4.35) are applicable while performing constrained optimization of the variational parameter. Due to the mean-field assumption, these constraints $C1$ to $C4$ apply only to a subset of the global update equations – specifically, to the global parameters λ for which domain knowledge is available – and do not impact the update of local variational factors. Thus, the computationally faster (unconstrained) variational inference can be used for the remaining parameters for which no auxiliary input is available.

Finally, as application of CVI to learning AMM does not change the update of local variational factors, this approach can be merged with all of the previous presented variants for AMM learning, i.e., parametric and nonparametric as well as supervised, unsupervised, and semi-supervised. Thus, we have arrived at the solution to Problem 1, namely: hybrid learning algorithms for recovering the complete AMM, which can learn from both data and domain expertise. Next, I summarize related approaches to learning models of other agents and empirically evaluate the efficacy of the proposed solution.

4.8 Related Approaches

Section 2.3 provides an overview of various paradigms of learning models of other agents. In this section, I provide a brief discussion of related research, with focus on approaches that learn models of sequential processes with explicit modeling of latent decision factors (or states).

Teh et al. provided an inference algorithm for the iHMM, a Bayesian nonparametric (BNP) hidden Markov model for scenarios in which the latent states are unknown [147]. Both sampling and variational inference algorithms for several extensions of iHMM have since been developed [38, 60, 127]. These approaches have been applied to segmentation and clustering of sequential data; however, they do not model agent policy or the decision-making process. The proposed approach to modeling the unknown state space, X, of other agent's behavior is inspired by these models and includes the explicit dependence of actions on states in order to model decision-making.

Computational human-robot interaction (HRI) research provides several examples of employing (variants of) supervised learning and inverse reinforcement learning [173, 118, 77] to estimate humans' policies during sequential tasks [125, 89, 159] – albeit with the complete specification and labeling of the decision factors. In contrast, in this chapter, we consider the learning problem when the state specification or labeling may be unavailable or only partially available. Although not utilized for HRI, BNP extensions of decision-making models, both for planning [29, 85] and IRL [97, 120, 71], have also been developed.

Nonparametric IRL approaches incorporate execution traces as the input data and aim to recover the decision-maker's latent state-dependent reward/policy. They result in better performance than parametric IRL approaches when complete state specification is unavailable, but aim to maximize accrued reward and do not seek to recover the true behavioral model. In contrast, the proposed solutions explicitly models the dependence of latent state transitions upon action and known states, and includes mechanisms for hybrid learning of the AMM.

In order to relax the assumption of rationality, AMM directly models policy and does not, explicitly, represent reward. However, the proposed approach for incorporating partial specifications of behavior is general and complementary to prior IRL approaches. In domains that include near-rational behavior, future extensions that combine CVI and IRL hold the potential to improve sample complexity.

In their work, Panella and Gmytrasiewicz provided a BNP approach to learning the PDFC from execution traces. They further used the learned PDFC to define subintentional interactive POMDPs and generate autonomous agent behavior during multi-agent tasks. The infinite AMM generalizes the PDFC model representation by considering stochastic transition of latent states. Further, to facilitate model alignment, the presented solution (a) includes a factored state representation to incorporate available information regarding the known states (S, T_s, ρ^s), and (b) incorporates a mechanism to utilize domain knowledge via CVI.

Recently, approaches that model latent states in sequential behavior using generative adversarial networks (GANs) and conditional variational autoencoders (CVAEs) have been developed [83, 132]. By modeling latent states, these approaches can predict and generate multimodal behavior. However, they require specification of the number of latent modes and do not model their dynamics. In contrast, by utilizing BNP priors, the proposed solutions can jointly learn the latent state space and their impact upon an agent's behavior. Another interesting direction for future work would be to explore the integration of these techniques with Bayesian nonparametrics.

4.9 Experiments

I conduct numerical experiments to confirm that the proposed approach for hybrid AMM learning can successfully incorporate partial specification of an agent's behavior. Specifically, through the experiments, I measure the ability of the proposed algorithms to infer the latent states of decision-making and learn models that are aligned with an agent's true model.

4.9.1 Methods

Access to the true model and the labels of latent states are necessary for measuring model alignment and validating the learning algorithms. Hence, in the experiments, I utilize simulated scenarios for which the true model was accurately known: namely, Line World, Highway, and an instantiation of an assembly task.

For each scenario, I first specify the true model as a complete AMM tuple. Using this tuple, I generate the problem inputs (namely, the partial AMM tuple, execution traces, labels, change-points, and partial knowledge of model parameters). I also generate a test dataset (a set of execution traces) to compare the predictive performance of the learned model. The problem inputs are then used to learn the missing elements of the AMM and infer the latent state sequences. I conduct experiments both for the parameteric and nonparametric versions of the problem of modeling other agent's behavior.

First, I present the evaluations for the nonparametric case, using the domains Line World and Highway. Next, I present the evaluations for the parametric case, using the assembly task, wherein the number of latent states (i.e., $|X|$) is known. In addition to the simulation experiments, for the assembly task, I also conduct experiments using data from humans. The unsupervised behavioral data from humans was collected using a motion capture system and divided into training and test sets. The data was manually labeled to arrive at the ground truth values of the latent states x and quantify the predictive performance of the learned models.

Metrics

In order to measure the state estimation performance, I utilize the normalized Hamming distance between the inferred and true state sequences using the Munkres algorithm [127]. The Munkres algorithm provides a correspondence between the inferred and true state labels, such that the normalized Hamming distance is minimized. This correspondence is also used to measure the predictive performance on the test set and the metrics of model alignment.

For measuring model alignment, I utilize a metric inspired by the weighted KL divergence [109]. Specifically, to quantify error in learning the transition probabilities T_x, I first use the Munkres correspondence to match the rows and columns of the learned $\hat{T}_x$ and true T_x, and then compute the KL divergence between each matched row. Finally, the metric is obtained as an average score weighted by the relative counts of the input data η described as follows:

$$w\mathrm{KL}(T_x, \hat{T}_x) = \sum_{x,s,a} \eta_{xsa} \mathrm{KL}(T_x(\cdot|x,s,a)||\hat{T}_x(\cdot|x,s,a)) \tag{4.37}$$

Similar procedures are used for measuring the model alignment for the remaining global variables π and b_x,

$$w\mathrm{KL}(\pi, \hat{\pi}) = \sum_{x,s} \eta_{xs} \mathrm{KL}(\pi(\cdot|s,a)||\hat{\pi}(\cdot|s,a)) \tag{4.38}$$

$$w\mathrm{KL}(b_x, \hat{b}_x) = \mathrm{KL}(b_x(\cdot)||\hat{b}_x(\cdot) \tag{4.39}$$

Baselines

The central aim of the evaluations is to assess the ability of the proposed solutions to incorporate partial specifications of behavior. I hypothesize that by incorporating successive auxiliary inputs (i.e., labels, change-points, and partial specifications) the proposed solutions can improve the accuracy of the learned model. Due to different model structures and input information, however, it is difficult to compare model alignment performance of AMM learning with related algorithms. Prior approaches cannot utilize several of the partial specifications, such as, partial knowledge of T_x and π as well as change-point information. Further, for methods that do not model latent states several metrics of interest are undefined.

Hence, to validate the ability of the proposed solutions to incorporate different partial specifications and improve the model accuracy, I compare the performance of the different variants of AMM learning – namely, (a) unsupervised AMM learning, (b) semi-supervised AMM learning, (c) hybrid, unsupervised AMM learning, and (d) hybrid, semi-supervised AMM learning.

Unsupervised learning serves as a proxy for approaches that exclusively rely on unsupervised execution traces, while the last version (hybrid, semi-supervised AMM learning) demonstrates my complete solution to Problem 1. The remaining versions demonstrate the utility of the proposed solutions for different input settings. Identical priors, hyper-parameters, initialization and termination conditions are used for all the algorithms. All four variants perform inference using the AMM model and thus learn the impact of latent states on the agent's behavior.

Further, I compare the performance of the developed algorithms with representative approaches from prior art. Variants of supervised learning have been used to learn human teammate models for HRI [77]; however, they require as an additional input labels of human's mental states. Labels of human's mental states can be obtained via manual annotation if the latent decision factors are known, i.e., the case with parametric priors. Thus, for the parametric case, I utilize supervised AMM learning as the representative approach for prior art. Supervised AMM learning also models the impact of latent states, but assumes that the latent state space is known and thus cannot be used for the case of nonparametric priors.

For the nonparametric case, I utilize an oft-used variant of inverse reinforcement learning for learning human models in HRI, namely, Maximum Entropy IRL (MaxEnt) [173]. Inference of unknown states is not possible using MaxEnt and the metrics Hamming distance, $w\text{KL}(T_x, \hat{T}_x)$ and $w\text{KL}(b_x, \hat{b}_x)$ are undefined. However, the policy alignment can be quantified using the weighted KL divergence between the true and learned policies.

Implementation

I implement the AMM learning algorithms as an extension of `pyhsmm`, a Python library for approximate unsupervised inference. For CVI, a constrained optimizer is required to solve the resulting constrained optimization problem; towards this end, I utilize SLSQP, a sequential least squares programming optimization algorithm as implemented in `scipy` [70, 62].

4.9.2 Partial Specification of Decision Factors

I first discuss the performance for scenarios where the developer has incomplete knowledge of the other agent's decision factors. This case corresponds to learning the AMM with nonparametric priors.

Line World

The first evaluation scenario models a simulated agent navigating a one-dimensional grid of length five. The agent's behavior depends on two decision factors: position as the known state s and goal as the latent state x. The action space of the agent includes three actions: left, right and wait. The known transition dynamics T_s are modeled as deterministic. The set of agent goals X include either ends of the grid. The agent exhibits goal-directed motion, and switches its goal after reaching the current goal.

The inputs to the learning algorithm include the partial AMM tuple (S, A, T_s), five execution traces (each of length 20), and the auxiliary inputs. Noisy change-point information for two execution traces with accuracy p_c=0.9 serves as the auxiliary input for semi-supervised learning. The region of the grid where the agent did not switch its goal is randomly selected and specified as the global input. Additionally, five execution traces are generated as the test set.

The objective of the learning algorithm is to recover the set of goals X, their switching dynamics T_x and the agent's policy π. Despite the small domain size, the learning problem is challenging since different model parameters (T_x, π) can generate identical behavior. Further, depending on its goal, the agent can choose different actions for the same location, and execution traces include cycles.

Explicit modeling of latent state improves learning performance. The results of the experiments, averaged over twenty-five trials, are summarized in Table 4.2. Undefined metrics are denoted by '—.' All versions of the AMM learning algorithm result in lower policy alignment error as compared to MaxEnt. Despite iden-

	MaxEnt	Unsup.	Semi-Sup.	Hybrid Unsup.	Hybrid Semi-Sup.
Hamming dist. (train)	—	0.51	0.39	0.30	0.13
Hamming dist. (test)	—	0.53	0.40	0.30	0.14
$w\text{KL}(T_x, \hat{T}_x)$	—	2.01	1.38	0.98	0.43
$w\text{KL}(\pi, \hat{\pi})$	1.08	0.77	0.56	0.41	0.19
$w\text{KL}(b_x, \hat{b}_x)$	—	0.85	0.87	0.53	0.51

Table 4.2: State inference and model alignment errors for learning agent models in the Line World domain.

tical input data, variational inference applied to AMM (i.e., unsupervised learning) demonstrate better model alignment and could perform state inference. This result indicates the utility of explicitly modeling the unknowns as part of the AMM and learning their impact on the agent's behavior.

Hybrid learning outperforms learning with data alone. The complete solution – hybrid, semi-supervised learning – results in models with better alignment (lower wKL scores) as compared to the remaining baselines. Along with improved model alignment, hybrid learning with both global and local inputs (i.e., hybrid semi-sup. learning) results in lower Hamming distance between the true and inferred sequences for both the training and test datasets.

Learning with even one partial specification improves model accuracy. Note that semi-supervised learning has access to local auxiliary inputs (noisy change-point information for two execution traces) and hybrid, unsupervised learning has access to the global auxiliary inputs (partial information of T_x). Utilization of even one auxiliary input – i.e., semi-supervised learning and hybrid, unsupervised learning – improves upon the baseline of unsupervised learning that relies only on execution traces. This validates that the proposed solutions for learning the AMM are capable of utilizing auxiliary information when available. Interestingly, while the auxiliary inputs do not include information regarding the agent's policy, the alignment between the true and learned policy improves by incorpo-

	MaxEnt	Unsup.	Semi-Sup.	Hybrid Unsup.	Hybrid Semi-Sup.
Hamming dist. (train)	—	0.56	0.36	0.48	0.35
Hamming dist. (test)	—	0.63	0.44	0.55	0.42
$w\text{KL}(T_x, \hat{T}_x)$	—	2.11	1.12	1.03	0.64
$w\text{KL}(\pi, \hat{\pi})$	1.04	0.48	0.38	0.38	0.33
$w\text{KL}(b_x, \hat{b}_x)$	—	1.67	1.75	1.50	1.58

Table 4.3: State inference and model alignment errors for learning agent models in the Highway domain.

rating any one or both of the auxiliary inputs. This improvement in learning is possible since the proposed approaches for AMM learning perform joint inference of the agent's model parameters (e.g., T_x and π).

Highway Domain

As the second domain, I utilize a modified version of the highway domain [2, 120]. Briefly, in the experiments, this domain models a highway with two lanes and two shoulders (one on each side), multiple civilian cars, and a police car. The police car can use the entire road, while the other cars are restricted to the two lanes. The objective of the learning algorithm is to model the behavior of the police car (henceforth, referred to as the agent).

The agent drives at a constant speed, which is faster than that of the other cars on the highway, and can choose to switch lanes at each timestep. The agent's actions depend on known, observable factors (distance to cars and current lane) as well as latent states (whether it is driving nominally or pursuing a civilian car). The agent switches from the nominal driving mode to pursuing a civilian car if two civilian cars seem to be racing (which the agent determined based on the relative distances of the cars), and returns to the nominal mode after catching a car. The learning algorithms have access to the known states of the agent, its action space and dynamics of the highway domain T_s; however, the number of latent states x and their dynamics T_x are unknown.

Experiments for the highway domain are conducted in a similar fashion to that for the line world scenario; however, longer execution traces (each of length forty) are used for both training and testing datasets. For this domain, the learning algorithms have to reason over a much larger problem with an observable state space of size 200.

Despite incomplete specification of decision factors, hybrid learning improves model alignment. The results of the experiments, averaged over twenty-five trials, are summarized in Table 4.3. Similar to the first scenario, the effect of modeling the unknowns is evident, as all approaches utilizing the iAMM as the underlying model (including unsupervised learning) result in lower policy alignment error as compared to MaxEnt. Further, we observe that the proposed algorithms can successfully incorporate auxiliary inputs and improve model alignment when one or both auxiliary inputs are available. Similar performance is observed for the state decoding and prediction errors (normalized Hamming distance on training and test data, respectively) when the auxiliary inputs are utilized.

In summary, the results listed in Tables 4.2-4.3 confirm the utility of the proposed algorithms for modeling other agents without complete specification of their decision factors. In the next set of experiments, I evaluate whether the benefit also translates to scenarios in which all of the decision factors are known.

4.9.3 Complete Specification of Decision Factors

In applications of human-robot collaboration, there exist scenarios where the decision factors of the human teammate are known based on the task structure or domain expertise of the robot developer. Such scenarios motivate the use of approaches for AMM learning with parametric priors. As an example, in this section, I utilize an assembly task instantiated in a kitchen environment (henceforth, referred to as the sandwich preparation task). Through experiments with both synthetic and human data, I evaluate the ability of the proposed algorithms to learn an AMM describing the human's task-specific behavior.

Figure 4-3: The analogue kitchen environment used for instantiating the single-agent assembly task (i.e., the sandwich preparation task).

Sandwich Preparation Task

Specifically, I consider a human tasked with preparing sandwiches in an analogue kitchen environment. The kitchen environment is depicted in Fig. 4-3. Key landmarks in the kitchen include: two cooking stations, two wrapping stations, and a pantry. Preparation of sandwiches requires the human to bring required ingredients from the pantry (cabinets), make a sandwich by assembling the ingredients and applying ketchup or mustard sauce (at one of the two cooking stations situated on the left half of the table), and wrap the sandwich in napkins (at one of the two wrapping stations on the right half of the table). The task structure is analogous to final assembly tasks in a factory, where the human goes to depots, collects parts/tools, and assembles them at different locations of interest.

The human needs to make four sandwiches and completely finish preparing one sandwich before proceeding to the next one. Further, if the human makes a sandwich with ketchup (placed on far left of the table), he has to wrap the sandwich in the red napkin. Similarly, if the human makes a sandwich with mustard (placed on middle left of the table), he has to wrap the sandwich in the yellow napkin. Thus, the sandwich recipe requires the human to move between different landmarks, subject to domain-specific constraints, and perform temporally-extended activities (e.g., bring ingredients, wrap sandwich) at the landmarks.

The developer is interested in modeling the task-specific motion of the human, so that a collaborative robot may use that model to safely and efficiently collaborate with the human. The human's task-specific motion depends both on its intent (i.e., the activity the human wishes to perform) as well as his current position. The human's position is observable and, thus, corresponds to the observable decision factor s. However, the human's intent is a mental state and, thus, corresponds to the latent decision factor x. While intent is a latent state, the developer can specify the set of intent (i.e., the latent state space X) using her knowledge of the task. Thus, the problem of modeling the human's task-specific behavior corresponds to that of learning the AMM with parametric priors.

Experiments with Synthetic Data

I first utilize a simulated version of the sandwich preparation task to evaluate the performance of AMM learning with complete specification of the decision factors. Similar to previous experiments, I create a hand-crafted AMM to describe the behavior of the simulated human. The human state is specified as position $s \in S$ and intent $x \in X$. The human action $a \in A$ corresponds to his motion primitives. The human had a total of five intents (i.e., $|X| = 5$) corresponding to the five landmarks of the kitchen environment: two cooking stations, two wrapping stations, and the pantry. The states, actions, and the motion kinematics complete the specification of partial AMM tuple (S, A, X, T_s).

Hand-Crafted AMM The initial distribution b_x and dynamics of the latent state T_x are specified using the task structure (i.e., the human starts at party, the human goes to one of the cooking stations after the pantry, the human goes to one of the wrapping stations after the cooking station, and so on) and the duration of each activity (which is used to compute the probability of self transitions). In order to arrive at the AMM policy, the simulated human is modeled to follow intent-driven motion, albeit with bounded rationality. For each intent x, first a rational policy is generated using a Markov decision process, with high positive reward

	Sup.	Unsup.	Semi-Sup.	Hybrid Semi-Sup.
Hamming dist. (train)	0.000	0.304	0.009	0.000
Hamming dist. (test)	0.002	0.308	0.010	0.002
$w\mathrm{KL}(T_x, \hat{T}_x)$	0.011	0.750	0.023	0.014
$w\mathrm{KL}(\pi, \hat{\pi})$	0.041	0.717	0.053	0.042
$w\mathrm{KL}(b_x, \hat{b}_x)$	0.367	1.011	0.368	0.368

Table 4.4: State inference and model alignment errors for learning agent models in the sandwich preparation task with simulated data.

for completing the intent and a small negative reward otherwise. Next, through a softmax transformation, the rational policy is converted to the bounded-rational policy π for the AMM. The global AMM parameters (π, T_x, b_x) coupled with the partial AMM tuple completely describe the hand-crafted AMM.

Inputs for Learning The inputs to the learning algorithms include the partial AMM tuple (S, A, X, T_s), sixteen execution traces, and the optional inputs. While generating the execution traces, the simulated human is constrained to finish the activity corresponding to its intent within a pre-specified time interval. Duration of each sequence is $\approx$ 200 timesteps. As the set of decision factors is known, labels of latent states can be manually specified. Thus, supervised learning with explicit modeling of latent state is used as the baseline representing prior art. For supervised learning, the labels of human's intent are provided for the entire training dataset (i.e., $\approx$ 3200 labels). In contrast, for semi-supervised learning, labels of human's intent are provided for only four execution traces (i.e., $\approx$ 800 labels). For hybrid learning, around twenty high-level inputs describing the task structure and minimum duration of each activity are used to specify the partial knowledge of the transition function T_x via the constraints $C3$ and $C4$, respectively.

In order to evaluate the predictive performance of learned models, additionally, sixteen execution traces are generated as the test set. All algorithms, including supervised learning, explicitly model the human's latent decision factors and use the Bayesian AMM as the underlying model.

Developer's labeling effort improves learning performance. The results of the experiment, averaged over ten trials, are summarized in Table 4.4. Among approaches that cannot utility high-level domain expertise (i.e., Sup., Unsup., and Semi-Sup.), we observe that supervised learning performs best, followed by semi-supervised and unsupervised learning. The result is not surprising; the three approaches utilize the same underlying representation and supervised learning has access to the highest number of labels. The result, however, confirms the utility of the developer's labeling effort for generating teammate models. Further, it demonstrates that, when decision factors are known and their labels are available, supervised AMM learning successfully replicates the paradigm considered in prior art: learning teammate models from labeled data.

Hybrid learning performs as well as supervised learning despite fewer labels. Interestingly, despite significantly fewer labels ($\approx$ 800 labels), we observe that hybrid semi-supervised learning performs equally well as supervised learning (which uses $\approx$ 3200 labels). The result confirms not only the potential of high-level domain knowledge for improving model learning but also the ability of the proposed approach for hybrid learning approach to realize this potential. The paradigm of hybrid learning is, thus, ripe for being utilized for other problems, including for specifying a collaborative robot's interaction policy. However, before proceeding to the discussion of a hybrid approach for specifying robot policy in the next chapter, I present the experiments with human data.

Experiments with Human Data

In the experiments thus far, we have sought to learn the model of simulated agents. By evaluating the approach in simulation, we have had access to the parameters of the ground truth model, which enables the validation of the proposed solutions via computation of metrics of model alignment (i.e., weighted KL divergence). In order to evaluate the approach for learning human models, next, I conduct experiments with data collected from human subject.

	Sup.	Unsup.	Hybrid Semi-Sup.
Hamming dist. (train)	0.000	0.336	0.037
Hamming dist. (test)	0.027	0.341	0.045

Table 4.5: State inference errors for learning agent models in the sandwich preparation task with human data.

Data Collection A dataset of humans performing the sandwich preparation task was collected in the analogue kitchen environment shown in Fig. 4-3. The position of the human's arm was tracked using a motion capture system; consequently, the human participants were required to wear an LED-glove during the task. The unsupervised execution traces collected via the motion capture system were manually annotated to obtain the ground truth sequences of the human's intent. The dataset included twelve training sequences and three test sequences. Since we do not have access to the true model of human subject's behavior, the metrics of model alignment cannot be computed for this set of experiments.

Experiments with human data confirm the utility of hybrid learning. The results of the experiment, averaged over ten trials, are summarized in Table 4.5. The training error of supervised learning (Sup.) is by definition zero, since it has access to labels for the entire training dataset. Similar to the experiments with synthetic data, the utility of both labels and high-level domain knowledge is evident from the results. Despite significantly fewer labels, the paradigm of hybrid learning results in near equal performance to that of the supervised approach.

Across the set of experiments, the consistency of state inference errors across training and test set demonstrate the benefits of using a Bayesian approach. The Bayesian priors, both in the parametric and nonparametric cases, act as regularizers and prevent over-fitting of the learned AMM to the training data. Finally, the joint learning of the latent variables (b_x, T_x, π, and values of the latent states x) improves both model alignment and predictive performance even with the help of one auxiliary input.

4.10 Discussions

Through the numerical experiments, I demonstrate that the proposed solutions for model learning are capable of learning AMM tuples that are aligned with an (human or artificial) agent's true model and that conform to the partial knowledge of an agent's behavior. Furthermore, as more information of varied types (e.g., execution traces, partial labels, noisy change-point sequences, partial knowledge of model parameters) is provided to the algorithm, the model alignment improves. This improvement in the agent model is possible despite incomplete specification of the agent's decision factors.

The ability to incorporate these novel input types enables the use of high-level information of the agent's behavior during the model learning and specification process. This approach is made possible through the use of a factored model representation (i.e., the Agent Markov Model) and a novel inference paradigm (i.e., Constrained Variational Inference). While prior approaches to learning an agent's model largely rely on either execution traces or label/feature queries, I provide a hybrid learning approach that can efficiently utilize both data and developer's domain expertise.

Thus, the solutions presented in this chapter realize *effective information transfer* via the paradigm of hybrid learning. The benefits of effective information transfer are demonstrated for the first problem, i.e., learning models of other agents with partial specifications of their behavior. In the following chapter, using collaborative versions of the multi-step assembly task, I investigate whether the benefits of effective information transfer also translate to Problem 2, i.e., for specifying a collaborative robot's interaction policy.

Future Directions

Existing active learning approaches query a domain expert for additional information [135, 21]. However, in these approaches, the developer is typically limited to providing her domain expertise as additional labels or features.

I posit that the development of learning approaches that are both active and hybrid can further advance effective information sharing between the developer and the robot. Thus, a promising direction of future work is the development of active learning approaches, wherein the observer can actively choose to query the domain expert regarding not only labels but also high-level specifications.

My solution for hybrid learning (namely, CVI) can serve as the building block for such approaches. For instance, it can enable the active and hybrid learning algorithm to reason about the potential benefit of different input types beyond labels (including domain knowledge regarding latent model parameters). Another related application of hybrid learning includes learning generative models from data that conform to preferences, available as logical or procedural constraints, provided by a human user. Both the global and local auxiliary inputs provide avenues to incorporate preferences in the developed hybrid learning approach.

Our motivation behind learning agent models is to enable effective human-robot collaboration. The algorithms presented in this chapter can be used both by a collaborative robot (for learning models of human behavior) and by the human teammate (to learn transparent models of robot behavior) [58, 170]. In the following chapter, I discuss one such application of hybrid learning for specifying the interaction policy of collaborative robots.

I identify three areas of further improvement of the proposed approach. Firstly, in the nonparametric case, the proposed approach is limited in that it can learn the latent states and the AMM tuple, but not the semantic labels or interpretation of these latent states. I posit that interaction between a human (domain expert or robot developer) and the learning algorithm will be necessary for incorporating interpretability in the latent states learned via Bayesian nonparametrics. Secondly, my approach is limited to a scalar latent state and, thus, utilizes a flat state representation for the latent state. Lastly, I utilize tabular representations for the unknown AMM parameters (namely, π, T_x). Promising future directions include addressing these limitations through the use of function approximation techniques and factored latent state representations.

4.11 Summary

In this chapter, I present solutions towards the problem of learning an agent's true decision-making model with partial specification of its behavior. In order to formalize and address this problem, I utilize a factored representation of sequential decision-making behavior, the agent Markov model (AMM). I pose the learning problem as one of Bayesian inference, and provide novel variational inference algorithms for the AMM that can utilize different types of information – including, sequences of observed behavior, prior knowledge of agent's policy, and partial specification of agent's state and dynamics. Through numerical experiments, I validate that the proposed solutions are capable of utilizing varied information types and, consequently, learning models that align with the true behavioral model of the other agent.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Reducing Developer Effort for Specifying the Interaction Policy

In this chapter, I present a framework that addresses Problem 2, i.e., the generation of a collaborative robot's *interaction policy* by utilizing both data and domain expertise. The framework enables the development of collaborative robots that exhibit two of the characteristics of effective teaming, namely,

- anticipate the teammate's (i.e., the human's) behavior; and
- adapt own (i.e., the robot's) behavior to improve collaboration.

As detailed in Section 2.4, a few approaches for realizing collaborative robots with these characteristics have been recently developed. However, in practice and especially for collaborative tasks with large problem sizes, the specification of the interaction policy remains effort-intensive. In order to enable human-robot collaboration at scale, approaches that can accelerate this specification process will prove to be critical. The proposed framework, thus, is designed to reduce the developer effort while specifying the interaction policy.

The design of the framework is informed by the opportunity of effective information transfer. Specifically, in addition to behavioral data, a robot developer often has domain knowledge that is difficult to utilize while specifying the interaction policy. In order to leverage this domain knowledge, encouraged by the perfor-

mance of hybrid learning for modeling the human teammate, I provide a hybrid approach to learning a robot's decision-making model.

Along with the specification of a decision-making model, interaction planning in real-world collaborative tasks is also challenging. The large problem size limits the use of offline planning techniques, while the limited planning time available during interaction makes the online computation of robot decisions difficult. In order to enable online robot decision-making, the proposed framework leverages the factored structure of the decision-making model and state-of-the-art algorithms for planning under uncertainty.

I begin this chapter with an overview of the proposed framework, followed by a detailed description of the model specification process and interaction planning. The framework is then demonstrated on two human-robot collaborative tasks with state spaces significantly larger than prior art (over 1 million states). Through both simulation and human-subject experiments, I confirm that the framework can generate fluent human-robot collaboration with significantly reduced developer effort and within short planning times (less than a second).

5.1 ADACORL: Adaptive Collaboration with Reduced Labeling

In Figure 2-3, I summarize the several paradigms available for specifying a collaborative robot's behavior. On the one hand, paradigms that require minimal effort from the developer (such as RL) require prohibitive amounts of interaction data (i.e., training time with the human teammate. On the other hand, paradigms that solely rely on the developer's specification not only result in sub-optimal teamwork but also require prohibitive amounts of developer effort.

Thus, I adopt an interim paradigm, wherein first, the developer specifies a model and then planning algorithms are used to generate the interaction policy. In order to reduce the developer effort, a hybrid approach is utilized for specify-

ing the model. The specification approach is hybrid in that model is learned by utilizing both behavioral data and the developer's high-level domain knowledge. Given the decision-making model, the interaction policy is generated using algorithms for planning under uncertainty. I refer to the problem of generating an interaction policy given the decision-making model as **interaction planning**.

The model specification and interaction planning paradigm is realized through a novel framework titled: Adaptive Collaboration with Reduced Learning or, briefly, ADACORL. By utilizing representation and algorithms tailored for hybrid model specification, ADACORL can generate fluent teamwork while reducing the developer effort. By employing online planning algorithms, ADACORL can generate interaction policies for large problems and during human-robot interaction. As summarized in Fig. 5-1, ADACORL includes four steps

- specification of the collaborative task;
- hybrid learning of the human teammate's task-specific behavior;
- analytical computation of the robot's decision-making model; and
- algorithmic generation of the robot's interaction policy.

Briefly, in step 1, the developer specifies a model of the collaborative task. Using behavioral data and the developer's domain expertise, in step 2, ADACORL creates a model of the human teammate's task-specific behavior. The requirement of manual effort (i.e., specifications required from the developer and the training data required from the teammate) is limited to the first two steps. In step 3, ADACORL analytically creates a decision-making model for the robot using the outputs of the previous steps (i.e., the task specification and human model). Finally, in step 4, the robot's interaction policy is generated using planning algorithms.

ADACORL includes several features to facilitate rapid prototyping and development of collaborative robots, including the use of hybrid learning, encapsulation of the manual effort to the first two steps, and modularity of task and human models. In the remainder of this section, I provide an overview of the four steps before discussing their mathematical details in the subsequent sections.

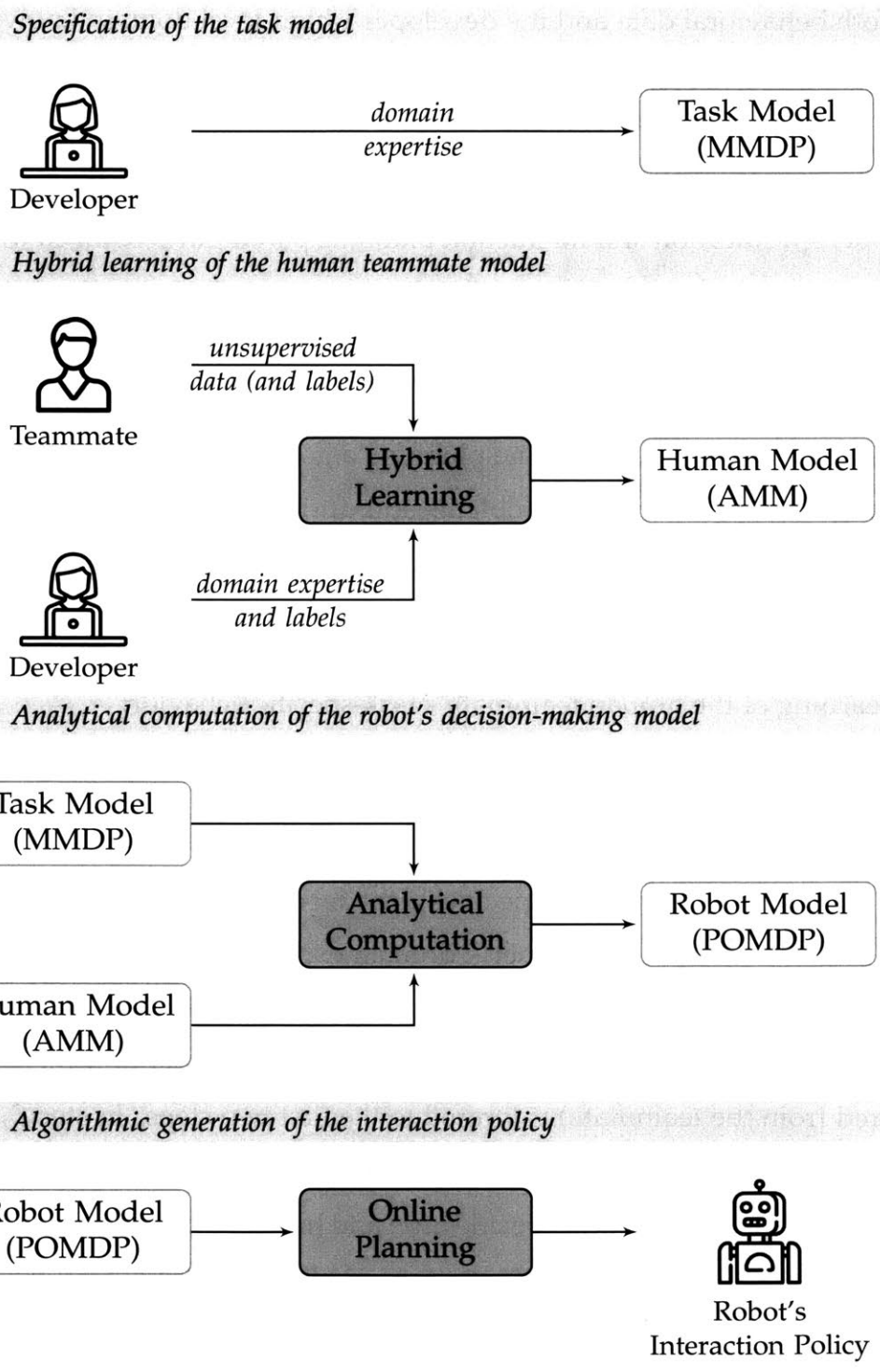

Figure 5-1: AdaCoRL: A hybrid framework for generating a robot's interaction policy for a human-robot collaborative task.

Step 1: Specification of the Task Model

AdaCoRL enables developers to design the behavior of a collaborative robot for a specific collaborative task. Thus, the framework implicitly requires the developer to have complete knowledge of the constraints and objectives of the task. In step 1, the developer specifies the rules and the objective of the collaborative task via a mathematical model. While the task is known to the developer, the choice of the model representation is critical to facilitating model specification.

Thus, AdaCoRL utilizes the framework of multi-agent Markov decision processes (MMDP) [106] with a factored representation of the state and transition model. Due to the factored nature of the model, the developer can specify the rules and the objective of the task as well as the dynamics of the human and robot in a modular fashion.

Step 2: Hybrid Learning of the Human Model

A model of human behavior is required to predict human states and actions and suitably select robot actions. While a developer has complete knowledge of the task for which she is designing the collaborative robot, she might only have a partial understanding of how the human behaves during the task. Thus, algorithmic approaches that learn models of human behavior from interaction data have been previously developed. However, making matters challenging, human teammate's behavior depends on mental states (such as goal, trust, workload, and attention) that are both difficult to quantify and measure. Hence, as detailed in Section 2.3, prior approaches require the developer to annotate behavioral data with labels of the mental states.

In order to reduce the effort required for labeling behavioral data, AdaCoRL leverages the hybrid learning algorithms developed in the previous chapter to generate models of human behavior. Consequently, the Agent Markov Model (AMM) is used to represent the task-specific behavior of the human teammate.

Step 3: Computation of Robot's Decision-Making Model

During the collaborative task, both the human and the robot have autonomy over their actions. The human does not dictate the robot's behavior and, conversely, the robot can only control its own actions and not that of the human. In order to solve for a policy that only specifies robot actions, thus, ADACORL requires a single-agent model of decision-making that only depends on robot actions. However, the multi-agent model of the collaborative task depends on the actions of both the human and the robot.

In the third step, ADACORL analytically computes the required single-agent model using the task and teammate model. Motivated by the presence of latent states that impact the human teammate's behavior, ADACORL utilizes the representation of partially observable Markov decision processes (POMDPs) [106] as the robot's decision-making model.

Step 4: Generation of Robot's Behavior (Interaction Policy)

Given the robot's decision-making model, the interaction policy is generated using a variant of the R-DESPOT algorithm, a state-of-the-art anytime online POMDP solver [171]. The time to plan the robot's behavior during human-robot collaboration is relatively short (often less than a second). An alternative is to utilize offline POMDP solvers, which precompute the robot policy before task execution. However, for problems with large state and action spaces, the applicability of offline solvers may be limited due to memory and computation requirements.

Hence, ADACORL utilizes an online solver, wherein decision-making is interleaved with task execution. In order to enable decision-making despite the short planning time, the online solver utilizes a precomputed but suboptimal default policy and an upper bound on the collaborative performance. Both the default policy and the upper bound are computed algorithmically prior to task execution. Further, since the default policy can be suboptimal, its computation is significantly faster than the computation of the interaction policy via an offline POMDP solver.

5.2 Specification of the Collaborative Task

In AdaCoRL, the generation of interaction policy begins with the specification of the human-robot collaborative task of interest. In the following description, the task specification refers to the rules and the objective of the task (i.e., a specification of what the team is to do). Importantly, the specification does not prescribe a policy (i.e., how should the human and the robot act to complete the task). In this section, I detail the representation to be used by the developer for specifying the collaborative task.

5.2.1 Multi-Agent Model of the Collaborative Task

As described in Section 3.1.2, this dissertation focuses on collaborative scenarios that can be represented using Markovian models. Specifically, the general model of human-robot collaboration is described as a factored variant of the decentralized POMDP. While addressing Problem 2, I limit the scope to problems where both the human and the robot have full observability of the task state. The full observability results in reducing the task model from a decentralized POMDP to a multi-agent Markov decision process.

I reproduce the task model from Section 3.1.2, with the additional feature of full state observability, as follows:

- $S \equiv S_H \times S_R \times S_E$ denotes the set of states (s) of the collaborative task. The model uses a factored representation where $s \equiv (s_H, s_R, s_E)$, in which $s_H \in S_H$ and $s_R \in S_R$ correspond to human- and robot-specific features, respectively, and $s_E \in S_E$ denotes additional features describing the task structure and the environment;
- $A \equiv A_H \times A_R$ denotes the set of joint actions $a \equiv (a_H, a_R)$. A_H and A_R denote the action space of the human and the robot, respectively. The actions of agents can be temporally-extended (i.e., macro actions). While the actions are represented as joint actions it does not prevent the agents from making decisions and acting autonomously – a necessary feature for modeling human-robot teams;

- The state dynamics are assumed to be Markovian and are governed by the transition function $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$. The transition function is assumed to have the following structure, $T(s|s,a) = T_H(s'_H|s,a) \cdot T_R(s'_R|s,a) \cdot T_E(s'_E|s,a)$, where T_H, T_R, T_E are transition functions for the state factors;
- The team receives a shared reward at each step, $R(s,a) : S \times A \rightarrow \mathbb{R}$;
- Both the agents, human and robot, have full observability of the state s.
- $\gamma \in [0,1]$ denotes the discount factor; and
- (optionally) $t_f \in \mathbb{Z}^+$ denotes the time horizon of the problem.

The objective of the human-robot team is to maximize the expected cumulative discounted reward. In addition to full observability of the MMDP state, ADACORL assumes that the developer can accurately specify the parameters of the MMDP task model (S, A, T, R, γ).

5.2.2 Mapping Domain Expertise to Model Parameters

While utilizing ADACORL, the MMDP parameters enable the developer to encode their domain expertise regarding the rules and the objective of the task. Specifically, the developer's knowledge maps to the MMDP parameters as follows:

- team's objective as the shared reward R;
- human-specific features via the state components $s_H \in S_H$;
- robot-specific features via the state components $s_R \in S_R$;
- task- and environment-specific features via the state component $s_E \in S_E$;
- abilities of the human and robot via the joint action $a = (a_H, a_R) \in A$;
- dynamics and constraints of the human-specific feature via the corresponding component of the transition model T_H;
- dynamics and constraints of the robot-specific feature via the corresponding component of the transition model T_R;
- dynamics and constraints of the environment as well as the rules of the task via the the transition model of s_E, i.e., T_E; and
- (optionally) time horizon of the collaborative scenario via t_f.

Implications of a Modular and Factorized Model

The modular nature of the MMDP tuple (S, A, T, R), the factored structure of the MMDP state (s), and the factorization of the transition model $(T = T_H \cdot T_R \cdot T_E)$ are especially useful during the model specification process. For instance, since the same robot model (s_R, T_R) can be reused for different humans, the factored structure facilitates personalization of robot behavior to different humans. Similarly, transferring the specification for a human-robot team to a new task requires modification primarily to the task-specific features s_E. Moreover, if the environment and team remain identical but the task's objective changes, only the reward function of the MMDP needs to be re-specified.

Thus, the task representation used by ADACORL facilitates rapid prototyping and implementation while transferring the specification between tasks, environments, or agents. On the flip side, the framework is limited in application to collaborative tasks that can be suitably described with a Markovian representation and exhibit the factored structure of the transition model. However, as elaborated in Chapter 3, several real-world scenarios of human-robot collaboration across a variety of domains (e.g., manufacturing, service robotics, and disaster response) satisfy these requirements.

5.2.3 Examples of Task Specification

In order to demonstrate the specification process, next, I discuss the task specification for two human-robot collaborative scenarios. The scenarios model two canonical human-robot collaboration tasks, namely, shared workspace and handover tasks. While these canonical tasks are general with instantiations in many domains, in these discussions, I situate them in a kitchen environment. Specifically, I utilize the kitchen environment considered in the experiments presented in the previous chapter. As we will see through these examples, the specification of the task model does not require the robot developer to reason about the human teammate's behavior.

Figure 5-2: A human-robot team preparing meals in a shared workspace.

Example 1: Shared Workspace Task

Consider a human-robot team assigned to prepare several meals in a kitchen. In the current example, one meal consists of a sandwich and a beverage. The kitchen environment, depicted in Figure 5-2, includes two cooking stations, two wrapping stations, four cups, and a pantry. The cups, cooking stations, and wrapping stations are all located on a table accessible to both the human and the robot. The pantry includes the ingredients required to prepare the sandwich but is accessible only to the human. The objective of the human-robot team is to prepare four meals as soon as possible while safely sharing space with each other.

Due to their complementary abilities, the dexterous human is tasked with preparing the sandwich, and the robot is tasked with pouring beverages in the cups. Sandwich preparation requires the human to pick up ingredients from the pantry, assemble them at one of the cooking stations, and then wrap the sandwich at one of the wrapping stations. The robot begins the task with a beverage in its hand and has to pour it in each of the four cups. While each teammate has been allocated a specific component of the task, the teammates have significant flexibility in performing the task. For example, while preparing the four sandwiches, the human teammate can choose the order in which to make them. Similarly, the robot can decide the order in which to fill the cups with beverages. Further, while preparing meals, both the human and the robot are free to choose their motion.

As the human-robot team is working in a shared workspace (the kitchen table), they need to coordinate their decisions in order to complete the task safely and efficiently. For instance, due to space constraints, the robot cannot pour a beverage in a cup while the human is working in an adjacent area (i.e., at a cooking station or wrapping station located next to the cup). Further, in order to preserve safety, the robot is programmed to stop if its distance to the human is less than a developer-specified safety radius. Thus, to complete the task efficiently and safely, the human and robot need to utilize the flexibility available in task execution and motion selection. This task is referred to as the *shared workspace task* [52].

Having described the collaborative task, I next exemplify how a developer can map these specifications to the parameters of the task model,

- For the safe and efficient sharing of space, the position of both the human and the robot are pertinent features for collaboration. The developer can encode these features via the appropriate state components — specifically, human's position via the state component $s_H \in S_H$ and the robot's position via the state component $s_R \in S_R$. Depending on the desired modeling detail, the position can be represented by in a Cartesian space (e.g., position of the end effector) or in the configuration space (e.g., joint angles of the robot).
- The task requires the human and the robot to visit multiple landmarks of interest (pantry, cooking station, wrapping station, and cups) and to perform temporally extended sub-tasks (such as make a sandwich, wrap a sandwich, pour a drink) at the landmarks. Thus, in addition to the position of the teammates, which and how many meals have been prepared are important to monitor task completion. The progress of the sequential task is encoded via $s_E \in S_E$.
- Action space, $A \equiv A_H \times A_R$, of the task model corresponds to the set of decisions available to the team for changing the state s. For the shared workspace task, where coordination of both the motion and the sub-tasks is important, the action space of each teammate (i.e., A_H and A_R) can include their motion primitives (e.g., joint angle increments in the configuration space) and macro actions (e.g., temporally-extended options, such as pour a drink).

- Having specified the state and action features, the specification of the factors of the transition model is straightforward. The transition models for positional features of the human and robot, T_H and T_R, correspond to the motion dynamics of respective agents. The motion dynamics depend on the physical capabilities (e.g., maximum velocity, acceleration, joint limits) of the agents as well as the static obstacles in the shared workspace. The transition model of features describing task progress, T_E, corresponds to an enumeration of potential ways (task-level recipes) to complete the task. In order to synchronize and enable planning of joint actions, transition models are specified such that the transition of each factor of the state occurs at the same temporal frequency.
- Finally, the reward function, R, is used to specify the desired collaborative behavior. In order to encode efficiency, the team receives a negative reward at each timestep until task completion. Further, in order to emphasize safety, the reward penalizes human-robot collisions.

Example 2: Handover Task

In the first example, we observe the specification process of the task for a scenario in which the human and robot had to coordinate their decisions (i.e., the joint actions) to safely and efficiently share space. Next, I present a scenario where the human and robot have to work together to complete a sequential assembly. Although such collaborative assembly tasks are found across domains (cf. Sec. 3.1.3), I continue to instantiate the examples in the kitchen environment.

The human teammate is tasked with making four sandwiches by bringing ingredients from one of the pantries. However, one of the ingredients is missing. The robot, however, has access to the missing ingredients. In order to complete the sandwich preparation, the robot has to handover the missing ingredient to the human. In contrast to the shared workspace task, the robot is not tasked with pouring beverages in this task, and the human does not need to wrap the sandwiches. The robot needs to predict when and where the human will be to effectively perform the handover and successfully accomplish the collaborative task.

Figure 5-3: The robot delivering the required ingredient to its human teammate.

I refer to this task as the *handover task* [52]. A still from the handover task, where the robot is delivering the required ingredient to the human teammate, is shown in Figure 5-3. Having described the collaborative task, let us discuss how the developer's domain expertise maps to the parameters of the task model,

- There are several similarities between the handover and the shared workspace tasks, including the kitchen environment, the capabilities of the human, and the motion dynamics of the robot. The modular nature of the task model enables the developer to leverage these similarities in the model specification process.
- For instance, the state components s_H and s_R still specify the position/configuration of the human and robot, respectively. Similarly, the state component s_E specifies the progress of the collaborative handover task. The action space, similar to the shared workspace task, encodes motion primitives and task-specific macro actions (i.e., perform a handover). Further, the structure of the transition model remains identical to that of the shared workspace task.
- However, since the task objective of the handover task is different than that of the shared workspace task, the reward function R needs to be modified. The team accrues a positive reward at each time a successful handover is completed. In order to encode safety, the team receives a negative reward when the human-robot distance is below a pre-specified safety radius. Finally, to emphasize efficiency, the team receives a negative reward at each timestep until task completion.

While specifying the two tasks, we observe that the ability to reuse part of the model significantly reduces the specification and programming effort. This ability is especially useful in real-world applications where a collaborative robot might have to be frequently re-purposed for different tasks, e.g., in small- and medium-sized enterprises and homes.

5.2.4 Solving the MMDP: Human-Robot Team Policy

Given the specification of the collaborative task as an MMDP, planning algorithms can be used for *solving* the MMDP and, consequently, generating a joint policy for the human-robot team [106]. However, the utility of this joint MMDP policy is limited for human-robot collaboration. Firstly, this paradigm requires the human to memorize and follow the joint policy in lockstep with the robot. Secondly, it limits the autonomy of the human and prevents them from exercising their preferences during task execution. Moreover, in tasks with multiple optimal policies, decentralized execution of a joint policy is challenging even for multi-robot systems [15]. Thus, approaches that enable a robot to coordinate its actions with the human, without limiting human autonomy, are necessary.

5.3 Model for Human's Decision-Making

For successfully collaborating with humans while preserving their autonomy, the robot needs the capability to anticipate and adapt to their behavior. Thus, in its second step, AdaCoRL utilizes hybrid learning to develop a model of the human teammate's task-specific behavior. As inputs for learning the human teammate model, AdaCoRL requires unsupervised behavioral data, labels of human teammate's mental states, and (optionally) domain expertise regarding human teammate's behavior. Modeling approaches that can effectively utilize the domain expertise are especially desirable, as they can reduce the sample complexity (i.e., the amount of behavioral data required for learning) and developer's labeling effort.

5.3.1 Agent Markov Model

Chapter 4 focuses on the problem of learning behavioral models of other agents that are performing sequential tasks. Recognizing that the human teammate can be viewed as the *other agent*, ADACORL builds upon the solutions developed in the previous chapter to learn a model of the human teammate. Consequently, when utilizing ADACORL, human behavior is represented using the Agent Markov Model (AMM). This section begins with a brief summary of the AMM, which is detailed in Chapter 4. Next, we discuss its application to modeling the human teammate, with emphasis on specifying domain knowledge regarding human behavior in the model learning process.

Briefly, the AMM models the sequential decision-making behavior of an agent with both observable s_A and latent x_A decision factors.[1] The latent decision factors x_A model the mental or unobservable decision factors of the agent. The agent's action selection is quantified by the agent's policy $\pi_A(a_A|x_A, s_A)$. The initial value of the mental state is characterized by the probability distribution $b_x(x_A)$. The transition dynamics of the decision factors are modeled as Markovian with the following factored structure: $T(s'_A, x'_A|s_A, x_A, a_A) = T_s(s'_A|s_A, a_A) \cdot T_x(x'_A|s_A, x_A, a_A)$, i.e., the mental states impact the observable states only via actions. In summary, the AMM is parametrized by the tuple $(X_A, S_A, A_A, b_x, T_x, T_s, \pi_A)$ and is depicted in Fig. 4-1.

5.3.2 Modeling the Human via Hybrid AMM Learning

Due to its focus on specifying robot policy for a particular collaborative task, ADACORL models the *task-specific* behavior of the human teammate. During the task, which is specified as an MMDP, the decisions available to the human teammate correspond to her actions (i.e., $a_H \in A_H$). Thus, the task-specific behavior corresponds to a predictive model of the human teammate's actions (or decisions).

[1] **A Note on Notation used in Chapters 4-5**: For ease of exposition, the AMM variables in Chapter 4 do not include a subscript. However, to avoid ambiguity between the state variables of the task model, human model, and robot model, I resume describing the AMM variables with the subscript A. Further, the variable A is overloaded. If used as subscript, it denotes AMM-specific variables. Else, it denotes the joint action space.

An essential component of such a predictive model is the human's decision-making policy π_H, i.e., a (potentially stochastic) function specifying the human's next action. The human's policy may depend on a variety of factors, including task-specific features (given by the task MMDP state s), robot's behavior, and human's mental states (such as intent, subgoals, trust). The AMM provides a mechanism to represent all of these variables of interest, i.e., the human's policy, the set of human's actions/decisions, and the set of human's decision factors.

Known AMM Parameters

The action space of the AMM is readily available from the task model, i.e., $A_A = A_H$. In order to ease the specification and model learning process, the AMM classifies the decision factors among two sets: observable and latent. In general, ADACORL requires the developer to specify the observable decision factor $s_A \equiv g(s, a_R)$ as a function of the MMDP state and robot action, resulting in $S_A \subseteq S \times A_R$. The function g is an arbitrary function, which provides the developer flexibility while specifying observable features impacting human behavior. As the observable decision factors and decisions are obtained from the task model, the developer can also leverage the transition function of the task model to specify the transition function of the AMM's observable decision factors $T_s(s'_A|s_A, a_A)$.

In this chapter, I limit the scope to scenarios where $s_A = s_H$, i.e., I assume that the human's decisions depend only on the human-specific feature of the task model s_H and unobservable mental states x_A. Within this scope, the specification of T_s is readily available from the task model due to its factored nature. Specifically, $T_s(s'_A|s_A, a_A) = T_H(s'_H|s_H, a_H)$. Thus, given the task model and the case where $s_A = s_H$, three parameters of the AMM tuple (S_A, A_A, T_s) are known a priori without any additional specification from the developer.

Inclusion of additional task features or robot actions as part of the observable decision factors would require the developer to specify the transition function T_s. In the following chapter, I provide one such extension where the known decision factors, in addition to s_H, include robot's communications.

Specification of Latent Decision Factors

The variable x_A enables the developer to specify latent features (i.e., mental states) that affect human behavior. For instance, the latent features can be used to capture the robot's influence on human behavior through constructs such as trust, adaptability, and compliance. In the example tasks, an important latent feature is human's intent (or subgoal), i.e., the landmark that the human intends to visit next (e.g., cooking station, wrapping station, or pantry). Other examples of latent features include cognitive measures such as attention, workload, fatigue, and stress. Similar to the previous chapter, I limit the scope to teammate models with only one latent feature impacting human behavior, i.e., x_A is both discrete and scalar.

In addition to the decision factors (or AMM states s_A, x_A), ADACORL also requires knowledge of the *set* of the decision factors (or AMM state spaces S_A, X_A). The state space of known decision factors is readily available from the task model, i.e., $S_A = S_H$. However, to enable parametric AMM learning, the developer needs to specify the set of latent decision factors X_A. In applications where the latent state represents the human's subgoal, this input corresponds to the specification of the number of goals. Similarly, if the latent state represents trust, this input corresponds to the number of discrete levels used to model and quantify trust.

Inputs for Learning the Human Model

Given the task specification, ADACORL requires only one additional specification (namely, X_A) from the developer to enable parametric AMM learning (X_A, S_A, A_A, T_s). In addition to these parameters, ADACORL assumes access to an unsupervised dataset of human behavior, i.e., N sequences of (s_A, a_A)-tuples. The developer, optionally, can provide labels of the latent state (i.e., M labels of x_A) and partial specifications (i.e., estimates of some elements) of the human's policy and latent state dynamics. In practice, providing labels of the latent states is effort-intensive for the developer and may require extensive instrumentation. Thus, approaches that can minimize the labeling requirement are desired.

Hybrid AMM Learning

Given the partial AMM tuple and inputs for learning, the latent parameters of the AMM (i.e., b_x, T_X, π_A) can be learned using algorithms for hybrid AMM learning. As observed in Chapter 4, despite significantly fewer labels, hybrid AMM learning can generate models of other agents (AMMs) that have near-equal accuracy to that of a fully supervised learning approach.

The encouraging performance of hybrid AMM learning is made possible by the use of constrained variational inference (CVI), which enables the use of partial specification of the human's policy and latent state dynamics during the learning process. For application of hybrid AMM learning to ADACORL, I consider the following types of partial specifications about latent state dynamics that a robot developer could provide,

PS_1) transitions from $x_A = i$ to $x_A \in X_n$ are not possible,
where the developer specifies i and X_n;

PS_2) transitions from $x_A = i$ to $x_A \in \{j, k\}$ are equally likely,
where the developer specifies j, k;

PS_3) minimum time t_i spent in the state $x_A = i$,
where the developer specifies i and t_i; and

PS_4) the subset of features from the set $\{s_A, a_A, x_A\}$
that impact the latent state transition.

These specifications enable the developer to provide partial specifications regarding the dynamics of the human's mental states. For instance, consider the case where the robot developer models x_A as the human teammate's subgoal (or intent) and has knowledge of the possible or likely subgoal sequences. If subgoal j is not possible after the subgoal i, this information can be specified via the first type of partial specification PS_1. Similarly, if any information is available about the duration of an activity i, it can be specified via PS_3. PS_{1-3} incorporate the developer's domain expertise specific to a single state.

For hybrid AMM learning, the partial specifications PS_{1-3} need to be converted to constraints over the variational parameters.[2] The specification PS_1 provides an upper bound on a few elements of the transition function and, thus, can be specified as the constraint type $C1$. Similarly, PS_2 and PS_3 correspond to the equality and lower bound constraints, $C2$ and $C3$, respectively. Finally, PS_4 enables the developer to perform feature selection while modeling the dynamics of human behavior.

5.3.3 Specification Steps of ADACORL

The first two steps of ADACORL, described thus far, correspond to the specification process of ADACORL. They require the robot developer to specify the task model (MMDP), and the set of human's latent decision factors ($x_A \in X_A$). Using data of human's task-specific behavior, the framework then algorithmically learns a model of human decision-making (AMM). It seeks to further reduce the developer's labeling effort by utilizing available domain expertise through a novel algorithm for hybrid learning. The final two steps, described next, do not require any developer involvement and algorithmically generate the robot's decision-making model (POMDP) and policy (π_R).

5.4 Model for Robot's Decision-Making

For successful collaboration, the robot needs to make decisions to maximize the team's objective. While the task performance depends on both the agents, the robot has autonomy only over its own actions. Thus, from the robot's perspective, the problem of interaction planning is viewed as a single-agent decision-making problem (cf. [58, 104, 125, 96]). Here, as the third step of ADACORL, I provide the approach to arrive at this model. Following prior research [58, 104, 20], I model the robot's decision-making using a single-agent POMDP [63].

[2]The constraint types $C1 - C4$ considered for hybrid AMM learning are described in Sec. 4.7

I denote the model parameters using the subscript c (for collaboration).

State Space The state of the decision-making model is denoted as $s_c \equiv (s, x_H)$, which is obtained by augmenting the MMDP state (s) with the human's latent decision factors (denoted by $x_H \in X_H$). The human teammate's latent decision factors are available from the human model, i.e., $x_H = x_A$ and $X_H = X_A$. The latent decision factors cannot be sensed during the interaction. I reiterate that, in contrast to the POMDP state, the MMDP state includes observable features of the human (such as position) but not the latent states (such as intent or subgoal).

Action Space $A_c \equiv A_R$ denotes the action space of the robot. The action space is readily available from the task specification. Only the robot's action space is considered for the single-agent model of decision-making.

Transition Model $T_c(s_c'|s_c, a_R) : S_c \times A_R \times S_c \rightarrow [0, 1]$ denotes the transition dynamics. The transition model only depends on the robot action. It models the effect of robot action on the collaborative task and the human teammate's task execution (via their effect on human's latent state). The transition model is derived from the task specification as follows,

$$
\begin{aligned}
T_c(s_c'|s_c, a_R) &= T_c(s', x_H'|s, x_H, a_R) && (5.1)\\
&= T_{cs}(s'|s, x_H, a_R) T_{cx}(x_H'|s, x_H, a_R) && (5.2)\\
T_{cs}(s'|s, x_H, a_R) &= \Sigma_{a_H \in A_H} T(s', a_H|s, x_H, a_R) && (5.3)\\
&= \Sigma_{a_H \in A_H} T(s'|s, a_R, a_H) Pr(a_H|s, x_H, a_R) && (5.4)\\
T_{cx}(x_H'|s, x_H, a_R) &= \Sigma_{a_H \in A_H} T(x_H', a_H|s, x_H, a_R) && (5.5)\\
&= \Sigma_{a_H \in A_H} Pr(x_H'|s, x_H, a_R, a_H) Pr(a_H|s, x_H, a_R) && (5.6)
\end{aligned}
$$

ADACORL assumes that the transition model is factored (Eq. 5.2). Further, the human's latent decision factor x_H impacts the task only through human's actions, i.e., $T(s'|s, x_H, a_R, a_H) = T(s'|s, a_R, a_H)$. Intuitively, this second assumption

of conditional independence models the fact that the human teammate's mental state can affect the task only via her actions. The term $T(s'|s, a_R, a_H)$ is the transition function of the task and, thus, is known based on the task specification. In addition, the probability distributions corresponding to human's decision-making policy $Pr(a_H|s, x_H, a_R)$ and latent state dynamics $Pr(x'_H|s, x_H, a_R, a_H)$ are required to define the POMDP transition model. These terms, however, are learned in the second step of AdaCoRL and, thus, are also readily available for analytically computing the transition model using Eqs. 5.1-5.6. Specifically, through hybrid AMM learning, AdaCoRL has access to the human's policy, $Pr(a_H|s, x_H, a_R) \equiv \pi_A(a_A|x_A, a_A)$, and latent state dynamics, $Pr(x'_H|s, x_H, a_R, a_H) \equiv T_x(x'_A|s_A, x_A, a_A)$.

Reward Similar to the transition model, the reward function $R_c(s_c, a_R)$ for the POMDP is obtained by combining the task specification and the human model,

$$R_c(s_c, a_R) = R_c(s, x_H, a_R) = \Sigma_{a_H \in A_H} R(s, a_H, a_R) Pr(a_H|s, x_H, a_R) \tag{5.7}$$

where, the shared reward $R(s, a_H, a_R)$ is known based on the MMDP.

Observation Space and Model The task state $s \in S$ is modeled as observable, while $x_H \in X_H$ is unobservable. The observation function models mixed observability, i.e., s is modeled as observable while x_H as unobservable.

The discount factor (as well as, if available, the time horizon) are identical to that of the MMDP. The above analysis describes the generation of the robot's decision-making model from the task specification (MMDP) and a model of human behavior (AMM). In addition to the modularity of the task model, the computation of robot's decision-making model is also modular. Specifically, the same human model can be utilized for different tasks, and vice-versa. This feature of AdaCoRL facilitates personalizing the behavior of collaborative robots and is especially useful in domains where the collaborative robot needs to interact with multiple humans.

5.5 Algorithm for Robot's Decision-Making

Having completed the specification of the robot's decision-making model, in the final step of the framework, we discuss ADACORL's approach to interaction planning. Given the robot's decision-making model, existing POMDP solvers can be used to arrive at the robot's interaction policy, π_R, which maps robot's belief about the state (s, x_H) to its actions (a_R). In prior human-robot collaboration research, different algorithms have been used to generate POMDP policies, including offline solvers [102] and hindsight optimization [58]. In order to reason about problems with large state spaces and short planning times, ADACORL leverages the R-DESPOT algorithm, an online POMDP solver, for interaction planning [171].

5.5.1 Regularized DESPOT

The Regularized DESPOT (or R-DESPOT) is an anytime online algorithm for solving POMDPs. By reasoning at execution-time, it can generate policies for problems with large state spaces, for which offline solvers may have memory bottlenecks. The anytime property of R-DESPOT is also desirable for interaction planning, as the planning time available during interaction is limited. Using the canonical equations of belief update and robot's sensory information, the algorithm maintains a belief $b(s_c)$ over the POMDP state. In order to identify the best action a_R^* given robot's belief, R-DESPOT performs forward simulations using the POMDP model and creates a sparse approximation of the belief tree through heuristic search.

Default Policy and Value Bounds In order to perform the heuristic search within a short planning time, the R-DESPOT additionally requires a default policy π_{R0} and approximate bounds on the value of a belief. For pre-computation of these inputs, ADACORL first arrives at a MDP corresponding to the robot's POMDP by assuming that the states are fully observable. Next, an optimal policy of this MDP, denoted as $\pi_{R,\text{MDP}}(a_R|s_c)$, is computed using value and policy iteration. Note that the MDP policy maps states, as opposed to belief, to the actions.

Following Ye et al. [171], ADACORL uses the mode-MDP policy to obtain the default POMDP policy (which maps beliefs to actions) from the MDP policy, i.e.,$\pi_{R0}(b) = \pi_{R,\text{MDP}}(a_R|s_c^*)$, where $s_c^* = \arg\max_{s_c \in S_c} b(s_c)$. The value of the MDP policy is used as the upper bound value for the state. The upper bound value for a belief is computed as the expectation of the MDP value under the belief, while the lower bound value is computed at planning time using forward simulations.

5.5.2 Interleaving Planning and Execution

During interaction, the outcome of the robot's action and the corresponding observation is received a timestep after the robot selects its action. Thus, to interleave planning and execution, ADACORL uses a variant of the R-DESPOT algorithm with modification in the belief tree construction and action selection.

While creating the sparse belief tree, R-DESPOT uses all of the available actions to expand the root belief node. In contrast, ADACORL only includes the previously selected action to expand the root node, as the action at the root node is known. At the end of planning time, when a new observation is received, ADACORL performs action selection using the constructed tree. It first identifies the child of the root node that has support for the new observation, and use the best action of this child node as the robot's action.

By interleaving planning and execution, this variant of the R-DESPOT algorithm is capable of making decisions during the execution of the collaborative task. As we discuss next, consequently, ADACORL can generate fluent human-robot collaboration despite the short planning times available during interaction.

5.6 Experiments

I evaluate the performance of ADACORL's model specification approach using two collaborative tasks, both with state spaces significantly larger than prior art. I conduct experiments both in simulation and with human participants.

I hypothesize that the hybrid learning approach of AdaCoRL, despite fewer number of inputs, can generate robot behavior that accrues equal or higher reward as compared to a supervised approach in human-robot collaborative tasks.

5.6.1 Human-Robot Collaboration Scenarios

As the two collaborative tasks, I utilize the shared workspace and human-robot handover tasks described in Sec. 5.2. Briefly, in both the tasks, the human is bringing ingredients from one or more cabinets (pantries) and making sandwiches on an analogue kitchen table with multiple cooking and/or wrapping stations.

Shared workspace task The shared workspace task includes one cabinet, and the human performs the task sitting down. The kitchen table is the shared workspace, and includes two wrapping stations and two cooking stations. In the shared workspace task, the robot is tasked with pouring drinks in four cups on the table, while the human is making sandwiches. The robot needs to finish pouring drinks as soon as possible while maintaining a safe distance with the human.

Handover task The handover task is conducted in a larger environment with three cabinets. The human has to walk to fetch items from the cabinet. In this task, there are two cooking stations and no wrapping stations on the shared kitchen table. The human has multiple ways to finish their task. In this task, the robot is tasked with giving additional ingredients to the human, which they use for making the sandwich. The robot needs to predict when and where the human will be to complete the handover. The task is sequential and, in total, the robot needs to complete three handover in every episode.

For both the tasks, to prevent collisions during task execution, a safety stop is implemented. The robot is stopped if it is within a safety radius (= 0.1 m) of the human or if the human is occluded. A threaded implementation is used for the safety system. Once the stop is triggered, the robot is static until the human leaves the safety radius.

5.6.2 Task Specification

Both the tasks are specified using the MMDP task model, where s_H represents the human's position, s_R represents the robot's joint angles and the progress of its macro actions. For both the tasks, s_E is used to monitor the progress of the respective task. A two-dimensional grid is used to represent the human's position; the size of grid is 12 for the shared workspace task and 16 for the handover task. The human's action space includes motion primitives in the gridworld. The robot's action space consists of motion primitives in the configuration space and task-specific macro actions. The macro actions include *pouring* in the shared workspace task and *handover* in the handover task. The size of s_R state space is $\approx 4,000$ for both the tasks. The task progress, s_E, is modeled with ≈ 30 states.

In total, the shared workspace and handover tasks have $\approx 1.8 \times 10^6$ and $\approx 2 \times 10^6$ states, respectively. The reward function penalizes unsafe executions (i.e., collisions) and emphasizes faster task completion. The timestep of the MMDP and, consequently, the planning time is 0.3 second.

5.6.3 Performance with Simulated Data

I first present the details of the simulation experiments, which were conducted for both the collaboration scenarios. In both the collaboration scenarios, the human had a total of five intents (i.e., $|X_H| = 5$) corresponding to the different landmarks of the kitchen environment (e.g., cooking areas, pantry).

Ground Truth Human Model

In order to generate the human behavior in simulation, I define a ground truth human model with the observable decision factor s_A as the human's position s_H, and the latent decision factor x_A as the human's intent. In order to arrive at the ground truth policy, the simulated human is modeled to follow an intent-driven motion, albeit with bounded rationality. Upon reaching a landmark in the kitchen, the simulated human is constrained to finish the activity at the landmark within a

pre-specified time interval. Since the simulated human always finished the activity at the landmark within a pre-specified interval, the ground truth behavior (i.e., the sequence of decision factors s_A and x_A) is non-Markovian.

The task structure is used to define the intent transitions, e.g., after collecting ingredients from the cabinet the human goes to the table for making the sandwich. The human can finish the task with different types of intent sequences. Execution traces generated using the true model are used to obtain the training and test datasets for the simulation experiments.

Baselines

I utilize a hand-crafted AMM human model and a model learned using supervised learning as the baselines. The hand-crafted model is specified as the Markovian version of the ground truth human model, i.e., without the constraint of finishing each activity within a pre-specified time interval. The ground truth human behavior is used to specify the hand-crafted AMM model; however, the AMM is Markovian. While the true model of behavior is unavailable in practice, it is useful for benchmarking learning approaches in simulation. The hand-crafted model represents those prior approaches wherein the developer manually specifies the robot's decision-making model.

Supervised learning is performed using the algorithm for Supervised AMM Learning described in Sec. 4.5. The supervised approach serves as a proxy for prior approaches that require labeled data of human's latent states and cannot utilize partial specifications. A training dataset of sixteen (s_H, a_H)-sequences and identical hyper-parameters are used for all learning algorithms. Duration of each sequence is $\approx$ 200 timesteps. For supervised learning, in addition, labels of x_H are provided for all sequences (i.e., $\approx$ 3200 labels).

I evaluate two variants of the specification approach of ADACORL: Hybrid-A and Hybrid-B. For both the variants, labels of x_H are provided for only four sequences (i.e., $\approx$ 800 labels). For Hybrid-A, additionally, $\approx$ 10 high-level inputs are provided that encode domain expertise about goal sequences (specified as PS_{1-2})

and minimum time to complete activity at each goal (specified as PS_3). For Hybrid-B, I additionally specify that the next goal only depends on the previous goal and not any other feature (i.e., feature selection specified as PS_4).

In practice, the training dataset of human behavior may not include all behaviors exhibited by the human teammate during collaborative task execution. To test this challenging case, for the handover task, half of the test dataset consists of intent sequences that are absent in the training dataset of human behavior.

Procedure

For each pair of task and learning algorithm, ten AMM models are learned. The AMM model with the lowest test error is chosen as the human model. The learned human model and the task specification are then used to arrive at the robot's POMDP and policy using the third and fourth steps of ADACORL, respectively. Thus, all approaches (hand-crafted, supervised, and hybrid) utilize identical algorithms for computing the decision-making model and the interaction policy.

Metrics

I compare the approaches on both their model specification and interaction planning performance. The model specification performance is quantified using the weighted KL divergence (wKL) between the learned and the true AMM models [109]. The weighted KL divergence measures model alignment (i.e., the distance between the true and learned models). Thirty-two simulation of human-robot collaboration are conducted to evaluate the performance of the learned model for interaction planning.

The interaction planning performance is evaluated using the total shared reward and task-specific metrics – namely, the timesteps for which robot's safety stop is engaged (denoted as Safety Stops) and the number of successful handovers. As the shared reward depends on both human-robot distance and task progress, it provides a composite measure of safety and collaborative efficiency.

Model	Shared Workspace Task			
	$w\text{KL}(T_x)$	$w\text{KL}(\pi_H)$	Reward	Safety Stops
Hand-crafted	0	0	-213.8 $\pm$ 22.2	0.03
Supervised	0.011	0.041	-223.6 $\pm$ 27.4	0.94
Hybrid-A	0.014	0.042	**-184.5 $\pm$ 23.1**	0.09
Hybrid-B	0.015	0.042	-208.9 $\pm$ 25.7	**0.03**

Table 5.1: Objective metrics of model alignment, safety, and collaborative efficiency for collaboration in the shared workspace task (simulation experiments).

Results

The results of the simulation evaluations are summarized in Tables 5.1-5.2. The model specification performance (i.e., the weighted KL divergence) is reported for the best model learned after ten learning trials. The interaction planning performance (i.e., the objective measures of safety and collaborative efficiency) reported in Tables 5.1-5.2 is averaged over the thirty-two trials.

In both the tasks, the models learned through ADACORL's hybrid semi-supervised approach (Hybrid-A and Hybrid-B) have model alignment comparable to that of the supervised approach. As the hand-crafted model is specified using the ground truth model, it has zero KL divergence. Near identical POMDP models can have vastly different policies. Thus, while the modeling performance of ADACORL approach is encouraging, evaluating its planning performance is critical.

Despite fewer labels, models specified using hybrid learning result in near equal or better performance than that of hand-crafted and supervised models. By reasoning about the human's latent state, all model specification approaches are able to complete the shared workspace task with minimal safety stops; the highest average safety stop time is 1 timestep for the supervised model. However, the hybrid semi-supervised model (Hybrid-A) on average obtains higher cumulative reward than both the hand-crafted model and the supervised model. As the ground truth behavior is not Markovian, the robot behavior with the hand-crafted model is not necessary optimal.

	Handover Task			
Model	$w\text{KL}(T_x)$	$w\text{KL}(\pi_H)$	Reward	Handovers
Hand-crafted	0	0	-163.8 $\pm$ 3.0	2.0
Supervised	0.066	0.051	-174.1 $\pm$ 5.4	1.4
Hybrid-A	0.035	0.055	-174.4 $\pm$ 2.6	1.5
Hybrid-B	0.030	0.052	**-168.4 $\pm$ 3.0**	**1.7**

Table 5.2: Objective metrics of model alignment, safety, and collaborative efficiency for collaboration in the handover task (simulation experiments).

Hybrid learning improves generalization. For the handover task, the hand-crafted model performs best. In their failure modes, the models either incorrectly identify human's subgoal or identify the subgoal too late to complete the handover. Among the learned models, we again observe that the hybrid semi-supervised approach (Hybrid-B) on average accrues higher reward and completes more handovers as compared to the model learned via supervised learning. The supervised approach relies on data alone and cannot utilize high-level inputs, which can result in models that overfit to the training data. Generalization to behavior absent in the training set is critical for human-robot collaboration, as the training data will seldom have all possible types of human behavior. In contrast, the hybrid approach can successfully utilize both partially labeled data and domain expertise (partial specifications), thereby improving performance in collaborative tasks.

5.6.4 Performance with Human Participants

In order to conduct experiments with human participants, ADACORL was implemented using a collaborative robot (Universal Robot 10 with a Robotiq gripper). The human-robot team was tasked to perform the handover task, where the human position was sensed using the PhaseSpace motion capture system. The human position measured by the motion capture system was also used for implementing the safety stop. Similar to the simulation experiments, to successfully complete the collaborative task, the human was assigned to make sandwiches and the robot was responsible for delivering the missing ingredient.

Study Description

I utilize a within-subject design, with one independent variable (namely, the approach for specifying the robot's decision-making model) and three treatment levels: hand-crafted, supervised learning, and hybrid semi-supervised learning. Participants for the study were recruited at the MIT campus, and the study protocol was approved by MIT's institutional review board. After obtaining informed consent, participants were briefed about collaborative robot, the safety systems in place, and the collaborative task.

Each participant completed the task six times (two consecutive repetitions for each condition) with the robot. In addition, in order to alleviate novelty effects, the participants performed a training trial at the start of the experiment session. The order of treatments was randomized across participants. The participants were administered a pre-experiment demographic survey, three identical questionnaires during the experiment (one after each condition), and an open-ended survey at the end of the experiment.

Baselines

In order to train the learned models, training data of twelve execution traces was collected from three humans (different from study participants) and manually labeled. Similar to the simulation experiments, the latent state (goal) labels of only 25% of training sequences were provided to the hybrid semi-supervised approach. Labels for the entire training dataset were provided to the supervised approach. The hand-crafted model remained identical to that of simulation experiments.

Ten AMM models were learned for both supervised and hybrid learning. For each approach, the AMM with the lowest test error was chosen as the human model for the experiments. The learned human model and the task specification were then used to arrive at the robot's decision-making model and policy by utilizing the third and fourth steps of AdaCoRL. Thus, all approaches utilized the solver described in Sec. 5.5 for interaction planning.

Collaborative Fluency	
Q_1	The robot was a good partner.
Q_2	The robot and I worked well as a team.
Q_3	I am dissatisfied with how the robot and I worked together.
Q_4	The robot perceived accurately what my goals were.
Q_5	I trusted the robot to do the right thing at the right time.
Safety	
Q_6	The robot kept a safe distance from me.
Q_7	The robot got in my way.
Q_8	The robot moved too fast.
Q_9	I felt uncomfortable working so close to the robot.

Table 5.3: Subjective measures of safety and collaborative fluency used in the experiments with human participants.

Dependent Measures

I compare the cumulative reward and number of handovers for the three models. Since the ground truth model of the human participants is unavailable, the KL-divergence-based metrics of model alignment cannot be computed. Instead, in addition to the objective metrics, I utilized the three in-experiment questionnaires to measure participant's subjective perception of collaborative fluency and safety. These questionnaires included nine items each administered on a seven-point Likert scale. The subjective measures were informed by established metrics of collaborative fluency [52] and are listed in Table 5.3.

Results

I next report the results of the study from interaction with 9 participants (5 female, 4 male, median age: 29 years). Two participants reported prior experience with robots. The objective and subjective measures from the study are reported in Tables 5.4 and 5.5, respectively. I evaluate the effect of the independent variable on the dependent measures using nonparametric statistical tests. First, I apply the Friedman test to verify that the modeling approach had a significant effect, and then used the Wilcoxon signed-rank test to perform pairwise comparisons.

	Handover Task	
Model	Reward	Handovers
Hand-crafted	-329.7 $\pm$ 12.0	2.2
Supervised	-315.7 $\pm$ 20.6	1.4
Hybrid	**-277.7 $\pm$ 10.5**	**2.9**

Table 5.4: Objective metrics of safety and collaborative efficiency for collaboration in the handover task (experiments with human participants, $N = 9$).

Hybrid learning outperforms the hand-crafted model during interaction with human participants. I first compare the objective criterion stated in the problem definition, i.e., the cumulative shared reward R. Averaged across the participants, the hybrid approach accrues a higher cumulative reward than both the baselines. The effect of the modeling approach is statistically significant ($p < 0.05$) as evaluated by the Friedman test, which rendered a Chi-square value of 7.1. Next, I conduct pairwise comparisons using the Wilcoxon signed-rank test. The shared reward of the hybrid approach is statistically significantly higher when compared with that of the hand-crafted model ($p < 0.01$). While a hand-crafted model may perform well in simulation, in practice, it may fail to capture the variance in behavior of human participants. However, when combined with data, the developer's knowledge can result in a useful model that generates fluent collaboration.

Hybrid learning results in more successful handovers than both the baselines. Similar results are found for the task-specific metric of number of handovers. On average, the hand-crafted model resulted in 2.2 handovers, the supervised model in 1.4 handovers, and the hybrid semi-supervised model in 2.9 handovers. The Friedman test rendered a Chi-square value of 11.6 indicating statistical significance ($p < 0.01$). Using the Wilcoxon signed-rank test for pairwise comparisons, I confirm that the hybrid approach completes statistically significantly higher number of handovers than both the hand-crafted model ($p = 0.01$) and the supervised model ($p < 0.01$). Similar to the simulation experiments, the supervised model completes the fewest handovers.

		Handover Task		
		Hand-Crafted	Supervised	Hybrid
Collaborative Fluency				
Q_1	The robot was a good partner.	4 ($p < 0.05$)	2 ($p < 0.01$)	**6**
Q_2	The robot and I worked well as a team.	3 ($p < 0.05$)	2 ($p < 0.01$)	**6**
Q_3	I am dissatisfied with how the robot and I worked together.	5 ($p < 0.05$)	6 ($p < 0.01$)	**2**
Q_4	The robot perceived accurately what my goals were.	3 ($p < 0.05$)	2 ($p < 0.01$)	**6**
Q_5	I trusted the robot to do the right thing at the right time.	4 ($p < 0.05$)	3 ($p < 0.01$)	**6**
Safety				
Q_6	The robot kept a safe distance from me.	6 ($p < 0.1$)	**7** ($p < 0.05$)	5
Q_7	The robot got in my way.	2	**1** ($p < 0.05$)	3
Q_8	The robot moved too fast.	2	1	1
Q_9	I felt uncomfortable working so close to the robot.	2	**1** ($p < 0.05$)	2

Table 5.5: Subjective metrics of safety and collaborative fluency for collaboration in the handover task (experiments with human participants, $N = 9$).

Likely reasons for the observed performance of supervised learning include its inability to utilize partial specifications and over-fitting to the supervised training data. These results further demonstrate that, in practice, it is difficult to rely on either manual specification or supervised data alone to generate useful models of decision-making for human-robot collaboration.

Collaboration achieved using hybrid learning is subjectively perceived to be more fluent than that achieved with either baselines. The median responses for the subjective measures of safety and collaborative fluency are summarized in

Table 5.5. The subjective measures were evaluated on a seven-point Likert scale. A higher response on Q_1, Q_2, Q_4, and Q_5 indicates higher collaborative fluency; in contrast, a lower response on Q_3 indicates higher collaborative fluency. Similarly, a lower response on Q_7, Q_8, and Q_9 indicates higher perceived safety; in contrast, a higher response on Q_6 indicates higher perceived safety.

In order to evaluate statistical significance, similar to the objective metrics, first a nonparametric Friedman test is conducted. If the Friedman test indicates statistical significance for the independent variable, then pairwise comparisons are conducted between the hybrid approach and the two baselines using the Wilcoxon signed-rank test. Table 5.5 also lists the p-value of the pairwise comparisons that are found to be statistically significant.

Through the responses for items measuring collaborative fluency (Q_{1-5}), we observe that the human participants subjectively assess the interaction with a robot utilizing hybrid learning to be statistically significantly more fluent than the interaction with a robot utilizing either of the baselines. For instance, for the item "The robot and I worked well as a team.", the median response for the hybrid approach was 6 (on a Likert scale of 7); whereas, the median responses for the hand-crafted and supervised approaches were 3 and 2, respectively. Similar responses are observed for the other items measuring collaborative fluency, where the pairwise difference in responses is both large (between 2 and 4 points on the seven-point Likert scale) and statistically significant ($p < 0.05$ for the hand-crafted model and $p < 0.01$ for the supervised approach).

Due to the implementation of the safety stop, the human-robot interaction was guaranteed to be safe. Since in each condition the robot utilized identical speed and a safety stop, the participants perceived the interaction for each condition to be safe (i.e., median responses of more than 4 for Q_6 and that of less than 4 for Q_{7-9}). However, through the subjective measures of safety (Q_{6-9}), we observe that the supervised approach was perceived to be the safest by the participants. Pairwise comparisons indicate the difference to be statistically significant $(p < 0.05)$ but relatively small (between 1 and 2 points on the seven-point Likert scale).

A likely explanation for the subjective responses for the supervised approach can be gleaned from the robot behavior observed during its failure modes, wherein the robot either incorrectly identified the human subgoal (leading to the robot waiting at a location far from the human) or identified the subgoal too late to reach the human and complete the handover. Consequently, the supervised approach resulted in marginally safer but significantly inefficient (e.g., fewest handovers) and less fluent collaboration. In contrast, the hybrid approach leverages the flexibility available in the task safety to optimize the shared reward (a composite measure of safety and efficiency) leading to safe, efficient, and fluent collaboration.

Experiments with human participants confirm the utility of ADACORL. In summary, the evaluations confirm that the hybrid model specification approach of ADACORL, despite 75% fewer labels, performs equally well or better than both a hand-crafted model and a model learned with fully-labeled behavioral data. Further, by leveraging ADACORL's online approach to interaction planning, the robot is able to exhibit the desired collaborative behavior by reasoning about the task's large state space within a planning time of 0.3 s.

5.7 Summary

In this chapter, we discussed ADACORL, a novel framework to specify decision-making models and generate robot behavior for human-robot collaboration. In prior art, labeled datasets of human's latent states have been a prerequisite to generate fluent robot behavior for collaboration. ADACORL's hybrid approach to model specification relaxes this requirement by learning decision-making models with partially labeled data and domain expertise. I demonstrate ADACORL in two human-robot collaborative tasks, with state spaces significantly larger than those considered in prior art. ADACORL's online approach to interaction planning enables the robot to make decisions in these tasks, despite their large state spaces and short planning times.

By leveraging both data and domain expertise, ADACORL addresses Problem 2 and enables *effective information transfer* from the developer to the robot. Specifically, it reduces the developer's specification effort by utilizing a modular task representation, hybrid learning of the human decision-making model, analytical computation of the robot decision-making model, and algorithmic generation of the robot's interaction policy.

Promising directions of future investigations include expanding the vocabulary of ADACORL to consider additional tasks (i.e., beyond the Markovian task model) and additional high-level specifications (i.e., in addition to the partial specifications PS_{1-4}). Another avenue is to leverage approaches that infer task objectives [18, 21, 50, 136] and further reduce the developer effort for specifying the task model and creating collaborative machines.

Chapter 6

Deciding to Communicate during Collaborative Task Execution

In the previous two chapters, I provide solutions for effective information transfer from the developer to the collaborative robot. These solutions enable the developer to specify the model of the human teammate and policy of the collaborative robot. In this chapter, we shift our attention to the information sharing between the teammates (i.e., the human and robot) during collaboration.

Recognizing the utility of information sharing, a significant amount of research focusing on human-robot communication has been conducted in the last two decades [150, 93, 21]. As detailed in Sec. 2.5, several communication modalities are being developed for enabling sharing of information among humans and robots [12, 92, 149, 4, 32, 72, 146]. Depending on the collaboration context, these modalities can enable robots to either convey information to the human, interpret the information received from the human, or both.

Despite the active focus on the development of communication modalities, however, there is a relative lack of approaches that enable a robot to purposefully utilize its communication capability during interactions with humans [93]. While modalities are a prerequisite for enabling communication, purposeful planning and use of the communication modality is essential for achieving the motivation behind developing these modalities, namely, fluent collaboration.

Moreover, as motivated in Sec. 2.5, while information sharing has the potential to improve team performance, it often has associated costs. These costs may arise due to the power requirements necessary to transmit data, computational requirements associated with processing new data, or the limitations of human information processing. Consequently, the benefits gleaned from using newly communicated information may not necessarily outweigh the associated costs, and excessive communication can hamper collaborative performance.

Thus, assuming the presence of a communication modality, this chapter focuses on the challenge of its effective use for human-robot collaboration (i.e., Problem 3). Specifically, I provide a novel framework that enables a collaborative robot to make decisions regarding not only actions but also *communications*. Encouraged by the performance of ADACORL, the framework utilizes a hybrid approach to model specification and an online approach to decision-making.

This chapter begins with an overview of the communication decision-making framework, followed by a review of related work. Before describing the framework in detail, I summarize the collaborative tasks and communication modalities of interest. The framework is then demonstrated on a human-robot collaborative task, where the teammates can communicate using speech. Through experiments with human participants, I confirm that the proposed framework can generate fluent human-robot collaboration through effective use of communication.

6.1 Framework for Specifying the Communication Policy

The primary contribution of this chapter is a principled framework for answering the question "If, when, and what to communicate?" for a given collaborative task and communication modality. The collaborative tasks and communication modalities of interest are described in Sec. 6.3 and Sec. 6.4, respectively. Figure 6-1 provides a visual summary of the communication decision-making framework.

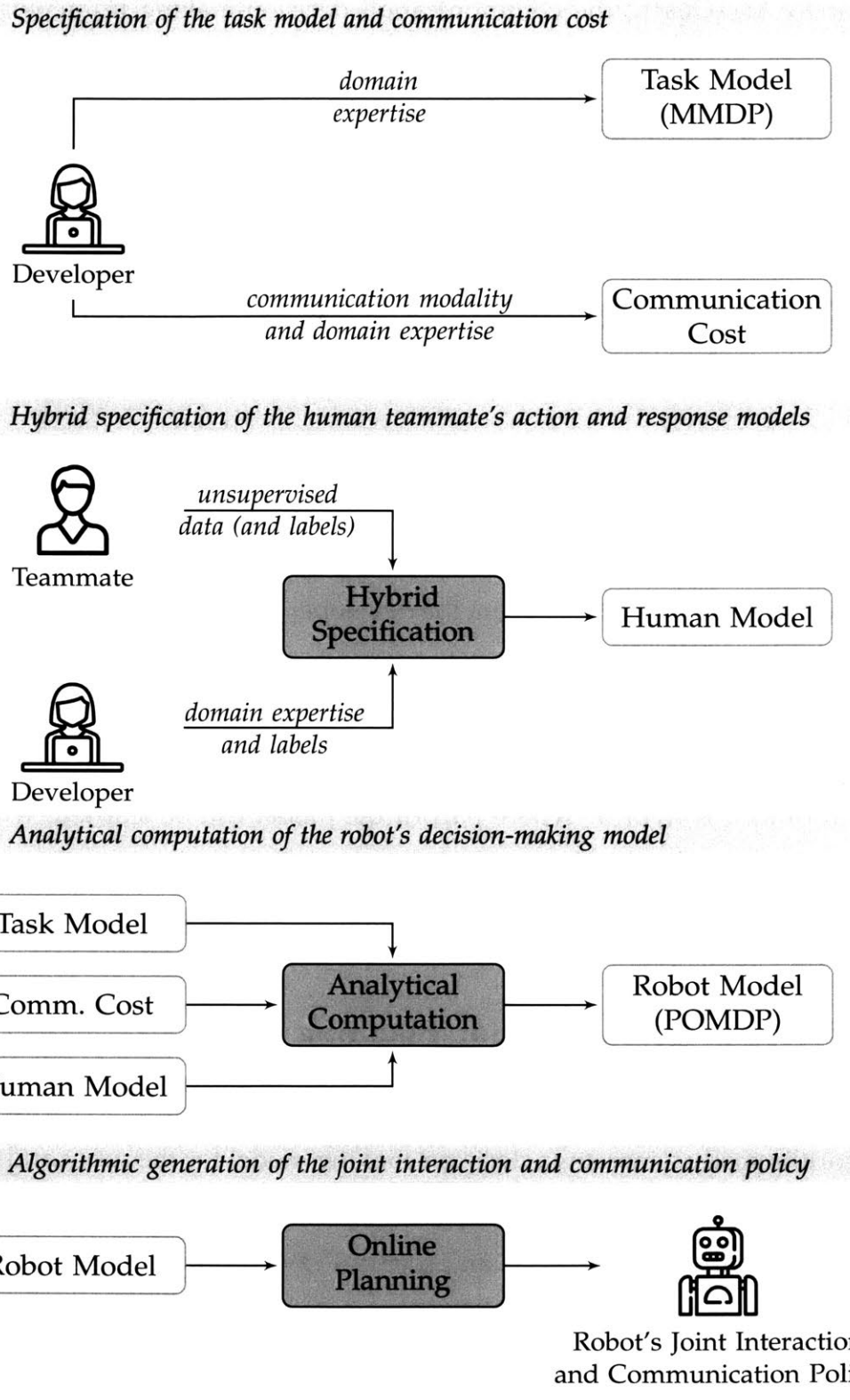

Figure 6-1: A hybrid framework for generating a robot's joint interaction and communication policy for a human-robot collaborative task.

Similar to AdaCoRL, the communication decision-making framework consists of a model specification process and an execution-time algorithm to address the problem of computing the robot's policy. A detailed problem formulation is provided in Sec. 3.4. Both the physical actions and communications of the robot affect the progress of the collaborative task and human's mental states, rendering the choice of the effective communication interdependent on the robot's physical actions. Thus, the communication decision-making framework jointly reasons about robot's actions and communications to arrive at the robot's policy.

Model Specification and Learning

The model specification process, described in Sec. 6.6-6.8, leverages the opportunity of *effective information transfer* from the developer to the robot. The robot developer needs to specify three modules: a task model, a model of human teammate's behavior, and the cost of using available communication modalities. Recognizing the difficulty of collecting communication-based interaction data, the model specification approach is hybrid: where part of the model can be learned, while the remainder can be manually specified. This hybrid specification approach builds upon the algorithms for learning teammate models described in Chapter 4.

Interaction and Communication Planning

Similar to AdaCoRL, upon completing the model specification process, the framework algorithmically generates a decision-making model for the robot and its policy. In order to reason about human's mental states during decision-making, it uses a partially observable Markov decision process (POMDP) as the representation for the robot's decision-making model. The robot's policy, which specifies its action and communication decisions, is computed at execution-time using a variant of DESPOT [171]. As detailed in Sec. 6.11, via a human-participant study, I demonstrate the utility of the framework in a sequential human-robot collaborative task with multiple communication types and short planning times.

6.2 Related Approaches for Communication Decision-Making in Human-Robot Teams

Analogous to the research on human teams, multiple human-robot interaction user studies have helped identify the utility of communication between humans and robots for fluent collaboration — e.g., by contributing to the agents' shared situational awareness and engendering trust among teammates [131, 143, 164, 138]. These studies, which use a hand-crafted communication policy, serve as a motivation for the development of computational approaches for effective human-robot communication.

In this section, I review related work on computational approaches to arrive at the robot's communication policy. Kaupp et al. provide one of the first computational approaches for communication decision-making [67]. Their probabilistic approach addresses the case where the robot has partial observability of its environment, and through its communication modality could request information/help from the human. The efficacy of the communication decision-making is confirmed through user studies.

Mavridis and Dong integrate planning for motor control and speech to address the question "to ask or to sense?." Similar to [67], they consider tasks where the robot is requesting help from the human. In contrast to these related works, this dissertation considers mixed-initiative tasks where both the human and the robot are tasked for completing the collaborative task.

Communication during mixed-initiative tasks has also been considered in recent research [19, 105, 164]. Timed Petri nets have been used to develop a general framework for planning interaction resources (including communication) for human-robot interaction [19]. Wang et al. provide an approach to generate communication vocabulary for robots operating in partially observable environments and use it to generate explanations during human-robot collaboration. However, these approaches do not explicitly model the impact of human's latent states on robot's communication decision-making, or vice versa.

The approach closest to the framework presented in this chapter is that of Nikolaidis et al., which explicitly considers the effect of human's latent states on communication decision-making. Their approach introduces compliance, an important parameter for modeling the impact of robot communications on human's behavior, and performs planning using a mixed observable MDP. However, their approach considers only a single communication type (either state-conveying actions or commands) during task execution. In contrast, the framework presented in this chapter can reason about multiple communication types, thereby enabling planning for bidirectional human-robot communication. Further, the framework extends the formalization of compliance, and is demonstrated using a decision-making problem with large state space and short planning time.

6.3 Specification of the Collaborative Task

In this chapter, I focus on human-robot collaborative tasks that can be modeled by the factored, MMDP task model described in Sec. 5.2. Further, I assume that the robot developer has complete knowledge of the parameters of the task model. Consequently, the task specification in the communication decision-making framework is identical to the first step of ADACORL. The task model does not include a specification of human's mental states or robot's communication modality.

While the human and robot have full observability of the task state s, they cannot observe each other's mental states (such as intent) during task execution. Through the effective use of communication, the robot can better infer human's mental states and convey information to the human, thereby improving the collaboration. For describing the proposed communication decision-making framework, I use a variant of the shared workspace task described in Sec. 5.2.3 as the running example. However, the discussion and the proposed framework is general in that it applies to other collaborative tasks that can be represented by the MMDP task model. Figure 6-2 depicts an example of human-robot communication from the shared workspace task.

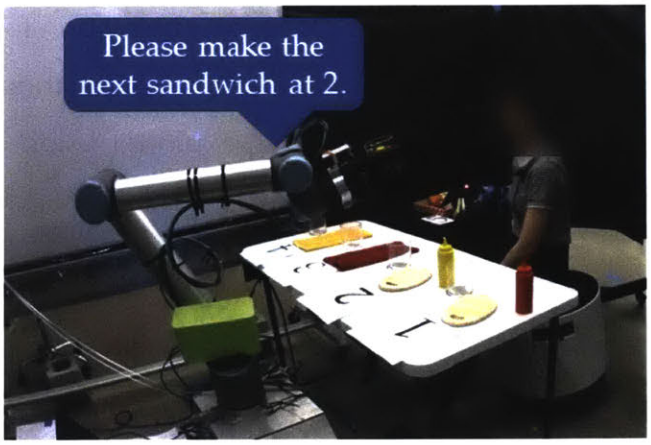

Figure 6-2: Robot communicating with the human teammate to facilitate collaboration in a shared workspace.

6.4 Specification of the Communication Capability

As discussed in Sec. 6.2, several modalities are being actively developed for human-robot interaction and the utility of effective communication has been identified across multiple user studies. In order to design an approach that applies across modalities, the design of the communication decision-making framework is centered around communication types (which specify the information being communication) instead of communication modalities (which specify how the information is being communicated). In this section, I discuss the different communication types for which the framework enables communication decision-making.

6.4.1 Communication Types

Guided by the communication context, multiple taxonomies for communication types have been developed across research on human-human, robot-robot, and human-robot communication [10, 27, 169, 164]. For instance, dialog and speech acts enable classification of natural language into different communication types [10, 27, 113]. Similarly, for a robot utilizing a POMDP for its decision-making, Wang et al. provide and study the relative utility of different communication types. In their taxonomy, communication types are classified based on the elements of the POMDP that the communication conveys.

For multi-agent systems, Xuan et al. provide an abstraction for communication decision-making, which includes three types of communication: *tell*, *query*, and *sync*. Similar communication types have been explored using Blocks World for Teams, a widely-used testbed for joint activity and teamwork, namely: *request*, *tell*, *inform*, and *answer* [59, 82, 165, 17]. Informed by these abstract types, in this chapter, I consider the following types of communications:

- `inform`, the robot informs the human about a latent state of its decision-making; since the task state is observable in our model, the inform message corresponds to the robot's mental state (such as robot's intent, denoted as, s_{RI});
- `ask`, the robot queries the human about the latent state of her decision-making; since the task state is observable in our model, the ask message corresponds to a question regarding the human's mental state (such as human's intent); and
- `command`, the robot requests the human to perform a specific action or plan; this communication type can also be used to represent requests or suggestions; and
- `answer`, the robot answers the human's question.

The proposed framework does not explicitly consider the type *answer* in its decision-making approach. Instead, it assumes that readily answering the human's queries is optimal in collaborative applications, thereby circumventing any need for decision-making for *answer*. The first three communication types, coupled with the above policy for answering human queries, enables the framework to perform decision-making for bidirectional human-robot communications.

I demonstrate the framework using speech as the communication modality. However, the framework is equally suitable for other modalities that map to the above communication types. For instance, the communication type `inform` can be achieved by a robot sharing its intention using visual signals or gestures. Similarly, a robot can request a human to complete a task using augmented reality. Finally, I view symbol grounding as an important but complementary problem. Hence, the framework assumes that both the human and the robot can associate the communications to the attributes of the collaborative task without any ambiguity.

6.4.2 Communication Space

Based on these communication types, the robot developer needs to specify a set of communications that the robot and the human can make during the task execution. I denote the finite set of robot's communications $a_{RC} \in A_{RC}$ and that of human's communications $a_{HC} \in A_{HC}$ as the communication space. Along with specifying the communication space (i.e., the set of robot's actions and communications), the developer has to specify grounding for these communications using the variables (state and action) of the robot's decision-making model.

Example In the example task, sharing information regarding the agent's intent can be useful. Thus, I specify the following set of robot communications $a_{RC} \in A_{RC}$ using task-specific actions and key landmarks of the shared workspace,

- `(inform)` "I am going to do *action* at *landmark*."
- `(ask)` "Where are you going?"
- `(command)` "Please make the next sandwich at *landmark*."

Similarly, to model the human's response, the set of human communications $a_{HC} \in A_{HC}$ is specified as "I am going to *landmark*." The human can provide this information as response to the robot's question (i.e., as the type `answer`) or offer it on her own accord (i.e., the type `tell`). Grounding for the key landmarks (i.e., cooking stations, wrapping stations, pantry) is manually specified using the MMDP state variables (specifically, positions of the human s_H and joint angles of the robot s_R) to enable communication understanding. Similarly, action grounding is available from the MMDP action space.

6.5 Need for Robot's Decision-Making Model

The objective of the framework is to arrive at the optimal robot policy π_R^* (i.e., a mapping from robot's sensory information to its actions a_R and communications a_{RC}) for the specified collaborative task and communication space. As detailed in

the problem statement (see Sec. 3.4), optimality is quantified using the expected cumulative reward of the human-robot team. However, arriving at the robot's policy is not readily possible given just the task model and communication space.

The collaborative task model of Sec. 3.1.2 depends on the team's joint actions (i.e., actions of both the human and the robot). Hence, similar to Chapter 5, to computationally solve for the robot's policy, the framework needs a single-agent model that only depends on robot decisions (i.e., robot actions a_R and communications a_{RC}). Here, I summarize the steps involved in specifying this single-agent model, which is referred to as the robot's decision-making model.

In order to streamline the process of specifying the robot's decision-making model and reduce the effort of robot developer, the framework requires specification of following modules, namely, a model for communication cost; a model for human's action decision-making (i.e., policy for a_H); and a model for human's communication decision-making (i.e., policy for a_{HC}).

These modules can be specified either entirely manually, learned from data, or both. In order to reduce the developer effort, learning these models from data, when feasible, is desirable. However, collecting communication-based interaction data (which is essential for learning approaches) is challenging in the real world. Hence, a hybrid approach – where part of the model is learned, while the remainder is manually specified using the developer's domain knowledge – is emphasized. Next, I describe the modules and, by providing their instantiation for the example task, demonstrate the specification process in action.

6.6 Model for Communication Cost

As noted in the introduction, while communication has the potential to improve interaction, it also incurs cost. Research on human teams and interruption management has identified varies latent causes of communication cost for humans [56, 87]. Consequently, communication cost is often task- and context-specific, and novel cost models are needed for human-robot communication.

6.6.1 Cost Model

In order to enable communication decision-making, I provide a parametric model to specify communication costs. For a particular collaborative task, the developer can specify these parameters based on domain knowledge to inform robot's decision-making. In order to stress the cost of too many communications, the proposed cost model is non-linear. Specifically, the model requires three parameters: minimum and maximum cost of communication (ρ_a, ρ_b), and the duration of the interruption interval h_{RC}. The communication reward (negative cost) R_c is then given as follows:

$$R_c = -\rho_a \exp\left[\left(1 - \frac{t_{RC}}{h_{RC}}\right) \log \rho_b\right] \tag{6.1}$$

where t_{RC} denotes the time since the last communication. The interruption interval begins after a communication is made and lasts for a duration of h_{RC}. For a communication outside this interval, the cost corresponds to the minimum ρ_a. The model penalizes multiple communications during the interruption interval.

Depending on the task, the cost could be tailored to different communications by specifying different parameters for each a_{RC}. For instance, to emphasize polite communication during equal partners teamwork, a developer could specify a higher cost for commands as compared to suggestions. Note that this model is just one alternative to specifying cost and highlight that modeling human-robot communication cost is an important but under-explored problem. However, irrespective of the model choice, I posit that the nonlinear penalization of communication is important when the team is performing sequential tasks.

Example For the example task, I specify $h_{RC} = 3$ s. The minimum and maximum cost of communications is set as $\rho_a = 1$ and $\rho_b = 15$. Finally, I tailor the cost model for `inform`, in which the robot commits to an action and landmark (i.e., intent). In order to emphasize that the robot follows through its commitment, denoted as s_{RI}, the model includes a penalty if the robot does not complete its communicated commitment and a positive reward (negative cost) if it does.

6.6.2 Communication State

In order to enable Markovian decision-making with a temporal model of communication cost, additional communication-specific states are required. I denote this communication state as $s_{RC} \in S_{RC}$. This state includes the latest communication, the time since last communication t_{RC}, and the robot's communicated intent s_{RI}.

6.7 Model for Human's Action Decision-Making

The need and utility of modeling the human teammate's mental states and anticipating her actions have been demonstrated across multiple studies of human-robot interaction [150]. Thus, a model of human's task-specific behavior is essential for making effective communication decisions during human-robot collaboration. Multiple data-driven approaches have been utilized to learn this model, including variants of supervised learning and inverse reinforcement learning. Further, in Chapter 4, I provide a novel representation and a suite of learning algorithms which reduce the developer's effort for modeling the human teammate.

Building upon the solutions presented in Chapter 4, the framework for communication decision-making utilizes a hybrid specification approach for modeling the human teammate. Specifically, in contrast to prior art that relies data-driven learning alone, this hybrid specification approach enables the developer to hand-craft part of the model for which training data might be insufficient or unavailable.

The ability to combine data and domain expertise is especially relevant for specifying models of communication decision-making. In practice, it is difficult to collect interaction data for training models, even more so when the interaction involves communications between human and robot. Further, collecting training data for communication-based interaction requires as input a developer-specified communication policy. However, training with a substandard communication policy is also undesirable, as it can negatively impact the user's trust and subjective perception of the collaborative robot.

6.7.1 Agent Markov Model

In order to represent human's sequential decision-making behavior, the framework utilizes the Agent Markov Model (AMM) described in Chapter 4 and used for modeling the human teammate in Chapter 5. AMM represents the behavior of an agent whose decisions are contingent on both observable (s_A) and latent (x_A) decision factors, and the dynamics of the decision factor are Markovian.

The choice for using the AMM is motivated by the need to model human's mental states and their dynamics (for which the AMM includes explicit parameters). Further, as described in Chaper 4, algorithms exist for learning the AMM that can utilize both data and domain expertise. Briefly, the AMM describing human behavior is defined by the tuple $(X_A, S_A, A_A, b_x, T_x, T_s, \pi_A)$, where

- $X_A \equiv X_H$ denotes the state space of human's mental states x_H;
- S_A denotes the space of human's observable states s_A;
- $A_A \equiv A_H$ denotes the set of human's task-specific actions a_H;
- b_x denotes the probability of the initial latent state;
- T_x and T_s denote the transition models of x_H and s_A; and
- $\pi_A \equiv \pi_H \equiv \pi_H(a_H|s_A, x_H)$ represents the human's policy, i.e., the probability of choosing action a_H in the state (s_A, x_H).

AMM Feature Specification In a collaborative task, the human's behavior depends on the task-specific features (i.e., the MMDP state s), mental states, actions, and communications. The robot has full observability of the MMDP state, its own actions, and communications. Hence, the specification of the human's observable decision factors s_A and their dynamics T_s is obtained from the MMDP state, robot actions, and communications, i.e., $s_A = g(s, a_R, s_{RC}, a_{RC})$ and $s_A \subseteq S \times A_R \times S_{RC} \times A_{RC}$. The function g is an arbitrary nonlinear function, which provides the developer flexibility while specifying observable features important for human behavior. AMM's action space A_H is also available from the task MMDP. Thus, the specification of AMM's known parameters follows in a fashion similar

to that of ADACORL. However, to account for effect of robot's communications, a_{RC} and s_{RC} are also included as the known decision factors. The parameter X_H enables the developer to specify important latent features (i.e., mental states corresponding to variables such as goal, intent, belief, and workload) that affect human behavior. Thus, using the task definition and domain knowledge, the developer can partially specify the AMM tuple; specifically (X_H, S_A, A_H, T_s).

Example In the shared workspace task, the behavior of the human is intent driven. The human first picks the ingredients from the pantry, then takes them to the cooking station, and finally wraps and keeps them. Thus, these task-specific intents (or subgoals) are important mental state x_H for human's decision-making. The set of task-specific intents specifies the latent state space X_H of the AMM. In addition to her intent, the human's behavior depends on her position (s_H) as well as robot's communications (s_{RC}, a_{RC}). This completes the specification of both known decision factors s_A as (s_H, s_{RC}, a_{RC}) and x_H as subgoals. The action of human a_H is known from the MMDP definition.

6.7.2 Two-Step Approach for AMM Learning

Given this partial AMM tuple and interaction data (i.e., execution traces of s_A, a_H and optionally x_H), I provide algorithms to recover the complete model of human in Chapter 4. The complete AMM tuple specifies the human's policy π_H and and dynamics T_x of the mental states. For the example task, this corresponds to learning how the human chooses her next subgoal x'_H and actions a_H based on previously completed subgoals x_H, current position s_H, and robot communications a_{RC}. However, as noted earlier collecting communication-based interaction data is challenging in practice. Hence, the frameworks adopts a hybrid two-step approach. First, it learns an interim model that does not capture the influence of robot communication on human behavior; learning of the interim model is done using interaction data without any communication. Next, this model is augmented using developer-specified effects of robot communications on human behavior.

Modeling Human Behavior in absence of Robot Communication

In order to learn the interim model, which does not model communications, the framework requires as input a dataset of interaction data without communications (i.e., execution traces of s_A with $s_{RC} = a_{RC} = \varnothing$, where $\varnothing$ represents null). The dataset can be optionally labeled to include sequences of the latent state x_H. Given this dataset and the partial AMM tuple (X_H, S_A, A_H, T_s), the interim model is generating using algorithms for learning the AMM.

Example In order to learn the teammate model for the example task, I use the dataset described in Sec. 4.9.3. During the data collection, the human was performing the task without any communications from the robot, which resulted in execution traces of (s_A, a_H) with $s_{RC} = a_{RC} = \varnothing$. Given this dataset, the interim model that captures the human's action-selection behavior is obtained computationally via Supervised AMM learning. The learned model includes the dynamics of human's mental state $T_x(x'_H|x_H, s_{F\varnothing}, a_H)$ and action-selection policy $\pi_H(a_H|s_{F\varnothing}, x_H)$, where $s_{F\varnothing} = g(s, a_R, s_{RC} = \varnothing, a_{RC} = \varnothing)$. However, the learned parameters of the interim model T_x and π_H are independent of a_{RC} and, thus, do not include effect of robot communications on human behavior.

Modeling Effect of Robot Communication on Human Behavior

A communication can affect human's behavior either directly (via the policy) or indirectly (by changing the human's mental states, which in turn affects human's choice of actions). Both of these effects can be specified in the proposed framework. For instance, for a communication that requests the human to perform an action, the effect is modeled by the impact of a_{RC} on policy π_H. In contrast, for a communication that requests the human to choose a subgoal, the effect is captured by the impact of a_{RC} on latent state dynamics T_x. Thus, mathematically, the objective of this step is to augment the terms of the interim model with $T_x(x'_H|x_H, s_A, a_H)$ and $\pi_H(a_H|s_A, x_H)$, where $s_A = g(s, a_R, s_{RC}, a_{RC} \neq \varnothing)$ or $s_A = g(s, a_R, s_{RC} \neq \varnothing, a_{RC})$.

Note that T_x and π_H are probability distributions over human's mental states and actions, respectively. Thus, the effect of a_{RC} corresponds to the probability of humans changing their behavior based on robot's communication. This probability is called *compliance*, which has been previously formalized for the communication type `command` [105]. In prior art, compliance is considered identical for all communications and task states. However, in practice, the effect of each communication is different and depends on the task context. I extend the formalization of compliance to other communication types and, through the specification of $s_A = g(s, a_R, a_{RC}, s_{RC})$, allow for it to vary based on the task state (via s) and communication (via a_{RC}, s_{RC}).

Example I provide examples of the specification process for the shared workspace task before summarizing its general version. In the example task, the robot can make three types of communication: `command`, `inform`, `tell`. Due to the task structure, the human will be able to follow the `command` (of making sandwich) only if she currently has access to the ingredients (i.e., human is at pantry). Thus, I specify a context-specific model of compliance for the robot's command, i.e., `command` affects human only if the human position s_H is pantry. Specifically, when the human is at the pantry the human will follow the robot's command with a context-specific compliance probability p_{c1}.

Further, by incorporating communication state s_{RC} as one of human's known decision factors s_A, a developer can specify that a communication has temporally-extended influence on human behavior. This is especially important in sequential tasks and to account for communication delay. For the communication type `inform`, the robot shares its intent (subgoal) with the human. I model that after learning about the robot's subgoal, the human will act to improve fluency of collaboration and not choose the same subgoal with probability p_{c2}. The probabilities p_{c1} and p_{c2} can be different, resulting in a context- and communication- specific compliance model. The `ask` type influences human's communication but not the action-specific behavior and, thus, does not require a compliance model.

Specification Process Thus, in general, for a communication a_{RC} that exerts indirect influence on human behavior, the developer needs to specify a probability distribution over the human's next mental state x'_H when the robot makes the communications a_{RC} for AMM state (x_H, s_A) and action (a_H). I denote this probability distribution as q_c, which is used to augment the model as follows: $T_x(x'_H|x_H, s_A, a_H) = q_c$. The distribution q_c mathematically represents the extended formalization of compliance, which is both context- and communication-specific and can be temporally extended. Further, these tuples need only be specified for the communications that influence human's task-specific behavior. Similar procedure is used for communications with direct influence. However, for this case, the compliance distribution q_c specifies the values of human's policy π_H.

Summary

Given the compliance distributions and the interim model, the specification of human's action decision-making is complete. When the robot is silent (i.e., $s_{RC} = a_{RC} = \varnothing$), the human behavior is specified by the interim model. When the robot communicates, the specifications provided by the developer are used. In tasks where communication-based interaction data is available, the two-step process is not necessary and the complete model can be learned using interaction data.

6.8 Model for Human's Communication Decision-Making

Along with performing the task-specific actions a_H, the human teammate too can communicate during the task. Thus, in addition to a policy of human's task-specific actions, the robot requires a models of human's communications to make effective decisions regarding its own communications. In this chapter, I provide a model for human's communication types `tell` and `answer`, which is then incorporated into robot's decision-making.

In response to the robot's query (type `ask`), the human may respond correctly, incorrectly, or not respond at all. For instance, if the human's attention is overloaded with the collaborative task, the robot's question may go unanswered. The proposed framework for communication decision-making represents this behavior using the property *responsivity*, which is summarized by the probability p_r of the human responding to a query.

In collaborative tasks, the human teammate is seldom expected to provide an incorrect response. However, due to implementation challenges (such as errors in natural language parsing), the robot might receive an incorrect response. Thus, the framework further models that the sensed response from the human is truthful with probability p_t.

Finally, the framework also considers the case when the human may share her intent (i.e., subgoal) with the robot preemptively (i.e., without being asked) with probability p_a. For the example task, I use the following values: $p_r = 0.9$, $p_t = 0.9$, and $p_a = 0.01$. Similar to compliance, in general, the parameters of the response model (p_r, p_t, p_a) may vary based on the communication action, context (i.e., the MMDP state), robot's behavior, and human's mental state. However, I assume that they are constant and leave the investigation of their dynamics for future work.

6.9 Model for Robot's Action and Communication Decision-Making

Given the problem inputs (task model and communication space) and the model specification (communication cost and human models), the robot behavior is generated computationally and without any additional developer effort. Similar to AdaCoRL, in its third step, the communication decision-making framework analytically arrives at a POMDP that serves as the robot's decision-making model. In this section, I describe the analytical derivation of this POMDP model. The POMDP model parameters are denoted using the subscript p.

State Space: S_p denotes the set of robot's decision-making states, $s_p \equiv (s_H, s_R, s_E, s_{RC}, x_H) = (s, s_{RC}, x_H)$. The POMDP state includes five factors, namely, the three components of the task MMDP state s, the communication state s_{RC}, and the latent state of human decision-making x_H. The inclusion of human's latent state enables anticipation of human's actions and generation of adaptive robot behavior, which is an essential feature for fluent collaboration [53].

Action Space: $A_p \equiv A_R \cup A_{RC}$ denotes the set of actions a_p, where A_R and A_{RC} denote the sets of robot's task-specific actions a_R and communications a_{RC}, respectively. Thus, the planning is done jointly for robot's actions and communications.

Transition Model: The state dynamics for the POMDP are Markovian and are denoted by $T_p(s'_p|s_p, a_p) : S_p \times A_p \times S_p \rightarrow [0, 1]$. For the factored state of the POMDP, the transition model is derived as follows,

$$T_p(s'_p|s_p, a_p) \equiv T_{ps}(s'|s_p, a_p) T_{pc}(s'_{RC}|s_{RC}, a_{RC}) T_{px}(x'_H|x_H, s_A) \tag{6.2}$$

The transition model of the variable s is available from the task model, $T(s'|s, a)$. However, the task model depends on the joint action $a \equiv (a_H, a_R)$. The robot's decision-making model requires a transition function that only depends on the robot actions and communications. Thus, using the human model (AMM) and the task model (MMDP), the transition model of the variable s for robot decision-making is derived analytically as follows,

$$T_{ps}(s'|s_p, a_p) = \Sigma_{a_H \in A_H} T(s'|s, a_R, a_H) \pi_H(a_H|s_A, x_H) \tag{6.3}$$

The variable $s_A = g(s, a_R, s_{RC}, a_{RC})$ is a function of the POMDP state and actions, thus by summing over a_H, the framework arrives at a model that only depends on the POMDP state s_p and action a_p. This analytical computation assumes that the communications do not affect the MMDP state directly. However, they may exert an indirect influence by affecting the human teammate's behavior.

The transition model of human's mental state x_H is also available from the AMM; however, it too depends on human actions. Thus, the frameworks analytically obtains T_{px} as follows,

$$T_{px}(x'_H|x_H, s_A) = \Sigma_{a_H} T_x(x'_H|x_H, s_A, a_H)\pi_H(a_H|s_A, x_H) \tag{6.4}$$

Finally, T_{pc} is the transition model of the robot's communication state that keeps track of the robot's communicated intent, latest communication, and time since last communication.

Reward Model: The reward $R_p(s_p, a_p) : S_p \times A_p \rightarrow \mathbb{R}$ is obtained by adding two components: the task reward R_{ps} and the communication cost R_{pc}. Specifically,

$$R_p \equiv R_{ps}(s_p, a_p) + R_{pc}(s_p, a_p) \tag{6.5}$$

Similar to the computation of T_{ps}, the task reward R_{ps} is obtained analytically using the human model (AMM) and the task model (MMDP),

$$R_{ps}(s_p, a_p) = R_{ps}(s, s_{RC}, x_H, a_R, a_{RC}) \tag{6.6}$$

$$= \Sigma_{a_H \in A_H} R(s, a_H, a_R)\pi_H(a_H|s_A, x_H) \tag{6.7}$$

The model for communication cost is described in Sec. 6.6, i.e., $R_{pc} = R_c$.

Observation Model: During the task, the robot has full observability of the MMDP state s and its own communications s_{RC}. Hence, the human's mental state x_H is the partially observable component of the POMDP. The observation model of x_H is specified as the human's model for communication decision-making, which is descried in Sec. 6.8. Thus, through its communication the robot can possibly influence human behavior and receive observations of the latent state.

Finally, the time horizon t_f and the discount factor γ of the robot's decision-making model (POMDP) are identical to that of the task model (MMDP).

6.10 Algorithm for Robot's Action and Communication Decision-Making

Given the robot's decision-making model, the robot's policy is generated algorithmically by *solving* the POMDP. Similar to ADACORL, to enable decision-making for problems with large state spaces and short planning times, the communication decision-making framework utilizes the variant of DESPOT [171] described in Sec. 5.5. The planning is done during interaction and jointly over robot's actions and communications. The planning time available during interaction is often small, thereby making the planning further challenging. As demonstrated next, the framework can make decisions at execution-time in problems with large state spaces, short planning times, and multiple communication types.

6.11 Experiments

In order to evaluate the utility of the communication decision-making framework, I implemented the framework on a collaborative robot and conducted experiments with human participants. Similar to the experiments presented in Chapter 5, Universal Robot 10 (with a Robotiq gripper) was used as the collaborative robot. To the best of my knowledge, my framework is the first to perform online decision-making for multiple communication types while anticipating human's behavior and latent states. Hence, I compare it against a hand-crafted policy for robot's communication decision-making.

Specifically, through the experiments with human participants, I evaluate the following two hypotheses: *During human-robot collaboration, a robot that reasons about its multiple communications using the communication decision-making framework obtains a shared reward higher than*

H1 *a robot that does not communicate*; and

H2 *a robot that uses a hand-crafted communication policy.*

6.11.1 Study Description

The human-robot team was tasked to perform the shared workspace task, which served as the running example in this chapter, during the experiments. A still from the human-participant study is depicted in Fig. 6-2. Similar to the experiments in Chapter 5, to ensure safety, a safety stop was implemented. Specifically, the robot is stopped if it is within a safety radius (= 0.1 m) of the human or if the human is occluded. Once the stop is triggered, the robot is static until the human leaves the safety radius. A motion-capture system was used for sensing the human.

Task Specification

The shared workspace task is specified using the MMDP task model. A two-dimensional grid is used to represent the human's position (i.e., the state feature s_H). The state feature s_R represents the robot's joint angles and the progress of its macro actions. The state feature s_E is used to monitor the task progress (e.g., which cups have been filled). For the version of the shared workspace task used in the experiments with human participants, the MMDP state space S includes $39,816$ states. The human's action space includes motion primitives in the gridworld. The robot's action space consists of motion primitives in its configuration space and task-specific macro actions (e.g., pouring). In addition to the task's large state space, the robot had to make decisions online within a planning time of 0.3 s.

Communication Capability

The human-robot team used speech as the communication modality. The robot's vocabulary was limited to its communication space, which is specified in Sec. 6.4.2. The robot communicated via a speaker and used a speech-to-text software for detecting human's response [76]. In order to facilitate symbol grounding, numbered labels were placed next to key landmarks in the shared workspace (e.g., at cooking and wrapping stations). The human and robot used the location numbers to refer to the landmarks while communicating.

Baselines

I utilize a within-subject design, with one independent variable (namely, robot's approach to communication decision-making) and three treatment levels:

- no-communication policy (denoted as Silent);
- a hand-crafted communication policy (denoted as Hand-crafted); and
- a policy generated using the communication decision-making framework presented in this chapter (denoted as POMDP-based).

All three approaches utilize identical specification of task model (MMDP), human's latent states x_H, and data of human's behavior. Human teammate's latent state is specified as her intent (also referred to as subgoal). Specifically, the human is modeled to have five intents corresponding to different landmarks in the shared workspace (i.e., cooking stations, wrapping stations, and pantry). Further, all approaches utilize identical subgoal-dependent human model, learned from interaction data, for anticipating human behavior in absence of robot communications (i.e., when $s_{RC} = a_{RC} = \varnothing$). However, they differ in their approach to communication decision-making.

For both the baselines, no-communication policy and hand-crafted communication policy, the robot action a_R-selection was done by using the POMDP model with empty communication state and action spaces, i.e., $A_{RC} = S_{RC} = \varnothing$. For the no-communication policy, the robot was silent during the task. For the hand-crafted communication policy, the following policy was used,

- `ask`: when the human's intent is ambiguous, i.e., when no human intent had a belief above a hand-crafted threshold;
- `inform`: when the robot is confident about its next subgoal;
- `command`: if the robot anticipated the human and robot subtasks to interfere and the human is at the pantry (i.e., the same domain knowledge used to specify the compliance model for the communication decision-making framework); and
- to prevent repetitive communications, stay silent during the duration of interruption interval (h_c).

The communication space (A_{RC}) and the duration of interruption interval (h_c) was identical for both the hand-crafted and POMDP-based approaches. I reiterate that the planning for robot's actions and communications was done jointly for the POMDP-based approach (i.e., in the treatment level corresponding to the communication decision-making framework). The POMDP model for joint action and communication decision-making included $\approx$ 31 million states and 13 actions.

Procedure

Participants for the study were recruited at MIT campus, and the study protocol was approved by MIT's institutional review board. The experimental procedure was analogous to the procedure for experiments presented in the previous chapter. After obtaining informed consent, participants were briefed about the robot, the safety systems in place, and the collaborative task.

Each participant completed the task six times (two consecutive repetitions for each condition) with the robot. The order of treatments was randomized across participants. In addition, to alleviate novelty effects, the participants performed a training trial at the start of the experiment session. Further, during each task trial the robot indicated the start and end of the experiment by communicating a pre-scripted message. The participants were administered a pre-experiment demographic survey, three identical questionnaires during the experiment (one after each condition), and an open-ended survey at the end of the experiment.

Dependent Measures

I evaluate the effectiveness of robot's communication using both objective and subjective measures. Objective measures include the cumulative shared reward (a composite measure of safety and collaborative efficiency) and task completion times. Subjective measures, summarized in Table 5.3, assess participants' perception of safety and collaborative fluency. Subjective measures and comments were acquired through the in-experiment and post-experiment questionnaires.

Communication Decision-Making	Shared Workspace Task		
	Reward	Task Time (s)	Number of Robot Comm.
Silent	-292.2 $\pm$ 24.4	95.4 $\pm$ 4.0	0 $\pm$ 0
Hand-crafted	-312.2 $\pm$ 19.5	98.7 $\pm$ 3.8	4.4 $\pm$ 0.1
POMDP-based	**-236.4 $\pm$ 11.2**	**88.4 $\pm$ 2.8**	5.4 $\pm$ 0.3

Table 6.1: Objective metrics of safety, collaborative efficiency, and communication decision-making for collaboration in the shared workspace task (experiments with human participants, $N = 15$).

6.11.2 Results

I next report the results of the study for interaction with 15 participants (5 male, 10 female, median age: 26 years). Six participants reported prior experience with robots. The objective measures and number of robot communications, averaged across the fifteen participants, are listed in Table 6.1. Table 6.2 includes the median responses for the subjective measures. I evaluate the effect of the independent variable on the dependent measures using nonparametric statistical tests.

Algorithmic computation of robot's communication policy outperforms both the no communication and hand-crafted communication policies. I begin with comparing the objective criterion stated in the problem definition, i.e., the cumulative shared reward $R(s, a)$. Averaged across the participants, the POMDP-based framework for communication decision-making accrues a substantially higher cumulative reward than both the baselines. The effect of robot's communication decision-making approach on the shared reward is statistically significant ($p = 0.01$) as evaluated by the Friedman test, which rendered a Chi-square value of 9.09. Through pairwise comparisons conducted using Wilcoxon signed-rank test, I confirm that the reward accrued by the POMDP-based framework is statistically significantly higher than that accrued by both the Silent ($p < 0.05$) and Hand-crafted policies ($p < 0.01$). Thus, the experiments provide evidence supporting both the hypotheses *H1* and *H2*.

Communication Decision-Making		Shared Workspace Task		
		Silent	Hand-crafted	POMDP-based
Collaborative Fluency				
Q_1	The robot was a good partner.	6	6	6
Q_2	The robot and I worked well as a team.	6	6	6
Q_3	I am dissatisfied with how the robot and I worked together.	2	2	2
Q_4	The robot perceived accurately what my goals were.	5	6	5
Q_5	I trusted the robot to do the right thing at the right time.	5	6	5
Safety				
Q_6	The robot kept a safe distance from me.	6	6	6
Q_7	The robot got in my way.	2	2	2
Q_8	The robot moved too fast.	2	2	2
Q_9	I felt uncomfortable working so close to the robot.	2	2	2

Table 6.2: Subjective metrics of safety and collaborative fluency for collaboration in the shared workspace task (experiments with human participants, $N = 15$).

The POMDP-based framework exhibits characteristics of effective information sharing. As reflected in the objective measures and the supporting evidence for hypothesis 1 *(H1)*, the experiments confirm the utility of communications for human-robot collaboration. Specifically, by effectively utilizing robot communications, the POMDP-based framework results in higher cumulative reward and lower task completion times as compared to the Silent policy. However, as observed by the inferior performance of the Hand-crafted policy, presence of a communication modality alone is insufficient to improve collaboration.

Instead, purposeful use of the communication modality is essential to realize its benefits. Thus, as motivated in Sec. 2.5, the experiments also provide evidence supporting the importance of effective information sharing for human-robot collaboration. Despite making only one more communication (on average) than the Hand-crafted policy, the POMDP-based framework accrues substantially higher reward. Further, the policy is algorithmically generated via a general framework that is applicable for other collaborative tasks and communication modalities.

No statistically significant differences are found in the subjective measures, possibly due to the difference in the task completion times across trials not being perceptible for the participants. However, small difference in task times are highly relevant for human-robot collaboration in the real world, e.g., for applications in manufacturing and healthcare domains. Despite no statistical differences, the participant's open-ended comments reinforce the need of effective communications. In addition to the communications implemented in the study, the participants indicated preference for "the communication back and forth", highlighting the need for multiple communication types and bidirectional communication.

The interactions with human subjects also provide guidance for applying the POMDP-based framework for communication decision-making to other collaborative scenarios. For instance, during the experiments, at times the robot repeated its communications to emphasize its effect. Depending on the requirements of the task, this behavior may be emphasized or de-emphasized, as need be, by tuning the cost model (e.g., the value of the interruption interval).

Finally, the design of the framework and its promising performance with human participants also lays the foundation for several near-term use cases as well as long-term research directions. For instance, the framework is ripe for applications in other collaborative tasks (which can be described using the MMDP model), novel domains, and with other communication modalities. In order to realize these applications and improve upon the framework, future research investigations into modeling human-robot communication cost as well as joint approaches for symbol grounding and communication decision-making will be important.

6.12 Summary

In this chapter, we discussed a novel decision-making framework that enables robots to decide *if, when, and what* to communicate while performing sequential and collaborative tasks with humans. The framework considers multiple communication types and jointly reasons about robot's actions and communications. Informed by the challenges of developing and deploying collaborative machines, the model specification approach of the framework is both modular and hybrid. As collecting communication-based interaction data is difficult in practice, the specification process enables merging modules learned from data with those obtained from the developer's domain expertise.

Through experiments with human participants, I demonstrate the utility of the communication decision-making framework for a human-robot collaborative task with large state space and short planning time. By realizing effective communications via hybrid model specification and online planning, the framework provides enabling technical advances for the practice of human-robot collaboration. Towards the theory of human-robot collaboration, I introduce *responsivity* and extend the formalization of *compliance* to incorporate temporal, context- and communication- dependent variations.

Chapter 7

Conclusion

In this dissertation, I identified three opportunities of *effective information sharing for human-robot collaboration*. By formalizing these opportunities as learning and decision-making problems, I developed and presented modeling and algorithmic advances to enable fluent human-robot collaboration. In this chapter, I summarize these contributions of the dissertation. Finally, I conclude with a discussion of a few future directions, which arise from my thesis research, for advancing the theory and practice of human-machine interaction.

7.1 Summary of Contributions

In summary, the dissertation details the following contributions that leverage and enable effective information sharing for human-robot collaboration:

- Agent Markov Model (AMM), a generative model of a Markovian agent's sequential decision-making behavior. The AMM describes the behavior of agents, both human and artificial, whose decisions depend on both observable and latent decision factors (e.g., mental states). The factored structure of the model facilitates the specification of known parameters of the agent's behavior. Finally, by modeling the conditional dependence among the agent's observable behavior and latent model parameters, the AMM enables the development of algorithms for joint learning of agent's dynamics and policy.

- Constrained Variational Inference (CVI), a hybrid approach to learning generative models that can incorporate both (semi-)supervised data and constraints on model parameters as learning inputs. Robot developers often possess domain knowledge regarding human-interpretable AMM parameters (such as policy and dynamics). CVI converts such domain knowledge to machine-interpretable variational parameters and enables hybrid learning.

- A suite of Bayesian inference algorithms for learning the AMM that can utilize different types of information – including sequences of observed behavior, prior knowledge of agent's policy, and partial specification of agent's decision factors and their dynamics. Inspired by the availability (or lack thereof) of different input types while generating human teammate models, the algorithms are developed and validated for a variety of cases: namely, (a) parametric and non-parametric, (b) supervised, semi-supervised, and unsupervised, and (c) hybrid.

- Adaptive Collaborative with Reduced Labeling or ADACORL, a hybrid framework to specify the robot's interaction policy with partially labeled data and the developer's domain expertise. ADACORL, despite significantly fewer labels, generates interaction policies that perform equally well or better than models learned with supervised data. By leveraging algorithms for hybrid AMM learning and planning under uncertainty, the framework can generate robot behavior for human-robot collaborative tasks with large state spaces (> 1 million states) and short planning times (< 1 s).

- A framework for joint decision-making of robot's actions and communications for sequential human-robot collaborative tasks. By following the hybrid specification paradigm, the framework enables the developer to specify a model for the robot's decision-making using both data and domain expertise. Using speech as the communication modality, I demonstrate the framework in a task with multiple communication types and short planning times. Through experiments with human participants, I confirm the ability of the framework to enable effective information sharing during human-robot collaboration.

7.2 Future Directions

In summary, the contributions of the dissertation support my thesis that *effective information sharing between the robot developer, the human teammate, and the collaborative robot is essential for human-robot collaboration and is enabled by the modeling and algorithmic advances presented in this dissertation.* The presented modeling and algorithmic advances are, however, only a few pieces of the puzzle for answering the motivating question:

How does one develop collaborative machines?

Specifically, many more opportunities of effective information sharing remain to be seized for answering this question. In this final section of the dissertation, informed by my research on effective information sharing, I highlight a few promising directions for human-machine collaboration research.

7.2.1 Taxonomy of Collaborative Tasks

The scope of the problems considered in this dissertation is limited to a subclass of collaborative tasks, namely, tasks for human-robot dyads with discrete state space, action space, and Markovian transitions. Extensions of this subclass are possible across a variety of dimensions, including

- the number of members of the human-machine team, leading to research and development for human-agent groups [40];
- prior training and coordination between the team members, leading to investigations into novice versus expert teams; and
- knowledge available to the team members before and during the collaboration.

Even for teams with two members, one human and one machine, a taxonomy of different subclasses of collaborative tasks remains to be developed. Such a taxonomy will be useful for formally understanding the applicability of (both existing and prospective) models and algorithms for human-machine collaboration as well as for computationally analyzing their coordination complexity.

7.2.2 Merging Specifications with Experience

For the collaborative tasks considered in this dissertation, I further assume that the developer has complete knowledge of the collaborative task. For instance, the solutions for generating a collaborative robot's interaction and communication policies (cf. Chapters 5-6) require the developer to specify a task model.

A few natural extensions of this problem setting include generating robot policies in collaborative scenarios where the task specification itself is unknown, incorrect, or incomplete. Research on such scenarios will be crucial for enabling both long-term and increasingly general human-machine collaboration. For instance, solutions are required for designing collaborative machines that can

- support humans in the wild, where the developer's knowledge of the human-machine team's environment may be incomplete or inaccurate [153];
- adapt their behavior as the objective of the task (i.e., the reward) or preferences of the human teammate change during the interaction; and
- collaborate with humans in tasks where the developer knows the task but finds it too difficult or cumbersome to specify, resulting in an incomplete specification of the collaborative task.

In a similar vein to ADACORL, a hybrid paradigm that merges the developer specifications with interaction data could address the challenges of unknown, incorrect, or incomplete task specifications. For instance, such a hybrid paradigm may be instantiated by fusing techniques for model-based planning (which utilize developer-specified task models) with reinforcement learning (which enables learning from machine's experience).

7.2.3 Realizing Characteristics of Effective Teams

As motivated in Chapter 2, research on human teams points to a variety of characteristics of effective teams [14, 98, 128]. The problems discussed in this dissertation are inspired by three of these characteristics – namely, modeling the teammate, adapting to the teammate, and communicating with the teammate.

Future investigations should explore modeling and algorithmic contributions to realize other characteristics of effective teaming; for instance,

- engendering trust among teammate and team cohesion, especially during long-term human-machine interactions;
- creating algorithmic strategies for conflict management, especially for teamwork in resource-constrained domains; and
- creating strategies to explain, discuss, and understand coordination failures.

7.2.4 Human-in-the-Loop Machine Learning

The contributions of the thesis — specifically constrained variational inference and its application to learning the teammate model — provide the building blocks for human-in-the-loop machine learning with partial and high-level specifications. Building on this contribution, novel approaches to human-in-the-loop learning are possible that hold the potential to reduce sample complexity and to improve learning performance. Specifically, promising future investigations are possible along the following three axes; namely,

- the model/parameter being learned;
- the inputs available from the human expert; and
- the expertise of the human in the loop.

Central to these investigations will be (a) theoretical developments in the area of active learning; (b) novel hybrid algorithms that accept as inputs not only additional labels but also high-level specifications; and (c) interactive systems that enable humans and machines to communicate during the active and hybrid learning process. In addition to enabling human-machine collaboration in new settings, these directions will have broad utility for learning generative models, transferring agent experience from simulations to the real world, and for enabling the use of learning algorithms by non-programmers.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] ABB. ABB linear axis units. `http://new.abb.com/products/robotics/industrial-robots/irb-6620lx`, 2017. (online; last accessed October 3, 2019).

[2] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*. ACM, 2004.

[3] Henny Admoni and Brian Scassellati. Robot nonverbal communication as an AI problem (and solution). In *AAAI Fall Symposium Series*, 2015.

[4] Henny Admoni and Brian Scassellati. Social eye gaze in human-robot interaction: A review. *Journal of Human-Robot Interaction (JHRI)*, 6(1):25–63, 2017.

[5] Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[6] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[7] Christopher Amato, George Konidaris, Leslie P. Kaelbling, and Jonathan P. How. Modeling and planning with macro-actions in decentralized POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 64:817–859, 2019.

[8] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[9] Jacob Arkin, Rohan Paul, Daehyung Park, Subhro Roy, Nicholas Roy, and Thomas M. Howard. Real-time human-robot communication for manipulation tasks in partially observed environments. In *International Symposium on Experimental Robotics (ISER)*, 2018.

[10] Kent Bach and Robert M. Harnish. Linguistic communication and speech acts. 1979.

[11] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. Learning robot objectives from physical human interaction. *Proceedings of Machine Learning Research (PMLR)*, 78:217–226, 2017.

[12] Kim Baraka and Manuela M. Veloso. Mobile service robot state revealing through expressive lights: Formalism, design, and evaluation. *International Journal of Social Robotics (IJSR)*, 10(1):65–92, 2018.

[13] Michael D. Basil. Multiple resource theory. *Encyclopedia of the Sciences of Learning*, pages 2384–2385, 2012.

[14] Jerome Bourbousson, Germain Poizat, Jacques Saury, and Carole Seve. Team coordination in basketball: Description of the cognitive connections among teammates. *Journal of Applied Sport Psychology*, 22(2):150–166, 2010.

[15] Craig Boutilier. Planning, learning and coordination in multi-agent decision processes. In *Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210. Morgan Kaufmann Publishers Inc., 1996.

[16] Frank Broz, Illah Nourbakhsh, and Reid Simmons. Planning for human-robot interaction in socially situated tasks. *International Journal of Social Robotics (IJSR)*, 5(2):193–214, 2013.

[17] Abhizna Butchibabu, Christopher Sparano-Huiban, Liz Sonenberg, and Julie Shah. Implicit coordination strategies for effective team communication. *Human Factors*, 58(4):595–610, 2016.

[18] Maya Cakmak and Andrea L. Thomaz. Designing robot learners that ask good questions. In *International Conference on Human-Robot Interaction (HRI)*, pages 17–24. ACM, 2012.

[19] Crystal Chao and Andrea Thomaz. Timed Petri nets for fluent turn-taking over multimodal interaction resources in human-robot collaboration. *International Journal of Robotics Research (IJRR)*, 35(11):1330–1353, 2016.

[20] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Planning with trust for human-robot collaboration. In *International Conference on Human-Robot Interaction (HRI)*, pages 307–315. ACM, 2018.

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

[22] Rohan Choudhury, Gokul Swamy, Dylan Hadfield-Menell, and Anca D. Dragan. On the utility of model learning in HRI. In *International Conference on Human-Robot Interaction (HRI)*, pages 317–325. ACM, 2019.

[23] Sharolyn Converse, Janis A. Cannon-Bowers, and Eduardo Salas. Shared mental models in expert team decision making. *Individual and Group Decision Making: Current Issues*, 221, 1993.

[24] J. Michael Crant. Proactive behavior in organizations. *Journal of Management*, 26(3):435–462, 2000.

[25] Kerstin Dautenhahn. Methodology & themes of human-robot interaction: A growing research field. *International Journal of Advanced Robotic Systems*, 4(1): 15, 2007.

[26] Arwen H. DeCostanza, Amar R. Marathe, Addison Bohannon, A. William Evans, Edward T. Palazzolo, Jason S. Metcalfe, and Kaleb McDowell. Enhancing human-agent teaming with individualized, adaptive technologies: A discussion of critical scientific questions. Technical report, US Army Research Laboratory, Aberdeen Proving Ground, United States, 2018.

[27] Barbara Di Eugenio, Pamela W. Jordan, Richmond H. Thomason, and Johanna D. Moore. The agreement process: An empirical investigation of human-human computer-mediated collaborative dialogs. *International Journal of Human-Computer Studies*, 53(6):1017–1076, 2000.

[28] Myron A. Diftler, J.S. Mehling, Muhammad E. Abdallah, Nicolaus A. Radford, Lyndon B. Bridgwater, Adam M. Sanders, Roger Scott Askew, D. Marty Linn, John D. Yamokoski, F.A. Permenter, et al. Robonaut 2-the first humanoid robot in space. In *International Conference on Robotics and Automation (ICRA)*, pages 2178–2183. IEEE, 2011.

[29] Finale Doshi-Velez, David Pfau, Frank Wood, and Nicholas Roy. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(2): 394–407, 2015.

[30] Anca D. Dragan. Robot planning with mathematical models of human state and action. *arXiv preprint arXiv:1705.04226*, 2017.

[31] Anca D. Dragan and Siddhartha S. Srinivasa. A policy-blending formalism for shared control. *International Journal of Robotics Research (IJRR)*, 32(7):790–805, 2013.

[32] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *International Conference on Human-Robot Interaction (HRI)*, pages 301–308. ACM, 2013.

[33] Mary T. Dzindolet, Scott A. Peterson, Regina A. Pomranky, Linda G. Pierce, and Hall P. Beck. The role of trust in automation reliance. *International Journal of Human-Computer Studies*, 58(6):697–718, 2003.

[34] Elliot E. Entin and Daniel Serfaty. Adaptive team coordination. *Human Factors*, 41(2):312–325, 1999.

[35] A. William Evans, Michelle E. Harper, and Florian Jentsch. I know what you're thinking: Eliciting mental models about familiar teammates. In *International Conference on Concept Mapping*. Citeseer, 2004.

[36] Jaime F. Fisac, Andrea Bajcsy, Sylvia L. Herbert, David Fridovich-Keil, Steven Wang, Claire J. Tomlin, and Anca D. Dragan. Probabilistically safe robot planning with confidence-based human predictions. In *Robotics: Science and Systems (RSS)*, 2018.

[37] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2137–2145, 2016.

[38] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, pages 1020–1056, 2011.

[39] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[40] Marlena Fraune*, Vaibhav V. Unhelkar*, et al. Reports of the AAAI 2017 fall symposium series: Symposium on human-agent groups. *AI Magazine*, 39(2), 2018. Asterisks denote equal contribution.

[41] Lex Fridman, Bryan Reimer, Bruce Mehler, and William T. Freeman. Cognitive load estimation in the wild. In *CHI Conference on Human Factors in Computing Systems*, page 652. ACM, 2018.

[42] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 16(1): 1437–1480, 2015.

[43] C. Lee Giles and Kam-Chuen Jim. Learning communication for multi-agent systems. In *Workshop on Radical Agent Concepts*, pages 377–390. Springer, 2002.

[44] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 137–144. IFAAMAS, 2003.

[45] Matthew Gombolay, Anna Bair, Cindy Huang, and Julie Shah. Computational design of mixed-initiative human-robot teaming that considers human factors: Situational awareness, workload, and workflow preferences. *International Journal of Robotics Research (IJRR)*, 36(5-7):597–617, 2017.

[46] Matthew Gombolay, Xi Jessie Yang, Bradley Hayes, Nicole Seo, Zixi Liu, Samir Wadhwania, Tania Yu, Neel Shah, Toni Golen, and Julie Shah. Robotic assistance in the coordination of patient care. *International Journal of Robotics Research (IJRR)*, 37(10):1300–1316, 2018.

[47] Michael A. Goodrich, Alan C. Schultz, et al. Human-robot interaction: A survey. *Foundations and Trends® in Human–Computer Interaction*, 1(3):203–275, 2008.

[48] Barbara J. Grosz, Luke Hunsberger, and Sarit Kraus. Planning and acting together. *AI magazine*, 20(4):23–23, 1999.

[49] Dylan Hadfield-Menell, Stuart J. Russell, Pieter Abbeel, and Anca D. Dragan. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3909–3917, 2016.

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

[51] Laura M. Hiatt, Cody Narber, Esube Bekele, Sangeet S. Khemlani, and J. Gregory Trafton. Human modeling for human-robot collaboration. *International Journal of Robotics Research (IJRR)*, 36(5-7):580–596, 2017.

[52] Guy Hoffman. Evaluating fluency in human–robot collaboration. *IEEE Transactions on Human-Machine Systems (THMS)*, 49(3):209–218, 2019.

[53] Guy Hoffman and Cynthia Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *International Conference on Human-Robot Interaction (HRI)*, pages 1–8. ACM, 2007.

[54] Guy Hoffman et al. *Ensemble: fluency and embodiment for robots acting with humans*. PhD thesis, Massachusetts Institute of Technology, 2007.

[55] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14(1):1303–1347, 2013.

[56] Eric Horvitz and Johnson Apacible. Learning and reasoning about interruption. In *International Conference on Multimodal Interfaces (ICMI)*, pages 20–27. ACM, 2003.

[57] Mithun George Jacob, Yu-Ting Li, George A. Akingba, and Juan P. Wachs. Collaboration with a robotic scrub nurse. *Communications of the ACM*, 56(5): 68–75, 2013.

[58] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S. Srinivasa, and J. Andrew Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. *International Journal of Robotics Research (IJRR)*, 37(7):717–742, 2018.

[59] Matthew Johnson, Catholijn Jonker, Birna Van Riemsdijk, Paul J. Feltovich, and Jeffrey M. Bradshaw. Joint activity testbed: Blocks world for teams (bw4t). In *International Workshop on Engineering Societies in the Agents World*, pages 254–256. Springer, 2009.

[60] Matthew J. Johnson and Alan S. Willsky. Bayesian nonparametric hidden semi-Markov models. *Journal of Machine Learning Research (JMLR)*, 14(Feb): 673–701, 2013.

[61] Matthew J. Johnson and Alan S. Willsky. Stochastic variational inference for Bayesian time series models. In *International Conference on Machine Learning (ICML)*, pages 1854–1862, 2014.

[62] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`, 2001. (online; last accessed October 3, 2019).

[63] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Partially observable Markov decision processes for artificial intelligence. In *AAAI Conference on Artificial Intelligence*. Springer, 1995.

[64] Daniel Kahneman. Maps of bounded rationality. *The American Economic Review*, 93(5):1449–1475, 2003.

[65] Ece Kamar, Ya'akov Gal, and Barbara J. Grosz. Incorporating helpful behavior into collaborative planning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 875–882, 2009.

[66] Tatsuya Kasai, Hiroshi Tenmoto, and Akimoto Kamiya. Learning of communication codes in multi-agent reinforcement learning problem. In *Conference on Soft Computing in Industrial Applications*, pages 1–6. IEEE, 2008.

[67] Tobias Kaupp, Alexei Makarenko, and Hugh Durrant-Whyte. Human-robot communication for collaborative decision making — a probabilistic approach. *Robotics and Autonomous Systems*, 58(5):444 – 456, 2010.

[68] Ross A. Knepper, Christoforos I. Mavrogiannis, Julia Proft, and Claire Liang. Implicit communication in a joint action. In *International Conference on Human-Robot Interaction (HRI)*, pages 283–292. ACM, 2017.

[69] Hema Swetha Koppula and Ashutosh Saxena. Anticipating human activities for reactive robotic response. In *International Conference on Intelligent Robots and Systems (IROS)*, page 2071. IEEE, 2013.

[70] Dieter Kraft. Algorithm 733: TOMP–Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

[71] Sanjay Krishnan, Animesh Garg, Ken Goldberg, et al. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[72] Thibault Kruse, Patrizia Basili, Stefan Glasauer, and Alexandra Kirsch. Legible robot navigation in the proximity of moving humans. In *Advanced Robotics and its Social Impacts (ARSO)*, pages 83–88. IEEE, 2012.

[73] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[74] KUKA. KUKA linear units. `https://www.kuka.com/en-us/products/robotics-systems/robot-periphery/linear-units`, 2017. (online; last accessed October 3, 2019).

[75] P. Kumar and Pravin Varaiya. *Stochastic Systems*. SIAM, 2016.

[76] Paul Lamere, Philip Kwok, Evandro GouvÃła, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The CMU SPHINX-4 speech recognition system, 2003.

[77] Przemyslaw A. Lasota and Julie A. Shah. A multiple-predictor approach to human motion prediction. In *International Conference on Robotics and Automation (ICRA)*, pages 2300–2307. IEEE, 2017.

[78] Przemyslaw A. Lasota, Terrence Fong, Julie A. Shah, et al. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5 (4):261–349, 2017.

[79] Hector J. Levesque, Philip R. Cohen, and José H.T. Nunes. On acting together. In *AAAI Conference on Artificial Intelligence*, volume 90, pages 94–99, 1990.

[80] Jing Li, Zhong Shen, Wen Yu Tian Xu, Walter Yu Hang Lam, Richard Tai Chiu Hsung, Edmond Ho Nang Pow, Kazuhiro Kosuge, and Zheng Wang. A compact dental robotic system using soft bracing technique. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1271–1278, 2019.

[81] Shen Li and Julie A. Shah. Safe and efficient high dimensional motion planning in space-time with time parameterized prediction. In *International Conference on Robotics and Automation (ICRA)*, pages 5012–5018. IEEE, 2019.

[82] Sirui Li, Weixing Sun, and Tim Miller. Communication in human-agent teams for tasks with joint action. In *Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN)*, pages 224–241. Springer, 2015.

[83] Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3812–3822, 2017.

[84] Claire Liang, Julia Proft, Erik Andersen, and Ross A. Knepper. Implicit communication of actionable information in human-AI teams. In *CHI Conference on Human Factors in Computing Systems*, page 95. ACM, 2019.

[85] Miao Liu, Xuejun Liao, and Lawrence Carin. The infinite regionalized policy representation. In *International Conference on Machine Learning (ICML)*, pages 769–776. ACM, 2011.

[86] Ruikun Luo, Rafi Hayne, and Dmitry Berenson. Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces. *Autonomous Robots*, 42(3):631–648, 2018.

[87] Jean MacMillan, Elliot E. Entin, and Daniel Serfaty. Communication overhead: The hidden cost of team cognition. *Team Cognition: Process and Performance at the Inter and Intra-Individual Level*, 2004.

[88] Aleck MacNally. Human-robot communication in automated planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, page 145, 2016.

[89] Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems (RSS)*, 2017.

[90] Amar R. Marathe, Kristin E. Schaefer, Arthur W. Evans, and Jason S. Metcalfe. Bidirectional communication for effective human-agent teaming. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 338–350. Springer, 2018.

[91] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

[92] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *International Symposium on Experimental Robotics (ISER)*, pages 403–415, 2013.

[93] Nikolaos Mavridis. A review of verbal and non-verbal human–robot interactive communication. *Robotics and Autonomous Systems*, 63:22–35, 2015.

[94] Nikolaos Mavridis and Haiwei Dong. To ask or to sense? Planning to integrate speech and sensori-motor acts. In *International Congress on Ultra Modern Telecommunications and Control Systems*, pages 227–233. IEEE, 2012.

[95] David Meger, Juan Camilo Gamboa Higuera, Anqi Xu, Philippe Giguere, and Gregory Dudek. Learning legged swimming gaits from experience. In *International Conference on Robotics and Automation (ICRA)*, pages 2332–2338. IEEE, 2015.

[96] Francisco S. Melo, Matthijs T.J. Spaan, and Stefan J. Witwicki. QueryPOMDP: POMDP-based communication in multiagent systems. In *European Conference on Multi-Agent Systems (EUMAS)*, pages 189–204. Springer, 2011.

[97] Bernard Michini and Jonathan How. Bayesian nonparametric inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2012.

[98] Sharon Mickan and Sylvia Rodger. Characteristics of effective teams: a literature review. *Australian Health Review*, 23(3):201–208, 2000.

[99] 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.

[100] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The Stanford entry in the urban challenge. In *The DARPA Urban Challenge*, pages 91–123. Springer, 2009.

[101] Stefanos Nikolaidis and Julie Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *International Conference on Human-Robot Interaction (HRI)*, pages 33–40. ACM, 2013.

[102] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *International Conference on Human-Robot Interaction (HRI)*, pages 189–196. ACM, 2015.

[103] Stefanos Nikolaidis, Jodi Forlizzi, David Hsu, Julie Shah, and Siddhartha Srinivasa. Mathematical models of adaptation in human-robot collaboration. *arXiv preprint arXiv:1707.02586*, 2017.

[104] Stefanos Nikolaidis, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in collaborative tasks: Models and experiments. *International Journal of Robotics Research (IJRR)*, 36(5-7):618–634, 2017.

[105] Stefanos Nikolaidis, Minae Kwon, Jodi Forlizzi, and Siddhartha Srinivasa. Planning with verbal communication for human-robot collaboration. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(3):22, 2018.

[106] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[107] Edwin Olson. Behavioral planning and how to escape the hype cycle of autonomous vehicles. CSAIL Alliances Lecture, 2019.

[108] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[109] Alessandro Panella and Piotr Gmytrasiewicz. Interactive POMDPs with Finite-State Models of Other Agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 31(4):861–904, 2017.

[110] Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2):230–253, 1997.

[111] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. Intention-aware motion planning using learning based human motion prediction. In *Robotics: Science and Systems (RSS)*, 2017.

[112] Mike Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In *International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2011.

[113] Steven Pinker, Martin A Nowak, and James J Lee. The logic of indirect speech. *Proceedings of the National Academy of Sciences*, 105(3):833–838, 2008.

[114] Robert W Proctor and Trisha Van Zandt. *Human Factors in Simple and Complex Systems*. CRC Press, 2018.

[115] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley & Sons, 2014.

[116] David V. Pynadath and Milind Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 873–880, 2002.

[117] Aditi Ramachandran, Chien-Ming Huang, and Brian Scassellati. Give me a break!: Personalized timing strategies to promote learning in robot-child tutoring. In *International Conference on Human-Robot Interaction (HRI)*, pages 146–155. ACM, 2017.

[118] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2586–2591, 2007.

[119] Ramya Ramakrishnan, Ece Kamar, Debadeepta Dey, Julie Shah, and Eric Horvitz. Discovering blind spots in reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1017–1025. IFAAMAS, 2018.

[120] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric Bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 471–477. IEEE, 2015.

[121] Harvard Business Review. Smarter smaller safer robots. `http://hbr.org/2015/11/smarter-smaller-safer-robots`, 2015. (online; last accessed October 3, 2019).

[122] Laurel D. Riek. Wizard of Oz studies in HRI: A systematic review and new reporting guidelines. *Journal of Human-Robot Interaction (JHRI)*, 1(1):119–136, 2012.

[123] Maayan Roth, Reid Simmons, and Manuela Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 786–793. ACM, 2005.

[124] Andrei A. Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning (CoRL)*, pages 262–270, 2017.

[125] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverages effects on human actions. In *Robotics: Science and Systems (RSS)*, 2016.

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

[127] Ardavan Saeedi, Matthew Hoffman, Matthew Johnson, and Ryan Adams. The segmented iHMM: a simple, efficient hierarchical infinite HMM. In *International Conference on Machine Learning (ICML)*, pages 2682–2691, 2016.

[128] Eduardo Salas, Nancy J. Cooke, and Michael A. Rosen. On teams, teamwork, and team performance: Discoveries and developments. *Human Factors*, 50 (3):540–547, 2008.

[129] Eduardo Salas, Deborah DiazGranados, Cameron Klein, C. Shawn Burke, Kevin C. Stagl, Gerald F. Goodwin, and Stanley M. Halpin. Does team training improve team performance? a meta-analysis. *Human Factors*, 50(6):903–933, 2008.

[130] Ruhi Sarikaya. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81, 2017.

[131] Kristin E. Schaefer, Edward R. Straub, Jessie Y.C. Chen, Joe Putney, and Arthur W. Evans III. Communicating intent to develop shared situation awareness and engender trust in human-agent teams. *Cognitive Systems Research*, 46:26–39, 2017.

[132] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[133] Oliver C. Schrempf, Uwe D. Hanebeck, Andreas J. Schmid, and Heinz Worn. A novel approach to proactive human-robot cooperation. In *International Workshop on Robot and Human Interactive Communication (Ro-Man)*, pages 555–560. IEEE, 2005.

[134] Daniel Serfaty, Elliot E. Entin, and Joan H. Johnston. Team coordination training. *Making Decisions under Stress: Implications for Individual and Team Training*, 1998.

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

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

[137] Julie Shah and Cynthia Breazeal. An empirical analysis of team coordination behaviors and action planning with application to human–robot teaming. *Human Factors*, 52(2):234–245, 2010.

[138] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using Chaski, a human-inspired plan execution system. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 29–36. ACM, 2011.

[139] Julie A. Shah. *Fluid coordination of human-robot teams*. PhD thesis, Massachusetts Institute of Technology, 2011.

[140] Thomas B. Sheridan. Human–robot interaction: status and challenges. *Human Factors*, 58(4):525–532, 2016.

[141] Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics (T-RO)*, 23(5):874–883, 2007.

[142] Matthijs T.J. Spaan, Geoffrey J. Gordon, and Nikos Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 249–256, 2006.

[143] Aaron St. Clair and Maja Mataric. How robot verbal feedback can improve team performance in human-robot task collaborations. In *International Conference on Human-Robot Interaction (HRI)*, pages 213–220. ACM, 2015.

[144] Aaron Steinfeld, Odest Chadwicke Jenkins, and Brian Scassellati. The Oz of Wizard: Simulating the human for interaction research. In *International Conference on Human-Robot Interaction (HRI)*, pages 101–108. ACM, 2009.

[145] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[146] Daniel Szafir, Bilge Mutlu, and Terrence Fong. Communication of intent in assistive free flyers. In *International Conference on Human-Robot Interaction (HRI)*, pages 358–365. ACM, 2014.

[147] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101 (476):1566–1581, 2006.

[148] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32 (4):64–76, 2011.

[149] Stefanie Tellex, Pratiksha Thaker, Robin Deits, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Toward information theoretic human-robot dialog. In *Robotics: Science and Systems (RSS)*, 2012.

[150] Andrea Thomaz, Guy Hoffman, Maya Cakmak, et al. Computational human-robot interaction. *Foundations and Trends® in Robotics*, 4(2-3):105–223, 2016.

[151] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[152] Pete Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *International Journal of Robotics Research (IJRR)*, 34(3):335–356, 2015.

[153] Vaibhav V. Unhelkar and Julie A. Shah. ConTaCT: Deciding to communicate in time-critical, collaborative tasks in unknown, deterministic domains. In *AAAI Conference on Artificial Intelligence*, 2016.

[154] Vaibhav V. Unhelkar and Julie A. Shah. Learning models of sequential decision-making without complete state specification using Bayesian non-parametric inference and active querying. Technical report, Massachusetts Institute of Technology, 2018.

[155] Vaibhav V. Unhelkar and Julie A. Shah. Learning models of sequential decision-making with partial specification of agent behavior. In *AAAI Conference on Artificial Intelligence*, 2019.

[156] Vaibhav V. Unhelkar, Ho Chit Siu, and Julie A. Shah. Comparative performance of human and mobile robotic assistants in collaborative fetch-and-deliver tasks. In *International Conference on Human-Robot Interaction (HRI)*, pages 82–89. IEEE, 2014.

[157] Vaibhav V. Unhelkar, X. Jessie Yang, and Julie A. Shah. Challenges for communication decision-making in sequential human-robot collaborative tasks. In *Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction at RSS*, 2017.

[158] Vaibhav V. Unhelkar, Stefan Dörr, Alexander Bubeck, Przemyslaw A. Lasota, Jorge Perez, Ho Chit Siu, James C. Boerkoel Jr., Quirin Tyroller, Johannes Bix, Stefan Bartscher, and Julie A. Shah. Mobile robots for moving-floor assembly lines: Design, evaluation, and deployment. *IEEE Robotics & Automation Magazine*, 25(2):72–81, 2018.

[159] Vaibhav V. Unhelkar*, Przemyslaw A. Lasota*, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A. Shah. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters (RA-L)*, 3 (3):2394–2401, 2018. Asterisks denote equal contribution.

[160] Vaibhav V. Unhelkar*, Shen Li*, and Julie A. Shah. Semi-supervised learning of decision-making models for human-robot collaboration. In *Conference on Robot Learning (CoRL)*, 2019. Asterisks denote equal contribution.

[161] Universal Robots. UR10: Industrial robotic arm. https://www.universal-robots.com/products/ur10-robot/, 2017. (online; last accessed October 3, 2019).

[162] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[163] Sonia Waharte and Niki Trigoni. Supporting search and rescue operations with UAVs. In *International Conference on Emerging Security Technologies*, pages 142–147. IEEE, 2010.

[164] Ning Wang, David V. Pynadath, and Susan G. Hill. Trust calibration within a human-robot team: Comparing automatically generated explanations. In *International Conference on Human-Robot Interaction (HRI)*, pages 109–116. ACM, 2016.

[165] Changyun Wei, Koen V. Hindriks, and Catholijn M. Jonker. The role of communication in coordination protocols for cooperative robot teams. In *International Conference on Agents and Artificial Intelligence*, pages 28–39, 2014.

[166] David Whitney, Eric Rosen, James MacGlashan, Lawson L.S. Wong, and Stefanie Tellex. Reducing errors in object-fetching interactions through social feedback. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[167] Simon A. Williamson, Enrico H. Gerding, and Nicholas R. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 641–648, 2009.

[168] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175 (2):487–511, 2011.

[169] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Modeling cooperative multiagent problem solving as decentralized decision processes. *Submitted for Publication to Autonomous Agents and Multi-Agent Systems*, 2004.

[170] X. Jessie Yang, Vaibhav V. Unhelkar, Kevin Li, and Julie A. Shah. Evaluating effects of user experience and system transparency on trust in automation. In *International Conference on Human-Robot Interaction (HRI)*, pages 408–416. ACM, 2017.

[171] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research (JAIR)*, 58:231–266, 2017.

[172] Andrea Maria Zanchettin, Nicola Maria Ceriani, Paolo Rocco, Hao Ding, and Björn Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 13(2):882–893, 2015.

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