AN INTROSPECTIVE APPROACH FOR COMPETENCE-AWARE AUTONOMY

A Dissertation Presented

by

CONNOR BASICH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

April 2023

Robert and Donna Manning College of
Information and Computer Sciences

AN INTROSPECTIVE APPROACH FOR COMPETENCE-AWARE AUTONOMY

A Dissertation Presented

by

CONNOR BASICH

Approved as to style and content by:

Shlomo Zilberstein, Chair

Joydeep Biswas, Member

David D. Jensen, Member

Shannon C. Roberts, Member

Ramesh Sitaramam, Associate Dean for
Educational Programs and Teaching
Robert and Donna Manning College of
Information and Computer Sciences

ACKNOWLEDGEMENTS

The road to completing a PhD is a long one, and one that I would never have been able to traverse without the help and support of numerous individuals, to whom I would like to express my sincere gratitude.

There is no one more important in this journey than my advisor, Shlomo Zilberstein. He has been a tireless force throughout my years in his lab, supporting me in every endeavor I undertook, and always working hard to ensure that I had both the funding and freedom needed to pursue my research passions. He taught me how to perform research of the highest quality, from inception to publication, and instilled in me the many insights he had gained throughout his career in the pursuit of contributing meaningfully to the academic community and beyond. There is no better mentor I could have asked for, and for all of that and more I am extremely grateful.

I would also like to thank the members of my committee who have supported and pushed me in my effort to complete this dissertation, and who have provided insightful and encouraging feedback each step of the way. Joydeep Biswas was a constant source of insight, guidance, and passion for research from my first paper which was, appropriately, the very seed that would bloom into this dissertation. David Jensen was one of the most passionate and motivating professors that I had the pleasure of learning from in my time at UMass, and he shaped the way I approach research to this day. Shannon Roberts, with whom I was extremely fortunate to collaborate on my very first project as a PhD student, always provided meaningful insights on research, and encouraged me to approach problems from new perspectives.

I would like to specially thank Stefan Witwicki who was not only a long-time collaborator, but a thoughtful, patient, and caring mentor, who worked hard to provide the resources necessary to make our collaborations highly successful, and taught me some of the most important skills I have learned at UMass for applying research ideas to real-world problems in a meaningful and substantive capacity.

I will never forget my time at the Resource-Bounded Reasoning (RBR) lab. I was extremely fortunate to work with the most patient, passionate, and helpful senior lab members in Kyle Wray, Luis Pineda, Sandhya Saisubramanian, and Rick Freedman. They were my day-to-day mentors for *many* days, and I have only developed into the researcher I am thanks to them. Justin Svegliato, Samer Nashed, John Peterson, Shuwa Miura, Abhinav Bhatia, Saad Mahmud, and Moumita Choudhary were constant, active collaborators, to whom I owe a great deal of my success, as research is almost never an individual pursuit, nor the result a single person's efforts.

I would also like to thank my undergraduate advisor, John Shareshian, who first taught me my love for math and theory, and without whom I would have never even considered pursuing a PhD, as well as Michele Roberts, the backbone of the RBR lab whose support and care made my PhD vastly more navigable than it otherwise would have been.

I have many friends who helped me to create many fond memories and make the entire PhD experience manageable even when it was at its most difficult: Claudia, Suyash, David, Jay, Ben, Samer, Conrad, Yume, Sam Witty, Sam Baxter, Su Lin, Tiffany, Katherine, Miguel, and Sophie. From courses, to dungeons and dragons, to trivia nights with bad "toothpaste mojitos", to bike rides (and crashes!), to boardgames and cocktails, to Montreal, and the many other joyful times and experiences, I will always be thankful for your friendship.

My parents, Anthony and Mary Basich, have been a constant source of support and love. I owe many things to them, not the least of which is the opportunity to have completely a PhD, but my passion for knowledge, and the drive to always be the best version of myself. My grandmother, Blazenka Basich, who made sure to call me frequently and express steadfast love and encouragement. My siblings, Chase and Arielle Basich, who helped in so many ways over the years, and always reminded me of my connection to home with our daily chats. I love you all very much.

And finally, to Maize, my best friend and the best dog in the world, who kept me healthy, happy, and sane for many years, and to whom I hope that I can do the same. I dedicate this to you, although you may never understand what it says.

ABSTRACT

AN INTROSPECTIVE APPROACH FOR COMPETENCE-AWARE AUTONOMY

APRIL 2023

CONNOR BASICH

B.A., WASHINGTON UNIVERSITY IN ST. LOUIS

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Shlomo Zilberstein

Building and deploying autonomous systems in the open world has long been a goal of both the artificial intelligence (AI) and robotics communities. From autonomous driving, to health care, to office assistance, these systems have the potential to transform society and alter our everyday lives. The open world, however, presents numerous challenges that question the typical assumptions made by the models and frameworks often used in contemporary AI and robotics. Systems in the open world are faced with an unconstrained and non-stationary environment with a range of heterogeneous actors that is too complex to be modeled in its entirety. Moreover, many of these systems are expected to operate on the order of months or even years. To more reliably handle these challenges, many autonomous systems deployed in human environments entail some measure of reliance on human assistance. This reliance on human assistance is an acknowledgement of a limited *competence* of the autonomous agent to complete its tasks fully autonomously in all situations. Consequently, in order for such systems to be effective in the open world, they, like humans, must be aware of their own competence and both capable and incentivized to solicit external assistance when needed. This thesis therefore proposes planning approaches based on the concept of *competence modeling* that equip an autonomous system with knowledge about both its capabilities and limitations to better optimize its autonomy and operate more effectively in the open world.

In the first part of this thesis, we introduce the notion of *competence modeling* and introduce a planning framework called a *competence-aware system*. A competence-aware system (CAS) enables a semi-autonomous system to reason about its own competence in the form of multiple levels of autonomy, and integrate that information into its decision-making model. We formulate the CAS model in the context of a fully-observable MDP, but show how it can be extended to the partially-observable setting in a well-defined manner. The result is a system that is not only more robust to unforeseen situations, but capable of optimizing its own autonomy online through interactions with a human operator who can provide varying levels of assistance.

Each subsequent chapter in the dissertation explores relaxing one or more of the assumptions made in the base formulation of the CAS model, and proposes a technique or model to address the resultant challenge(s) to improve the applicability of the CAS model to real-world problems. First, we propose a method for a competence-aware system to improve its competence over time by exploiting apparent inconsistencies in the existing human feedback to iteratively refine its state representation. This method, which we call *iterative state space refinement*, leads to a more nuanced drawing of the boundaries between regions of the agent's state-action space with different degrees of competence. The result is a system that can better exploit human assistance, improving its overall competence. Second, we propose an extension to the base CAS model called a *contextual competence-aware system* (CoCAS), which extends the CAS model to the setting with multiple, heterogeneous human operators with stochastic states and contextual competence dependence. We show that the same theoretical guarantees exhibited by a CAS extend to the CoCAS, with strictly greater representational power. Finally, we propose to extend the learning model, which is dependent on the assumption that feedback from the human is provided *re-actively* for the current state and action of the system, to consider *proactive* feedback that is generated by the human conditioned on their inferred behavior of the system in the near future. We conclude with a summary and discussion of the contributions presented in the preceding chapters of the thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Autonomous robotic systems are increasingly being built and deployed in highly complex and unstructured domains in the "open world" where they are expected to operate reliably for extended durations while facing challenging problems such as environmental variability, partial observability, multi-agent interactions, and unexpected scenarios [63]. These domains range from space exploration [42, 79] to autonomous underwater vehicles [25, 62, 109] to service robots [50, 75], and to self-driving cars [20, 21, 33]. Often in such domains, fully enumerating every scenario that the system can encounter during deployment is infeasible, and the systems must consequently rely on approximate models of their domains that do not capture the full space of possible situations in its entirety.

Nevertheless, these systems are expected to maintain safe and reliable operation throughout their deployment. To enable this, many autonomous systems deployed in the open world already include some form of reliance on human assistance. These systems are therefore better described as *semi-autonomous systems* that can operate autonomously under certain conditions, but may require human intervention or assistance in other situations in order to achieve their assigned goals [128]. For example, an autonomous car may request that the driver take over in the presence of unexpected obstacles, or if lane demarcation is lost; a space exploration rover may suspend operation and wait for a new plan from ground control in the event that system parameters fall outside some predetermined scope. In each case, the reliance on human assistance suggests a limitation of the agent's ability to compete its task fully autonomously all of the time, in the same capacity as how humans may also seek assistance from other humans or external resources when unable to complete a task by themselves. This thesis, therefore, aims to answer the question: how to best perform planning for autonomous systems that are aware of both their strengths *and* limitations, and which have the ability to seek various forms of external assistance to better complete their tasks.

To answer this question, this thesis proposes the notion of *competence-aware autonomy*, a paradigm for enabling semi-autonomous systems to learn and reason about (1) their limitations in executing a task autonomously, (2) the environmental or situational factors that influence these limitations, and (3) the proper form and extent of human assistance to request to optimally compensate for their limitations. We argue that a semi-autonomous system deployed into the real world should be able to reason, at any point, about whether it has the requisite *competence* to act autonomously, and, if not, reason about the appropriate level of human assistance needed to compensate for its limited competence. In particular, the robot should aim to not be over-reliant on human assistance, placing unnecessary burden on the human that may lead to a higher cost, an over-burdened human, and potentially lower trust in the system and less willingness to use the system. At the same time, the robot should also aim to not be under-reliant on human assistance, taking excessive time and energy to perform what may be a simple, low-cost operation for the human, or worse, attempting what may be an unsafe operation for the robot. We refer to this notion as *competence modeling*.

Competence is generally understood to capture some notion of an individual's skill, aptitude, performance, or capabilities [4], and there have been many attempts over the last several decades to define it in the context of humans. In 1984, the Further Education Unit (FEU) defined competence as "the possession and development of sufficient skills, knowledge, appropriate attitude, and experience for successful performance in life roles" [115]. While intuitive, this definition fails to be precise in the mathematical sense. What does "sufficient" entail? How do we define a "skill" or a "life role"? Many other definitions of competence have been proposed since [35, 101, 111], but they remain largely atomistic and unwieldy, lacking a well-defined mathematical representation in their generality.

Gilbert [45] provides a more measurable definition of competence as a function of the ratio of valuable accomplishments to costly behavior, although the function is left unspecified. This definition seems to fit well within the AI and robotics sentiment of efficiency and optimality: the better the outcome and the less the cost needed to derive said outcome, the more competent the agent is. However, this is arguably closer to a definition of *efficiency* or *effectiveness*, and leaves unaddressed both the relative performative capabilities of different agents with respect to a given task's satisfactory completion (which is an essential component of competence [49]), and the function of competence as an indication of authoritative permissibility. To derive a new definition of competence for semi-autonomous systems, we bridge Gilbert's definition with the FEU's definition of *competency* as "a performance capability needed by workers in a specified occupational area," noting that "competency does *not* imply perfection; it implies a performance at a stated level" [51].

Hence, in this thesis we propose a formal, well-defined mathematical definition of the competence of a semi-autonomous system measured as the *true optimal extent of autonomous operation in any situation conditioned on the available human assistance and feedback*. Generally, the more competent the system is, the more that it can do autonomously, and the less reliance on human assistance it requires. This reliance on human assistance, however, necessitates well-described limitations on autonomous operation and explicit modeling of both the human-agent interactions itself, as well as where and when each is appropriate. Often, this has been represented by a set of *levels of autonomy* [107, 81], where each level corresponds to a set of constraints, limitations, or requirements on autonomous operation. In fact, this paradigm has already taken hold across multiple industrial fields where safety and reliability is critical in their exploitation of autonomous systems. The Society of Automotive Engineers (SAE) introduced perhaps the most well known articulation of levels of autonomy in the context of autonomous vehicles [96], but similar articulations have recently been presented in other fields such as the medical community [12, 40, 125] as well as the legal community [36, 37]. Common across these articulations is a clear progression in the extent of autonomy as the level increases, and the respective extent of human assistance provided and expected at each level. However, these examples also illustrate that the mechanisms of interactions, and the language with which they are articulated, can change with each domain and each implementation.

Several challenging factors must therefore be considered when designing such a system. The semi-autonomous system should only operate in a fully unsupervised autonomous capacity (i.e., without *any* human involvement) when it can be done safely and reliably, and should otherwise be able to identify the appropriate reliance on human assistance needed to optimally complete its task within any constraints imposed on its autonomous behavior. As the human cannot be expected to always have an intimate knowledge of the semi-autonomous system's underlying model and technical specifications, the system should be capable of engendering the appropriate level of *trust* from the human over the course of its deployment, which is an important factor in the level of reliance on the system, particularly in complex or unanticipated situations [64]. Determining the appropriate level of trust is itself a complex problem [53, 46] and may depend on several factors such as the relationship of the human to the system (e.g., designer, tester, consumer, etc.), the opaqueness of the system's abilities (e.g., using black-box methods versus more directly interpretable ones), and the criticality of the domain (e.g., a self-driving car versus a robotic vacuum). The challenge of precisely determining the appropriate level of trust *a priori* suggests that to be successful in any setting, the system should also have the capacity to update its own model of how best to interact with the human as the human's understanding of the system's abilities evolves over time. In effect, the system should maintain a well-defined model of its competence, capable of learning its true competence online through interactions with the human, and additionally have the means to improve its competence over time when possible. This suggests that an effective approach to modeling control of a system with with multiple levels of autonomy, multiple forms of human assistance, and multiple forms of feedback and communication, is necessary to efficiently capture the full breadth of problems considered. To this end, we introduce *competence-aware autonomy* as a paradigm for enabling a semi-autonomous system to learn and reason about its competence from interactions with a human operator.

1.1 Related Work

Researchers in automated planning [44] and reinforcement learning [112] have produced a vast literature devoted to models, languages and algorithms that enable agents to reason about their environment and choose actions intelligently. In this work, we specifically focus on advancing proactive reasoning under uncertainty about when and how to obtain human assistance to improve goal achievement or safety. We discuss below three areas of research that are particularly relevant to competence-aware autonomy.

1.1.1 Systems with Variable Levels of Autonomy

Recognizing the value of human knowledge in planning has led to several research efforts on human-agent collaboration in automated planning and control. *Mixed-initiative planning/control* [19, 39, 23, 43] is a paradigm based on *mixed-initiative interaction* [1, 56] wherein multiple different agents, generally a human an an autonomous system, can take the initiative to act at different stages to best utilize their respective abilities. Recent work has investigated applying mixed-initiative control in the context of variable autonomy [27] in which the level of autonomy (LoA) can change dynamically online. Chiou et al. [28] introduced the *expert-guided mixed-initiative control switcher*, which dynamically adjusts the level of autonomy based on a comparison of the expected performance of a task expert and the observed performance of the current system. Petousakis et al. [84] extended this approach by explicitly modeling the cognitive availability of the human based on real-time vision of the human to better inform the LoA switching decision between the autonomous agent and the human. Our work differs from this prior work in several key aspects. First, we assume that an automated planner determines the level of autonomy for the human-agent team, thereby designating the workload to both the human and the autonomous agent rather than allowing for each to initiate control on their own. Second, we are focused on the problem of learning the true competence of the human-agent system online through the acquisition of feedback from the human in response to actions taken by the agent at different levels of autonomy. Finally, much of the previous work is either tied to, or focused on, systems with only two levels of autonomy—no autonomy and full autonomy—whereas we emphasize a general model for arbitrary levels of autonomy.

Rigter et al. [90] considered a similar setting in which control of a system is selected from a set of autonomous controllers and a human operator. To reduce the reliance on the human over time, they propose to learn one of the controllers online from demonstrations gained from the human operator. While similarly motivated, we consider a slightly different problem setting. First, we consider only a single acting agent operating in different levels of autonomy, each of which may involve some degree of human assistance, rather than all-or-nothing involvement of the human, and allow for the level to change at every time step, rather than being fixed throughout an episode. The idea of learning a controller from human demonstrations is similar to how we propose to learn a model of the human's transition function when they are in control, but in our case we use it only to predict their behavior, not to learn or alter autonomous control.

Symbiotic autonomy is similar in that the aim is to enable the completion of complex tasks by distributing tasks and information across multiple agents. However, the term has been used both to represent human-agent systems where the two agents act asynchronously to perform tasks for *each other*, that is both the human and agent may seek assistance from the other to complete their task [94, 117, 118], as well as systems in which there is a *smart environment* in addition to the autonomous agent and human that provides auxiliary information to the autonomous agent to facilitate it [31, 97, 24]. Generally, our work differs in that we do not consider the environment and we emphasize the use of human assistance to better facilitate the completion of the autonomous agent's task, rather than asynchronously acting to help the other agent with their task.

Adjustable autonomy [78, 34, 102, 103, 17, 116, 127] is a closely related paradigm for human-agent teams that is characterized by the ability to dynamically change between different levels, or modes, of autonomy, each of which corresponds to some set of constraints or allowances that affect the actions the human-agent team can successfully perform. It is worth noting that these approaches are largely complementary, and there has been work specifically designed to combine multiple of these approaches [17, 76]. Our work falls generally in the category of adjustable autonomy, but adds two important capabilities to such systems, on top of the fundamental notion of competence. First, we explicitly model multiple forms of human feedback and use this feedback to enable a semi-autonomous system to learn its competence over time. Second, in the CAS model the system learns a predictive model of the human's feedback allowing the system to converge to the optimal level of autonomy over time.

1.1.2 Learning from Human Feedback

Much of the work in this thesis is highly related to the general area of learning from human feedback, which is both broad in scope and extensively studied. In reinforcement learning, some work has investigated the effect of additional information provided by a guiding human. Specifically, Chernova and Veloso [26] consider the inclusion of a guidance period after a robot's action, which can restrict the set of actions that the robot can take in the next step to improve the efficiency of the learning process. Moreira et al. [77] apply this method in the context of deep reinforcement learning to expedite the learning process of a deployed system in a new environment. Similarly, Rosenstein and Barto [93] propose a generalization to the actor-critic reinforcement learning framework [6] that includes a supervisor who can provide additional feedback to the system in the form of auxiliary guiding rewards, action selection guidance, or even direct control of the system. These differ from our work in that we assume that the agent has access to a well-defined and fully-specified model of its domain, including the reward (or cost) function from which to compute an optimal policy, and hence we are not concerned with learning a better world model online (rather, we are only concerned with learning the system's competence model online).

On the other hand, Knox et al. [58, 59] proposed a framework for training a robot *solely* from human feedback (sometimes called interactive shaping or interactive reinforcement learning) in which the human supervising the robot provides real-valued rewards for the actions that were just taken by the robot in a way that is assumed to account for the long-term impacts of the action. However, in our work we are not training the agent to act by learning a reward function, but rather providing the agent labeled data from which it can compute a distribution that is integrated into an explicit transition function. Additionally, we do not consider the use of real-valued feedback from the human, but rather discrete information tokens. More similar to our learning setting, Griffith et al. [48] proposed an approach in which the agent learns two policies in parallel, one derived from reward signals from the environment, and one derived from "right/wrong" labels from the human to infer what the *human* believes is the optimal policy, and then combines the two policies into one that is used for action exploitation. The key difference from Griffith et al. [48] is that we seek to learn a predictive model of the human's feedback rather than what the human believes the correct policy to be, and then use this predictive model to analytically determine the optimal policy given the domain model.

Ramakrishnan et al. [88] examined a problem similar to what we consider in Chapter 4, wherein an autonomous agent trained in simulation may have "blind spots" when deployed in real-world environments driven by missing or ignoring features that are important in the real-world. Similar to how our method exploits human feedback to identify new features that the human is using in generating their feedback, their method applies imitation learning to demonstrations collected from the human to identify features used by the human but not by the agent. Our work differs primarily in the type of information that the human provides to the system as well as how the missing features are used. We integrate the missing features into the existing model to improve the accuracy of the predicted human feedback, which consequently improves the quality of the overall policies generated by the system. On the other hand, [88] use the learned information to learn blind spot models in the real world to perform safe transfer-of-control to a human when encountering a blind spot to avoid potentially dangerous situations.

Bajcsy et al. [5] introduced a methodology in which a robot can learn an objective parameterized by a set of features via physical corrections made by a human supervisor, focusing on learning one feature at a time to reduce unintended learning from the human's interventions. Li et al. [65] and Liu et al. [67] both consider human-in-the-loop IRL settings in which an agent learns via reward signals generated by the environment to act in its domain while being constantly supervised by human operators who are ready and capable of intervening and taking control when the agent attempts something risky, providing a subsequent demonstration trajectory.

Constraint inference and learning from human feedback has also been of great interest in the robotics community, as it is generally easier to infer a set of constraints on allowable behavior from feedback that dictates what the agent can't do, rather than attempting to learn an explicit reward function that best explains the observed human feedback or even an entire policy. Additionally, constraints are often shared across many tasks and environments within a domain, which is useful for generalization [29]. Scobee and Sastry [105] present a method for maximum likelihood constraint inference in an inverse reinforcement learning setting that iteratively infers the constraints that best explain behavior observed in human demonstrations. Their work focused on purely deterministic systems, but was later extended to non-deterministic systems [73] and continuous, model-free settings [70]. Papadimitriou et al. [80] introduced a Bayesian constraint inference method based on human demonstrations that, unlike maximum likelihood methods, is able to work with both partial trajectories and sets of disjoint state-action pairs, in addition to the full demonstration trajectories used in maximum likelihood inference.

To the best of our knowledge, the work we present in Chapter 6 is the first to explicitly study both the problem of learning constraints from interventions specifically, as well as learning from proactive feedback. Here, we use the term intervention to indicate any discrete instance where a human intervenes in the agent's execution and brings the agent to a new state. Spencer et al. [110] considered a similar problem in their *expert intervention learning* framework, which leveraged demonstrations that were generated during *interventions* gated by a human supervisor's decision to learn a portion of the state-action space that is deemed to be "good enough" to operate within. The agent's objective is consequently to minimize the time spent outside of the "good enough" region while simultaneously minimizing misclassification of intervention actions. However, their approach still relies on partial trajectories of human behavior and assumes non-proactive interventions.

1.1.3 Competence Modeling

The term *competence* has been used widely in the context of intelligent systems. The classification literature, in particular, has often defined the term as some measure of performance based on standard metrics for classification systems on their input space [61], including accuracy estimation [122], potential function estimates [89], Bayes-based confidence measures [55], relative performance to random guessing or otherwise randomized classifiers [119], and probabilistic models [68, 120, 121]. More recently, Platanios et al. [85] defined the competence of a curriculum learner to be the proportion of training data that the learner is allowed to use at any given time based on the difficulty of training samples, and Rabiee et al. [87] proposed competence as a distribution over failure classes that is learned via introspective perception in the context of robotic path-planning. Common across these examples is an evaluative approach to defining competence; that is, competence is a measure of the *performance* of a system or algorithm. Most closely related to the formalization of competence presented in this thesis was suggested by Smyth and McKenna [108] who defined the competence of a case-based reasoning (CBR) system as the set of problems that the system can solve successfully. The authors provide a rigorous model and analysis of competence for CBR systems, but the work is highly specific to CBR systems on non-probabilistic domains, and consequently does not apply to stochastic decision-making processes considered in this work. Rather, we aim to enable a system to handle *all* problems by utilizing the appropriate level of human assistance to ensure safe operation.

1.2 Summary of Contributions

The main objective of this thesis is to enable semi-autonomous systems to operate more reliably in the open world by explicitly accounting for violations of common assumptions made in contemporary planning models. We first introduce the primary planning model used in our competence-aware autonomy paradigm, the *competence-aware system* (CAS) in Chapter 3, as a solution for handling problems where a system will be faced with unexpected scenarios during deployment, and is initially unaware of what it can handle safely and reliably. We propose the model initially in the context of full state observability, but later generalize the model to partially-observable domains as well, and define a mathematically precise notion of *competence* for such systems based on levels of autonomy. Additionally, we provide both a theoretical analysis of the convergence properties of a competence-aware system in addition to rigorous empirical evaluations of its performance in simulation.

In each subsequent chapter of the thesis, we explore the challenge(s) that arise from relaxing one or more assumption made in the base formulation, and propose a novel model or method as a solution. In Chapter 4, we present a method called *iterative state space refinement* that enables a CAS to identify from emergent inconsistencies in the human's feedback features missing from its initial model, to handle scenarios where it is impossible to know *a priori* all features relevant to the human's decision-making. We prove that, when possible, our method will find all such missing features and reach a point where the human's feedback is well-discriminated everywhere. Chapter 5 proposes a generalization of the CAS model called a *contextual CAS* (CoCAS) that relaxes the assumption of a single, invariant human operator, to allow for multiple, heterogeneous, stochastic human operators where competence may be conditioned on additional contextual parameters of the domain. We prove that this model strictly generalizes both the CAS model, and the recently proposed SO-SAS model [32], but still retains the same convergence guarantees of the base

CAS model, and performs the best empirically across several metrics. Finally, Chapter 6 proposes a methodology for learning constraints from human feedback that is temporally conditioned on the inferred future behavior of the system by the human, rather than on the current state and action as is often assumed in the the learning from feedback literature. We empirically demonstrate the benefits of the proactive learning approach in two domains wherein the proactive learning model significantly outperforms the reactive model, and prove that the optimal policy under the learned constraint model will be at most a constant factor worse than the optimal policy under the true constraint model in finite-horizon problems. Finally, we offer concluding thoughts on the use, advantages, and limitations of competence-aware autonomy, and the important avenues for future work that remain.

Competence-Aware Systems

In the open world, complete *a priori* determination of the competence of a human-agent system, and the respective capabilities of each actor in the system, is generally either impractical or impossible to do across all situations that the system may encounter. As a result, the system's initial policy is likely to be either over-reliant or under-reliant on the human in many circumstances. Without an ability to learn and adjust the appropriate level of autonomy of a deployed system in a *safe* way, the chance of failure and the resources wasted will grow over time. In expensive or safety-critical domains such as autonomous driving and space exploration, the resultant cost can be very high. Consequently, developing formal mechanisms to explicitly represent, reason about, and optimize the autonomy of a system is an important challenge in artificial intelligence. To address this issue, we introduced a formal model called *competence-aware systems* (CAS) for optimizing autonomy in semi-autonomous systems over time by learning to optimally leverage the available human assistance. CAS is a decision making framework for semi-autonomous systems where systems can operate at multiple levels of autonomy, each of which corresponds to different constraints on autonomous operation and their commensurate levels of human assistance, each associated with a set of unique feedback signals. By learning from these feedback signals, the CAS can quickly grow to operate at its competence when feasible, effectively optimizing its autonomous operation by minimizing unnecessary reliance on human intervention while relying on the human when optimal to do so.

Improving Competence with Iterative State Space Refinement

Like many fixed planning models, a competence-aware system is limited in what it can learn and represent by its initial fixed model. In practice, particularly when the human agent in the system is not a designer of the system, it is likely that there will be divergence between the autonomous agent's model of the world and the human's model of the world. The result is that feedback provided by the human may appear noisy or random to the agent, which may not have the representational capability of distinguishing the feedback appropriately. This phenomenon can lead to low competence and inefficient operation, causing a decrease in the human's trust of the agent's capabilities. To address this, we propose a method called *iterative state space refinement* that enables the competence-aware system to refine the granularity of its state representation through online model updates. This process produces a more nuanced partitioning of the state-action space with different levels of competence, allowing the system to better learn and act at its true competence.

Contextual Competence-Aware System

The standard CAS model assumes that there a single, invariant human with perfectly consistent feedback. However, in many real-world domains, that assumption may be unreasonable; there may, for instance, be multiple human operators interacting with the autonomous agents, each may have their own unique set of skills or preferences, and their performance and internal state may vary stochastically dependent on contextual variables like the time of day. These may in turn affect the competence of the overall human-agent system. To account for this, we propose a generalization of the CAS model called a *contextual CAS* (CoCAS), which can operate in multiple levels of autonomy and is capable of optimizing its autonomy with respect to global contextual information about the world, and local contextual information about each stochastic, heterogeneous operator.

Learning Constraints on Autonomy from Proactive Feedback

Both in our approach to learning competence from human feedback, and the general literature on learning from human interventions, it is often assumed that feedback is provided *reactively* by the human, i.e., for the current or previously executed state-action pair of the system. However, evidence in human cognitive control suggests that human's often act *proactively* when trying to ameliorate anticipated goal interference prior to occurrence. Consequently, in this chapter we propose a novel learning methodology for learning from proactive interventions, where the human's feedback is not reactively generated for the system's current state-action pair, but instead for inferred potential autonomy constraint violations in the projected near-term future. We show that our approach makes minimal assumptions about the human, but still enables the system to learn a model of autonomy constraints from proactively generated human feedback in the form of sparse interventions.

1.2.1 Relevant Publications

- C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, and S. Zilberstein. Learning to Optimize Autonomy in Competence-Aware Systems. *In Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).* 2020.

- C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, and S. Zilberstein. Improving Competence for Reliable Autonomy. *In Proceedings of the ECAI Workshop on Agents and Robots for reliable Engineered Autonomy (AREA).* 2020.

- C. Basich, J. Svegliato, A. Beach, K. H. Wray, S. Witwicki, and S. Zilberstein. Improving Competence via Iterative State Space Refinement. *In Proceedings of the International Conference on Robotics and Automation (ICRA).* 2021.

- C. Basich, J. Peterson, and S. Zilberstein. Planning with Inconsistent Sensors: Knowing When to Act Blind. *In Proceedings of the IJCAI Workshop on Robust and Reliable Autonomy in the Wild (R2AW).* 2021.

- C. Basich, K. H. Wray, S. Witwicki, and S. Zilberstein. Introspective Competence Modeling for AV Decision Making. US Patent App. 16/668,584.

- C. Basich, J. Peterson, and S. Zilberstein. Planning with Intermittent State Observability: Knowing When to Act Blind. *In Proceedings of the International Conference on Intelligent Robots and Systems (IROS).* 2022.

- C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, and S. Zilberstein. Competence-Aware Systems. *In* Artificial Intelligence Journal *(AIJ).* 2023.

- C. Basich, J. Biswas, and S. Zilberstein. Competence-Aware Autonomy: An Essential Skill for Robots in the Real World. *In Proceedings of the AAAI Bridge Session on AI and Robotics (AAAI-AIROB).* 2023.

- S. Mahmud*, C. Basich*, and S. Zilberstein. Semi-Autonomous Systems with Contextual Competence Awareness. *In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).* 2023.

CHAPTER 2
BACKGROUND

In this chapter we provide as background the formal definition and representation of several key planning models for stochastic sequential decision making. Figure 2.1 illustrates the traditional methodological view of robotic control, and while the models discussed in this chapter all fall within the "planning" component of the architecture, we emphasize that – as illustrated – *planning does not exist in a vacuum.* In particular, the motivation and formalism of key planning models arise directly from limitations of, or considerations for, the other components.

Common to all models for sequential decision making is a formal representation of (1) the states and their features that describe the domain or environment, (2) the actions that the agent operating in the domain can execute, (3) the dynamics of how the world evolves by itself or in response to the actions performed by the agent, and (4) the dynamics of how the agent is rewarded or penalized for the actions that it executes and how the world changes around it.

While there are numerous models for sequential decision making, we focus our background section on only those used explicitly throughout the course of this thesis as the basis of one or more contribution. Our focus is on models that can represent stochastic domains, where each action performed by the agent may probabilistically lead to more than one successor state. In particular, we provide background on (1) two *fully observable, single-objective* models – the *Markov decision process* (MDP) and the *stochastic shortest path problem* (SSP) – in which at all times, the agent is aware of exactly what state it is in and the exact values of all current state features, and

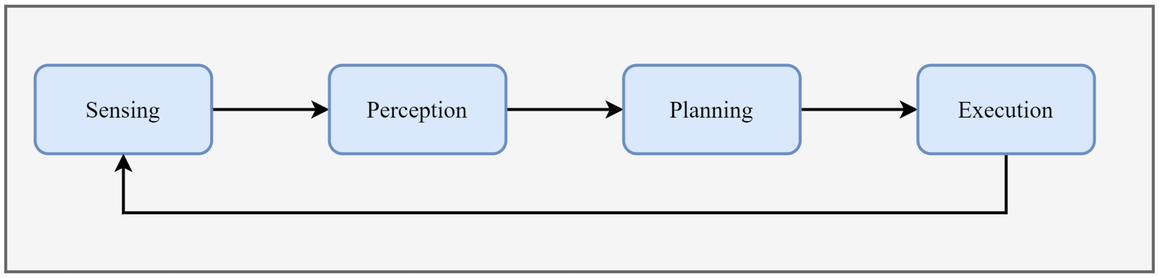


Figure 2.1: The traditional robot-control methodology, adjusted for contemporary separation of "sensing" and "perception".

the system only seeks to optimize a single objective function; (2) two *fully-observable, multi-objective models* – the *multi-objective Markov decision process* (MOMDP) and the *lexicographic Markov decision process* (LMDP) – which extend the MDP to situations where the agent must optimize over more than one objective functions in different ways; and (3) two models for *partially-observable, single-objective* domains – the *partially-observable Markov decision process* (POMDP) and *belief-state Markov decision process* (bMDP) – in which the agent may have limited knowledge of what state it is in at any given time and the true features of that state.

2.1 Fully-Observable, Single-Objective Planning

Fully-observable, single-objective models represent the simplest class of stochastic decision making models used in this thesis, and are widely-used due to their flexibility, computational tractability, and guarantees on solution optimality.

2.1.1 Markov Decision Process

A Markov decision processes (MDP) is a formal model for sequential decision making in fully observable, stochastic environment, and provides the foundation of all of the decision-making models used in this thesis. Many of these models exist to handle problems where one or more assumption made in formulating a problem as an MDP (e.g., full state observability) does not hold; however, MDPs have been shown to be effective in a wide array of domains [38].

Definition 1. A **Markov decision process (MDP)** is represented by the tuple: $\langle S, A, T, R, \gamma \rangle$ where

- S is a finite set of states,
- A is a finite set of actions,
- $T : S \times A \times S \to [0, 1]$ is a transition function representing the probability of arriving in state $s' \in S$ having taken action $a \in A$ in state $s \in S$,
- $R : S \times A \to \mathbb{R}$ is a reward function representing the myopic utility of taking action $a \in A$ in state $s \in S$, and
- γ is a discount factor that represents the discounted value of future rewards.

An MDP is a discrete-time control process where, at each time step the system performs an action $a \in A$ in state $s \in S$, receives the immediate myopic reward $R(s, a)$, and then transitions to a new state $s' \in S$ drawn according to $T(s, a, s')$. An MDP can be either finite-horizon, in which case the process reaches until exactly $h \in \mathbb{N}$ time steps are reached at which point the process terminates, infinite-horizon, wherein the process continues *ad infinitum*, or indefinite-horizon, in which case there is a finite horizon $h \in \mathbb{N}$, but h is not known ahead of time.

A solution to an MDP is a mapping from states to actions, $\pi : S \to A$, called a **policy**, and we denote by Π the space of all policies. In an infinite horizon MDP, the objective is to find the policy $\pi \in \Pi$ that maximizes the expected reward over all states for all time, discounted by $\gamma \in [0, 1)$:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s^t, \pi(s^t)) | \pi\Big] \tag{2.1}$$

where s^t denotes the random variable for the state of the system at time step t following the transition function T. Any valid policy π induces the state–value function $V^\pi : S \to \mathbb{R}$ using the *Bellman equation*

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s') \tag{2.2}$$

that represents the expected cumulative discounted reward $V^\pi(s)$ when starting in state s following the policy π. Similarly, π induces the action–value function $q^\pi : S \times A \to \mathbb{R}$

$$q^\pi(s, a) = R(s, a) + \sum_{s' \in S} T(s, a, s') V^\pi(s') \tag{2.3}$$

that represents the expected cumulative discounted reward when the action $a \in A$ is taken in state $s \in S$, and the policy π is thenceforth.

Any policy that maximizes these functions is referred to as an optimal policy and denoted π^*; formally:

$$\pi^* := \operatorname*{argmax}_{\pi \in \Pi} V^\pi \tag{2.4}$$

Without loss of generality we may assume the optimal policy is unique unless explicitly stated otherwise. Given π^*, we can define the optimal state–value function following policy π^* using the *Bellman optimality equation* as follows:

$$\begin{aligned} V^*(s) &= \max_{a \in A} \Big[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \Big] && (2.5) \\ &= \max_{a \in A} q^*(s, a), && (2.6) \end{aligned}$$

where q^* is the action–value function under the policy π^*. Usefully, then, we can intuitively map $\pi^*(s) = \operatorname{argmax}_{a \in A} q^*(s, a)$. Any policy that is independent of time is called a *stationary policy*, and it is known that infinite-horizon MDPs admit at least one optimal stationary policy [15]. It is often convenient for infinite-horizon problems to include states that have the property that $T(s, a, s) = 1.0$ for every $a \in A$; these are referred to as *terminal absorbing states*.

For a *finite-horizon* MDP, there exists some known horizon $h \in \mathbb{N}$ where the objective becomes to find the policy $\pi \in \Pi$ maximizing the expected reward over all states within the horizon h, discounted by $\gamma \in [0, 1]$:

$$\mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t R(s^t, \pi(s^t))|\pi\Big]. \tag{2.7}$$

Unlike in the infinite-horizon case, policies in the finite-horizon setting can be non-stationary, inducing the time-dependent state-value function $V^\pi : S \times [h] \to \mathbb{R}$:

$$V^\pi(s,t) = R((s,t), \pi(s,t)) + \sum_{s' \in S} T((s,t), \pi(s,t), (s', t+1))V^\pi(s', t+1) \tag{2.8}$$

where $V^\pi(s, h) = 0$ for every $s \in S$.

2.1.2 Stochastic Shortest Path Problem

A **stochastic shortest path problem (SSP)** is a specific case of a Markov decision process that, while similar, is often more intuitive particularly when dealing with problems like navigation or task completion. An SSP is an MDP with a specified start state and (possibly singleton) set of goal states, where the objective is to reach any goal state from the start state while incurring the least cost possible. This is in contrast to an MDP, which has no notion of start or goal states, and in which an agent is seeking to maximize some reward. An SSP is always undiscounted and indefinite-horizon.

Definition 2. An SSP is formally represented by the tuple: $\langle S, A, T, R, s_0, G\rangle$ where:

- S is a finite set of states.
- A is a finite set of actions.

- $T : S \times A \times S \to [0, 1]$ is a transition function representing the probability of arriving in state $s' \in S$ having taken action $a \in A$ in state $s \in S$.

- $C : S \times A \to \mathbb{R}^+$ is a positive cost function representing the myopic cost of taking action $a \in A$ in state $s \in S$.

- $s_0 \in S$ is the start state.

- $G \subset S$ is the set of goal states.

As with an MDP, a solution to an SSP is a policy $\pi : S \to A$ that indicates that action $\pi(s) \in A$ should be taken in state $s \in S$, where the objective is to find the policy $\pi \in \Pi$ that minimizes the cumulative cost incurred when starting in state s_0:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t C(s^t, \pi(s^t)) | \pi, s_0\Big]. \tag{2.9}$$

However, the existence of an optimal solution to the SSP is guaranteed only under the condition that there exists a *proper policy*, i.e., a policy under which a goal state is reachable from all states with probability 1, and that all *improper policies* generate infinite cost when starting from at least one state; under this assumption, the optimal value function is also unique.

A policy π induces the state–value function $V^\pi : S \to \mathbb{R}$

$$V^\pi(s) = C(s, \pi(s)) + \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s'), \tag{2.10}$$

which represents the expected cumulative *cost* of reaching any $s_g \in G$ from state $s \in S$ following the policy π.

Naturally, there exists a similar action–value function, $q^\pi : S \times A \to R$

$$q^{\pi}(s,a) = C(s,a) + \sum_{s' \in S} T(s,a,s') V^{\pi}(s'), \tag{2.11}$$

which represents the expected cumulative cost of reaching any $s_g \in G$ from state $s \in S$ given that the action $a \in A$ was taken and the policy π was followed thenceforth. Any policy that minimizes these functions is referred to as an optimal policy and denoted π^*; formally:

$$\pi^* := \operatorname*{argmin}_{\pi \in \Pi} V^{\pi}(s_0) \tag{2.12}$$

Without loss of generality we may assume the optimal policy is unique unless explicitly stated otherwise. Given pi^*, we can define the optimal state–value function following policy π^* using the *bellman optimality equation* as follows

$$V^*(s) = \min_{a \in A} \Big[C(s,a) + \sum_{s' \in S} T(s,a,s') V^*(s') \Big] \tag{2.13}$$

$$= \min_{a \in A} q^*(s,a) \tag{2.14}$$

where q^* is the action–value function under the policy π^*.

2.2 Fully-Observable, Multi-Objective Planning

Although MDPs and SSPs have been used in a wide variety of domains, not all domains can be captured easily, or at all, through the use of a single objective function. For example, in a self-driving car, one may reasonably consider that minimizing the cumulative time to reach a destination is the primary objective to optimize when driving; however, other objectives such as maximizing safety, complying with the legalities of surface driving, and behaving in a socially-acceptable manner are all important factors that a successful system *must* reason about when determining its behavior. Consequently many systems need to solve for policies that optimize their performance across multiple competing objectives.

2.2.1 Multi-Objective Markov Decision Process

A *multi-objective Markov decision process* (MOMDP) is a formal model for sequential decision making in fully-observable environments where the agent's behavior is dictated by multiple competing objectives which each induce a unique reward (or cost) function. In a MOMDP, the objective of the agent is to find the policy that maximizes the cumulative value over all objectives according to a given weighting scheme.

Definition 3. A MOMDP is formally represented by the tuple $\langle S, A, T, \mathbf{R}, f_{\mathbf{w}}, \gamma \rangle$, where:

- S is a finite set of states.
- A is a finite set of actions.
- $T : S \times A \times A \rightarrow [0, 1]$ is a transition function representing the probability of arriving in state $s' \in S$ having taken action $a \in A$ in state $s \in S$.
- $\mathbf{R} = [R_1 \cdots R_k]^T$ is a vector of reward functions, $R_i : S \times A \rightarrow \mathbb{R}$.
- $f_{\mathbf{w}} : \mathbb{R}^k \rightarrow \mathbb{R}$ is a weighting function parameterized by a k-length vector $\mathbf{w}$.

A MOMDP operates analogously to a traditional MDP, and the objective remains to find the policy that maximizes the expected rewards over all states:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t f_{\mathbf{w}} \mathbf{R}(s^t, \pi(s^t)) | \pi\Big] \tag{2.15}$$

and

$$\mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t f_{\mathbf{w}} \mathbf{R}(s^t, \pi(s^t)) | \pi\Big] \tag{2.16}$$

for the infinite, and finite cases respectively.

As in an MDP, the solution to a MOMDP is a policy $\pi : S \rightarrow A$ that induces a value function, however in a MOMDP the weighting function is used to produce a single value function of the form $V^{\pi}_{\mathbf{w}}(s) = f_{\mathbf{w}}(\mathbf{V}^{\pi}(s))$ where $\mathbf{V}^{\pi} = [V^{\pi}_1(s) \cdots V^{\pi}_k(s)]$ is the vector of value functions computed as in an MDP under the policy $\pi \in \Pi$ for each of the k objectives. We specifically consider only the common *linear scalarization* weighting method in which $f_{\mathbf{w}}(\mathbf{V}^{\pi}(s) = \mathbf{w}^T\mathbf{V}^{\pi}(s)$, where each weight element $w_i \in \mathbf{w}$ is a positive real number, and $\mathbf{w}$ sums to 1.

2.2.2 Lexicographic Markov Decision Process

A *lexicographic Markov decision process* (LMDP) is similar to the MOMDP in that it aims to optimize performance across multiple different objectives. However, in an LMDP, objectives are defined in a strict lexicographic ordering, where any value in a given objective is weighted more heavily than any value in a different objective lower in the ordering.

Definition 4. An LMDP is formally represented by the tuple $\langle S, A, T, \mathbf{R}, \mathbf{\Delta}, o\rangle$ where:

- S is a finite set of states.
- A is a finite set of actions.
- $T : S \times A \times A \to [0, 1]$ is a transition function representing the probability of arriving in state $s' \in S$ having taken action $a \in A$ in state $s \in S$.
- $\mathbf{R} = [R_1 \cdots R_k]^T$ is a vector of reward functions, $R_i : S \times A \to \mathbb{R}$.
- $\mathbf{\Delta} = \langle \delta_1, ..., \delta_{k-1}\rangle$ is a tuple of *slack* variables with each $\delta_i \in \mathbb{R}^+$ for every $i \in [k - 1]$.
- $o = \langle o_1, ..., o_k\rangle$ is a strict ordering over the k objectives.

As above, a solution to an LMDP is a policy $\pi : S \to$; however, due to the lexicographic ordering over objectives, there is no close-form value function representation to solve for. Instead, solving an LMDP involves sequentially optimizing for each objective o_{i+1} given a solution to objective o_i and slack value δ_i, denoting the maximum allowable deviation from the optimal expected reward for objective o_i when objectives lower in the lexicographic order. This can be done by constraining the available actions when solving for objective o_{i+1} by setting $A_{i+1}(s) = \{a \in A | \max_{a' \in A_i} Q_i(s, a') - Q_i(s, a) \leq \gamma_i\}$ [123].

2.3 Partially-Observable, Single-Objective Planning

The earlier models presented all rely on the assumption of full state observability; that is, at every time step, the system fully, and exactly, knows its current state. This is a useful assumption because it (1) applies in many domains of interest and (2) makes the decision-making problem tractable to solve. Unfortunately, many systems in the open world, such as robots that rely on sensor data to infer their state, do not exhibit this property, and assuming that it holds can lead to poor, or even dangerous, performance. Instead, the system may need to reason over what state it may be in at any given time given partial information about its state in the form of noisy observations, and act according to this *belief* over its current state.

2.3.1 Partially Observable Markov Decision Process

A *partially observable Markov decision process*, or POMDP, is a formal model for sequential decision making in partially observable, stochastic environments. In other words, a POMDP is an MDP where there is uncertainty over what state the agent is actually in. In a POMDP, rather than observing the full state of the world directly, the system receives state information in the form of *observations*, which may provide only partial information about the current state of the world. Based on these observations, the system determine what the state of the world *might be* and act accordingly.

Definition 5. A POMDP is formally represented by the tuple $\langle S, A, \Omega, T, R, O, \gamma \rangle$ where:

- S is a finite set of states.
- A is a finite set of actions.
- Ω is a finite set of observations.
- $T : S \times A \times S \to [0, 1]$ is a transition function representing the probability of arriving in state $s' \in S$ having taken action $a \in A$ in state $s \in S$.
- $R : S \times A \to \mathbb{R}$ is a reward function representing the myopic reward of taking action $a \in A$ in state $s \in S$.
- $O : A \times S \to \Delta^{|\Omega|}$ is an observation function representing the likelihood over observations given that action $a \in A$ was taken and the agent ended up in state $s \in S$.
- $\gamma \in [0, 1]$ is a discount factor.

As in an MDP, a POMDP can either be finite-horizon, infinite-horizon, or indefinite horizon, for horizon h, and at each time step $t \geq 0$ the system performs an action $a^t \in A$ in state $s^t \in S$. However, the system does not know its current state, and consequently must instead infer its true state from the history of its behavior and observations $\langle s_0, a_0, \omega_1, ...a_{n-1}, \omega_n \rangle$. This history can be expressed as a *belief state*, which is a probability distribution over states; i.e., if we define B to be the $|S|$–dimensional simplex, $\Delta^{|S|}$, then a belief state b is any point in B. We use the notation $b(s)$ to represent the belief of being in state $s \in S$ under the belief state $b \in B$. A belief state can be computed directly from the actions and observations taken by the agent, given that it knows what state it started in. Formally, after executing action $a \in A$ and observing observation $\omega \in \Omega$, the agent updates its belief state $b \in B$ to some

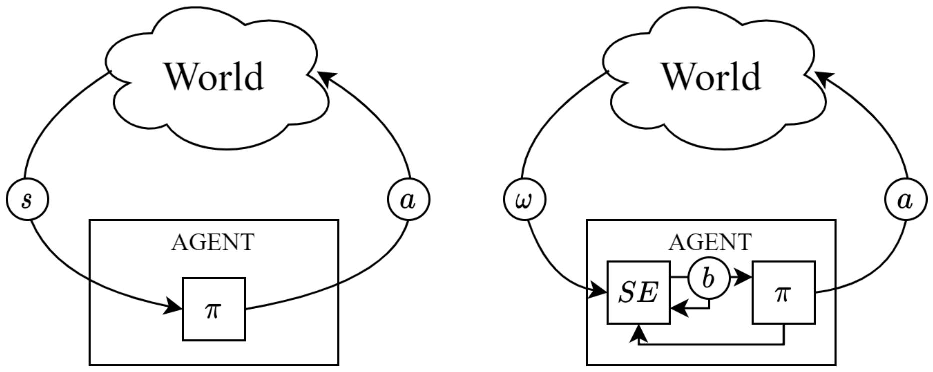


Figure 2.2: A illustration of the difference between a fully observable MDP (left) and a partially observable MDP (right). In an MDP, the agent receives the true state (s) of the world at each time step and acts directly on that. In a POMDP, the agent receives an observation, ω, about its state in the world, enabling it to compute its belief over which state it is in and act accordingly.

new belief state $b' \in B$ using the following **belief–state update equation** for every state $s' \in S$:

$$b'(s'|b,a,\omega) = \eta O(\omega|a,s') \sum_{s \in S} T(s,a,s') b(s) \tag{2.17}$$

where $\eta = \frac{1}{Pr(\omega|b,s')}$ is a normalizing constant.

In a POMDP, a policy, $\pi : B \to A$, maps belief-states to actions. In an infinite-horizon POMDP, the objective is to find the policy $\pi \in \Pi$ that maximizes the expected reward over all belief-states over all time with discount $\gamma \in [0,1)$:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(b^t, \pi(b^t))|\pi, b^0\Big] \tag{2.18}$$

where b^t denotes the random variable for the belief-state of the system at time step t following the transition function T and observation function O, and starting in belief state b^0, and $R(b,a) = \sum_{s \in S} b(s)R(s,a)$. Similarly, for a finite-horizon POMDP with horizon $h \in \mathbb{N}$ and discount factor $\gamma \in [0,1]$, we seek to find the policy maximizing the cumulative reward over all states within the h time steps:

$$\mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t R(b^t, \pi(b^t))|\pi, b^0\Big] \tag{2.19}$$

However, it is often convenient to map POMDPs to another model, called a *belief-state MDP*, as discussed below, when solving for the optimal policy. Note that, partial-observability conditions have also been applied to SSPs in a similar capacity [82].

2.3.2 Belief-State Markov Decision Processes

Every POMDP can be equivalently expressed as a **belief-state MDP**, which is a Markov decision process where the states are replaced by the belief state space B that is expressed above, and where the action set remains the same.

Definition 6. G a POMDP $\langle S, A, \Omega, T, R, O, \gamma \rangle$, the corresponding belief-state MDP is represented by the tuple $\langle B, A, \tau, r, \gamma \rangle$ where:

- $B = \Delta^{|S|}$ is the set of belief states created from the POMDP.
- A is the finite set of actions from the POMDP.
- $\tau : B \times A \times B \rightarrow [0, 1]$ is the belief state transition function representing the probability of reaching the belief $b' \in B$ after taking the action $a \in A$ is performed in belief state $b \in B$.
- $r : B \times A \rightarrow \mathbb{R}$ is the belief state reward function representing the myopic reward for taking action a in belief state $b \in B$.
- $\gamma \in [0, 1]$ is a discount factor.

When the state and observation spaces in the original POMDP are discrete, we can define the belief-state transition function, τ, as follows:

$$\tau(b, a, b') = \sum_{\omega \in \Omega} Pr(b'|b, a, \omega) \sum_{s' \in S} O(\omega|a, s') \sum_{s \in S} T(s, a, s')b(s) \tag{2.20}$$

Here, $Pr(b'|b, a, \omega)$ is computed using the belief update rule discussed above. We can define the belief-state reward function as well, to simply be the expected reward over the belief states:

$$r(b, a) = \sum_{s \in S} b(s) R(s, a) \tag{2.21}$$

Given a belief-state MDP, we can define the optimal state-value function, $V^* : B \to \mathbb{R}$ as follows:

$$V^*(b) = \max_{a \in A} \left[r(s, a) + \gamma \sum_{\omega \in \Omega} Pr(\omega | b, a) V^*(\mathit{update}(b, a, \omega)) \right] \tag{2.22}$$

where $\mathit{update}(b, a, \omega)$ represents the belief state, b', that is achieved by applying the belief update equation (Eq. 2.17) over all states s, given the belief state b, action a and observation ω.

Given a belief-state MDP, we can define the optimal action-value function, $q^* : B \times A \to \mathbb{R}$ as follows:

$$q^*(b, a) = r(s, a) + \gamma \sum_{\omega \in \Omega} Pr(\omega | b, a) V^*(\mathit{update}(b, a, \omega)) \tag{2.23}$$

Finally we can define the optimal policy, $\pi^* : B \to A$, as follows:

$$\pi^*(b) = \operatorname*{argmax}_{a \in A} q^*(b, a) \tag{2.24}$$

Naturally, the belief-state MDP extends in the analogous fashion for partially-observable SSPs (sometimes referred to as Goal POMDPs) [16].

2.4 Conclusion

In this chapter, we present the core decision-making models that are used throughout the remainder of this thesis, each of which is a specific type, or generalization of, the traditional Markov decision process. Each model has different strengths and weaknesses, often trading computational tractability for model or objective expressivity, meaning that no one model fits all problems, and selecting the appropriate model for the setting is a critical aspect creating successful autonomous systems in the open world.

CHAPTER 3

COMPETENCE-AWARE SYSTEMS

As discussed in Chapter 1, the vast majority of autonomous systems under development are in fact *semi-autonomous systems* (SAS) that can operate autonomously under certain conditions, but may require human intervention or aid to ensure that they achieve their assigned goals [128]. For example, a space exploration rover may suspend operation and wait for a new plan from the command center when its wheel encounters unexpected resistance; an autonomous car may request that the driver take over when an unexpected obstacle blocks its path requiring it to drive into the oncoming traffic's lane to circumvent. Human assistance could come in different forms that correspond to different limitations of the autonomous system; e.g., allowing a system to operate autonomously *under human supervision* may indicate a higher level of competence relative to a system that must first present its plan and get approval for every action before the action is executed.

Reliance on human assistance has been explored extensively to address the limited competence of autonomous systems [39, 97, 118, 43, 54, 78, 84]. Often, this has been explored in the context of *varying levels of autonomy*, a paradigm for modeling gradations in autonomous behavior within a human-agent team [107, 81], where each level of autonomy corresponds to some set of constraints, limitations, or requirements on autonomous operation. For example, on the two extremes would be full autonomous operation, and full human control (no autonomy). Levels of autonomy have been proposed in several industrial applications where safety and reliability are critical, including driving automation [96], robotic medical devices [12, 40, 125], and autonomous legal reasoning [36, 37].

Human assistance may be available in different forms or modalities, corresponding to different degrees of competence of a semi-autonomous system. Different forms of human assistance compensate for the limitations imposed in each level of autonomy and consequently mitigate the potential for risky behavior, while still ensuring that the system's task is completed. For example, Veloso et al. [117, 118] designed the CoBot system that can aid humans in an office environment as an assistive robot in a variety of pick-up and delivery tasks. However, as the CoBot has no arms to grasp objects, it cannot perform its tasks entirely autonomously, and must instead seek assistance from humans to compensate for its limitation, for example by placing or removing objects in its basket. Ficuciello et al. [40] proposed a level of autonomy framework for a surgical assistive medical robot with four levels of autonomy, where the lowest two involve purely assistive actions to aid the human who is the primary executor, and the highest two involve fully autonomous execution by the robot with assistance from the human in the form of surgical strategy selection.

In this dissertation, we are primarily concerned with the risk associated with a system that operates at a level of autonomy that is inappropriate for a task given its capabilities; for instance, an office robot that autonomously handles fragile items it is not competent to handle safely (i.e., without a high risk of breaking). Hence, we aim to develop systems that are aware of their *own competence*, which we define to be the *optimal level of autonomy* to employ in any given situation conditioned on the availability of suitable human assistance. A system that is aware of its own competence when generating plans can therefore mitigate the potential for risky behavior by optimizing the degree of human assistance that it requests, leveraging the human where the system's competence is low, and acting autonomously where it is high.

To further mitigate risks, humans may impose constraints on autonomous operation based on the perceived competence of the system, for instance, by allowing them to intervene in time to prevent risky behavior or by disallowing autonomous behavior entirely. In fact, the perceived risks may be outside the scope of what the autonomous system can detect or reason about, hence enabling us to mitigate a broader range of risks. For example, a robot's sensors may be unable to perceive black ice on a sidewalk, or a nearby obstacle in dense fog, leading to risky behavior if left to operate without supervision in these conditions.

Determining the exact competence of an autonomous system at design time can be very difficult, particularly when the environment is not fully specified, or is simply too complex to fully anticipate, *a priori*. For example, a self-driving car may initially be authorized to drive autonomously without supervision only on highways and during the daytime with clear weather. Hence, an initial level of autonomy may be determined *a priori* through testing and evaluation, but adjustments must be made when the system is deployed. Even when developers aim to err on the side of caution, initializing the level of autonomy to be below the system's true competence, it is possible to unintentionally infer from initial testing that the system is more competent than it really is [113, 86]. Therefore, developing mechanisms to explicitly represent, reason about, and adjust the level of autonomy is critical for the success of autonomous systems deployed in the open world.

We propose a planning model called *competence-aware system* (CAS) for operating at multiple levels of autonomy where each level is associated with different forms of human assistance that compensate for the constrained abilities of the system [7, 10, 11]. Motivated by ideas from *collaborative control* [41], the structure of a CAS is illustrated in Figure 3.1. The model associates with each type of human assistance a set of feedback signals that the system can receive from the human, the likelihood of which can be learned over time. This model enables the system to operate more

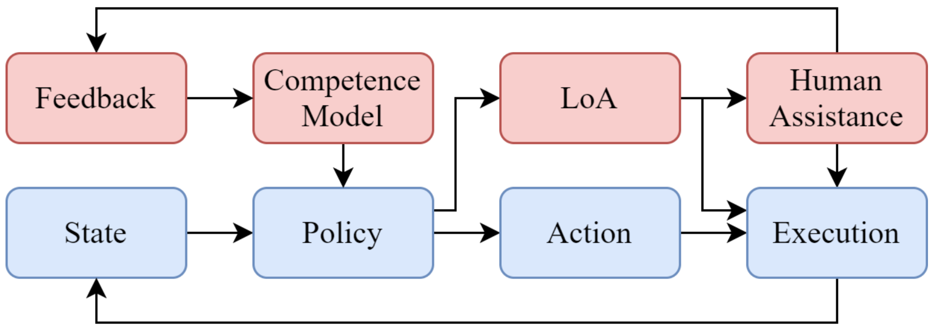


Figure 3.1: An overview of how competence modeling impacts plan execution. Here, the system's current state is provided as input to the system's policy, which traditionally would only output an action, but in our case also outputs a level of autonomy determined by the competence model in which to perform the action. The level of autonomy dictates the type and degree of human assistance used in the execution of the intended action. The human assistance can also provide additional feedback to the system, which can be used to update and refine the competence model online.

reliably in the open world, reduce improper reliance on the human and ultimately optimize the autonomous behavior of the system. To address situations where the initial domain model has insufficient information to correctly model human feedback, we introduce an iterative approach in Chapter 4 to refine the system's state space in order to better discriminate human feedback, producing a more nuanced partitioning of the state-action space with different levels of competence, and allowing the system to better learn and act at its true competence.

One of the main characteristics of CAS is that the system must *recognize* the limits on its autonomy, but it is not required to *know the reasons* for these restrictions. This could be seen as a limitation, but we argue that it is an advantage because it allows us to build autonomous systems that respect constraints on autonomy derived from human knowledge that is beyond the scope of the system's reasoning abilities. While we allow for situations in which the system does not have complete knowledge of the risks that justify the limitations on its autonomy, the system may acquire that knowledge over time through interactions with the human.

Our contributions in this chapter are thus (1) a mathematically rigorous formalization of *competence* for automated decision making; and (2) a planning framework for a *competence-aware system* that integrates a model of competence with a planning model to enable the system to reduce unnecessary reliance on humans and optimize its autonomous behavior. We provide a theoretical analysis of our model and algorithm, a concrete example of a CAS and considerations in its design and implementation, and empirical evaluations of our model in simulation.

3.1 Competence-Aware Systems

We start with a description of a general *competence-aware system* that can operate in and plan for multiple *levels of autonomy*. Each level of autonomy is defined by a unique set of constraints on autonomous operation and consists of different forms of human feedback that can be provided to the autonomous agent. To enable the agent to reason about its own competence, we integrate information from three different models: a *domain model*, an *autonomy model*, and a *feedback model*. Throughout this section, we use the problem setting in Example 1 as a running example to better illustrate the concepts and terminology that we introduce throughout the chapter.

Example 1. An autonomous vehicle (AV) with a human driver (shown in blue in Figure 3.2) encounters an obstruction (e.g., a parked truck) blocking its lane on a one-lane road (red). To overtake the obstruction, the AV would need to drive around the obstruction necessarily driving through the oncoming traffic's lane. In the oncoming lane, there may or may not be a vehicle (yellow), but while stopped behind the obstruction, the AV cannot detect it. The AV may `Stop` to let oncoming traffic go past or see if the obstruction resolves itself (e.g., starts moving again), `Edge` into the oncoming lane to gain better visibility without risking crashing, or `Go` and begin passing the obstruction through the oncoming lane.

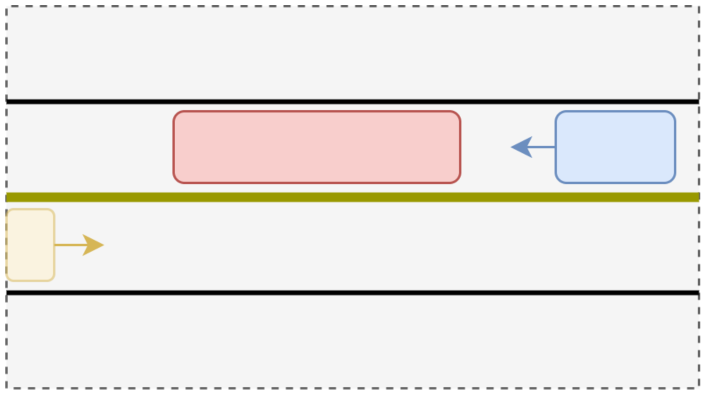


Figure 3.2: Illustration of Example 1.

3.1.1 Domain Model

The *domain model* describes the environment in which the agent operates and the dynamics of the agent's actions within that environment. We model this as a *stochastic shortest path* (SSP) problem, a commonly used form of *Markov decision process* (MDP) for reasoning in fully-observable, stochastic environments where the objective is to find the least-cost path from a start state to a goal state [14]. For the purposes of this chapter, we consider goal-oriented cost-minimizing problems as they align more naturally with the problem domains that are considered in our experiments. On the other hand, extending the theory to mixed-observable and partially-observable MDPs introduces additional sources of uncertainty, particularly with respect to human interaction, that are non-trivial to handle in our model. We discuss this challenge and present the partial-observability generalization in Section 3.4.

Definition 7. A **domain model**, $\mathcal{D}$, is an SSP represented by the tuple $\langle S, A, T, C, s_0, G\rangle$.

For a more complete overview of SSPs and their objective, see Chapter 2, Section 2.1.2.

3.1.2 Autonomy Model

The *autonomy model* describes the levels of autonomy that the agent can operate in, restrictions on the situations under which each level is allowed, and the (possibly negative) utility associated with operating in each level.

Definition 8. An **autonomy model**, $\mathcal{A}$, is represented by the tuple $\langle \mathcal{L}, \kappa, \mu \rangle$ where:

- $\mathcal{L}$ is the finite, partially ordered set of levels of autonomy where each level $l \in \mathcal{L}$ corresponds to some set of constraints on the system's autonomy,
- $\kappa : S \times \mathcal{L} \times A \rightarrow \mathcal{P}(\mathcal{L})$ is the autonomy profile where $\kappa(s, l, a)$ returns the subset of levels of autonomy $L \subseteq \mathcal{L}$ allowed when performing action $a \in A$ in state $s \in S$ given that the agent just acted in level $l \in \mathcal{L}$, and
- $\mu : S \times \mathcal{L} \times A \times \mathcal{L} \rightarrow \mathbb{R}^+$ is the cost of autonomy where $\mu(s, l, a, l')$ describes the cost of taking action $a \in A$ in level $l' \in \mathcal{L}$ in state $s \in S$ given that the agent just acted in level $l \in \mathcal{L}$.

While most interpretations of levels of autonomy are presented as ordered sets of increasing autonomy, in general this need not be the case. In fact, in some cases different levels of autonomy may be directly compared. Hence, we choose to model ours more generally as a partially ordered set[1] where $l_i \leq l_j$ if and only if, given any task (s_0, G), $V^{l_i}(s_0) \leq V^{l_j}(s_0)$ where V^{l_i} is the value function induced by the optimal policy when the level of autonomy is fixed at l_i. Note that we consider two levels, l_i and l_j, to be *adjacent* if $l_i < l_j \wedge \nexists l_k \in \mathcal{L} \,|\, l_i < l_k < l_j$. The constraints corresponding to each level of autonomy can be technical in nature, i.e., internally imposed constraints such as requiring human supervision in poor weather conditions that may be known *a priori* to cause errors, as well as externally imposed constraints such as ethical or

[1]$\mathcal{L}$ could be structured as a polytree or an arbitrary directed acyclic graph, however, for the sake of clarity we do not consider such levels of autonomy in this thesis.

	Levels of Autonomy	Human Involvement
l_0	No Autonomy	Human driver fully in control of vehicle.
l_1	Verified Autonomy	Autonomous agent in control of vehicle conditioned on explicit approval from human for maneuver prior to execution.
l_2	Supervised Autonomy	Autonomous agent in control of vehicle conditioned on a human driver supervising the system ready and capable of taking control.
l_3	Unsupervised Autonomy	Autonomous agent in unconditional control of vehicle, *possibly* with (but not requiring) a human who can take over control.

Table 3.1: Levels of autonomy with $\mathcal{L} = \{l_0, l_1, l_2, l_3\}$ where $l_0 \to l_1 \to l_2 \to l_3$.

	Constraints on Autonomy
Ethical	The AV may not be allowed to initiate a transfer of control to a human who is drowsy or otherwise deemed unfit to operate the vehicle safely.
Legal	The AV may not be allowed to operate autonomously inside of a school zone.
Technical	The AV may be disallowed from operating autonomously in snowy weather due to the interference of perception and object detection systems.
Tentative	The AV may be initialized to drive in l_1 when it has no visibility, but may learn to perform the action `Edge` in l_3 as it introduces an allowable level of risk by the human in the car.

Table 3.2: Examples of different types of constraints on autonomy.

legal requirements. Each constraint is associated with a corresponding form of human assistance or involvement. Intuitively, the higher the level of autonomy, the lower the cost of human involvement, although this is not a requirement of the model. An example of a set of levels of autonomy can be seen in Table 3.1.

Additionally, κ can be defined to not only reflect hard constraints such as ethical, legal, or technical constraints [66, 72, 47, 114] that are fixed throughout the system's deployment, but also tentative constraints that can be updated over time. Tentative constraints allow for a period of learning or adjustment early in the deployment of the system as the human familiarizes themselves with the system, or the system learns to act appropriately in its environment. An example of different constraints on autonomy can be seen in Table 3.2.

The cost of autonomy, μ, is the cost associated with the act of operating in a given level of autonomy and is distinct from the base domain cost of the action's execution. For example, in a level of autonomy that requires tele-operation from an off-site human to provide verification to a waiting autonomous vehicle, there may be an additional cost of operating in that level corresponding to the amount of time waiting to reach an available tele-operator and receive feedback. In a system with a finite energy supply that can perform sensing and perception at different levels of fidelity (corresponding to different levels of autonomy), each level may utilize a different amount of energy.

3.1.3 Feedback Model

The *feedback model* describes the agent's knowledge about and predictions of its interactions with the human, including the types of feedback it can receive from the human, how likely each possible type of feedback is at any given time, and the expected cost to the human for assisting the agent. In this chapter, we only consider the case where feedback is provided *reactively* by the human, i.e., provided in response to the current action being executed by the agent, and generated in the course of the action's execution. In Chapter 6 we relax this assumption and consider additionally feedback that might be provided *proactively* by the human in response to inferred future behavior.

Definition 9. A **feedback model**, $\mathcal{F}$, is represented by the tuple $\langle \Sigma, \lambda, \rho, \tau_{\mathcal{H}} \rangle$, where:

- Σ is the finite set of feedback signals that the agent can receive from the human,
- $\lambda : S \times \mathcal{L} \times A \times \mathcal{L} \to \Delta^{|\Sigma|}$ is the feedback profile where $\lambda(s, l, a, l')$ represents the probability distribution over feedback signals that the agent will receive when performing action $a \in A$ in level $l' \in \mathcal{L}$ in state $s \in S$ given that the agent just operated in level $l \in \mathcal{L}$,

	Feedback Signal	Interaction	Levels of Autonomy
$\emptyset$	No feedback	N/A	$\{l_0, l_2, l_3\}$
$\oplus$	Approval	Verbal or Tactile Response	$\{l_1\}$
$\ominus$	Disapproval	Verbal or Tactile Response	$\{l_1\}$
$\oslash$	Override	Arrested Control	$\{l_2, l_3\}$

Table 3.3: Each feedback signal is provided via a fixed and known interaction; for instance, the feedback signal *approval* may be provided either by a verbal "Yes" from the human, or via a tactile response such as pressing a button on a touchscreen, similarly for *disapproval. Override* may be recognized by any form of arrested control by the human during autonomous operation, for instance braking, accelerating, or steering while the AV is in control. Each signal is only recognized when the AV is operating at the corresponding level of autonomy.

- $\rho : S \times \mathcal{L} \times A \times \mathcal{L} \to \mathbb{R}^+$ is the human cost function where $\rho(s, l, a, l')$ represents the cost to the human when the agent performs action $a \in A$ in level $l' \in \mathcal{L}$ in state $s \in S$ given that the agent just operated in level $l \in \mathcal{L}$, and
- $\tau_{\mathcal{H}} : S \times A \times S \to [0, 1]$ is the human state transition function where $\tau_{\mathcal{H}}(s, a, s')$ represents the probability of transitioning to successors states $s' \in S$ when the human takes control of the system when the agent attempts to perform action $a \in A$ in state $s \in S$.

Although there are many forms of human feedback that have been studied, we limit our focus specifically to *feedback signals*, which are represented as discrete tokens of feedback that the human can provide to the autonomous agent, either implicitly (e.g., facial gestures or body posture), or explicitly (e.g., verbal responses or physical control), as opposed to real-valued reward signals [59, 58] or full demonstrations [30, 88, 90]. The primary reason is to keep the feedback signals semantically simple in the sense that they are represented compactly by the system while still being easily and unambiguously associated with the human's intentions. This reduces the overhead associated with the human-agent interactions. Each feedback signal may be associated with a distinct level, or subset of levels, of autonomy and a corresponding form of human involvement. An example of such feedback signals and how they may function is described Table 3.3, and an illustration of the manner in which feedback is provided during execution can be seen in Figure 3.3. Future directions of research may investigate extending these feedback signals to address such questions as how to learn from feedback when there is a *degree* of severity associated with it, how to handle *proactive feedback*, which is intended by the human to be for inferred future states or trajectories, or feedback in the form of direct action commands.

The human cost function, ρ, is the cost *to the human* when operating in a given level and hence is separate from the costs incurred directly by the autonomous agent. This cost could measure the human's opportunity cost for being unable to engage in other activities while assisting the autonomous agent. However, it may additionally capture other costs to the human, such as additional stress or work added to them in addition to the time they spend assisting. That is, assisting two different actions, which take the same time may require different levels of exertion from the human, for example supervising an autonomous action making a left turn, or manually making the left turn. In practice, the human's cost function may be non-Markovian; for instance becoming fatigued after repeatedly performing manual control, or becoming frustrated after extended periods of oscillating between different levels of autonomy, constantly shifting the demand on the human. While this can be coarsely approximated by conditioning the cost on the previous level of autonomy (as done here), one can improve this by maintaining a model of the human's state, similar to what is done by Costen et al. [32].

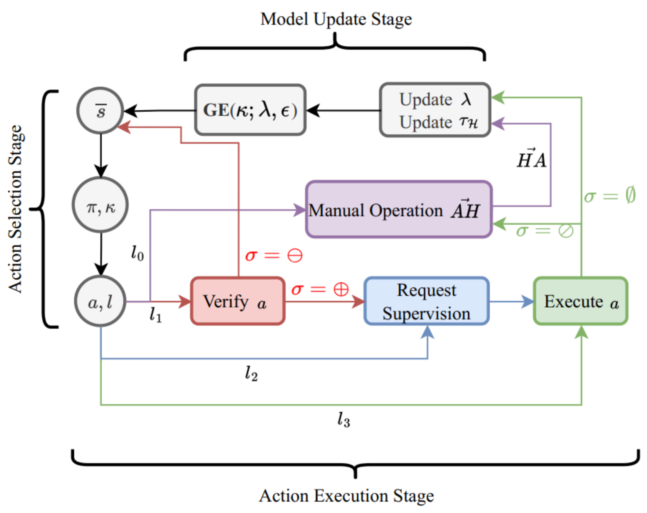
If λ and $\tau_{\mathcal{H}}$ are known exactly *a priori* then the system's true competence (Definition 16) can be immediately computed exactly under any κ, and the problem reduces to a straightforward planning problem. Furthermore, in some problem instances where the feedback model is known exactly there may be no need to even constrain the policy space at all (i.e., $\kappa(s, a) = \mathcal{L}$ for every $(s, a) \in S \times A$). This is the case when the feedback mechanisms are sufficient to prevent the agent from taking actions that would violate hard constraint; for example, if the human authority always overrides an action at a level that would violate an ethical, legal, or technical constraint. This introduces a trade-off in distributing the burden of effort between the designers and operator(s) of the system to ensure safe and reliable operation in all cases.

However, in this work we are primarily concerned with systems where λ and $\tau_{\mathcal{H}}$, and by consequence the system's true competence, are unknown *a priori*. In this case, they must be estimated by functions $\hat{\lambda}$ and $\hat{\tau}_{\mathcal{H}}$, which are based on observed data collected online through interactions with the human at various levels of autonomy that can generate feedback signals. These feedback signals can be analogously treated as labels in a labeled data set where the data is the state, action, and level that generated the feedback signal. In Chapter 4, we address situations where the human's model of the world does not align with that of the autonomous agent, leading to feedback that is poorly discriminated by the agent, which reduces its ability to learn from the signals it receives from the human.

Note that, in many real-world problems, the process of acquiring feedback signals may not be instantaneous, and in some cases could require a complex process of fully or partially transferring control to and from a human over an indefinite amount of time, where each element of the transfer process, such as the communication interface, is important. The problem of transfer of control in semi-autonomous systems has been separately studied [103, 124]; however, for the sake of clarity, we do not model this process explicitly in this work as we focus on the orthogonal problem of modeling levels of autonomy and competence.

3.1.4 Competence-Aware Systems

A *competence-aware system* (CAS) represents a planning problem that accounts for the different levels of autonomy available to the agent and factors in the agent's



l_0	No Autonomy
l_1	Verified Autonomy
l_2	Supervised Autonomy
l_3	Unsupervised Autonomy
$\oplus$	Approval
$\ominus$	Disapproval
$\oslash$	Override
$\emptyset$	No Feedback
κ	Autonomy Profile
λ	Feedback Profile
$\tau_{\mathcal{H}}$	Human Transition Function
GE	Gated Exploration
$\overrightarrow{AH}$	Agent-to-Human Transfer of Control
$\overrightarrow{HA}$	Human-to-Agent Transfer of Control

Figure 3.3: Illustration of Example 2. During the action selection stage, the policy π, constrained by autonomy profile κ, is queried for state $\bar{s}$, and returns an action a to be performed at a level of autonomy l. During the action execution stage, the level of autonomy dictates the manner in which the agent executes the action, and the form of human assistance involved. Additionally, human feedback signals can be provided depending on the level of autonomy at different stages of the action's execution, including prior to, which is recorded by the CAS and used to update its internal models during the model update stage.

expectations regarding the likelihood and cost of human feedback (e.g., assistance, queries, intervention, etc.). The objective of a solution to a CAS planning problem is to create a plan that best balances the cost of reaching the goal with the cost of human assistance to achieve the most cost-effective strategy given the constraints of the problem. Hence, the CAS uses the autonomy model to proactively generate plans that operate across multiple levels of autonomy by leveraging the feedback model to predict the likelihood of different feedback signals to optimize the level of autonomy and minimize the reliance on humans. To this end, we represent a CAS as a *multi-objective* planning problem.

Example 2. A competence-aware system with four levels of autonomy—verified, supervised, unsupervised, and no autonomy—and four type of feedback signals—approval, disapproval, override, and no feedback. The policy, π, constrained by the autonomy profile κ, produces an action a at a level l to be performed in state $\bar{s}$.

The level l determines the execution process of the action a, as depicted in the lower section of the figure. Certain levels may prompt the human for feedback, with a possibility of complete transfer of control from the autonomous agent to the human. After the action is executed and data is collected, internal model parameters, λ and $\tau_{\mathcal{H}}$, are updated. Finally, the agent may perform gated exploration (Definition 14) to update the autonomy profile κ, although in practice this would be performed on a less frequent basis.

Definition 10. A **competence-aware system** $\mathcal{S}$ is a multi-objective planning model represented by the tuple $\langle \overline{S}, \overline{A}, \overline{T}, \overline{C}, \overline{s}_0, \overline{G} \rangle$, where:

- $\overline{S} = S \times \mathcal{L}$ is a set of factored states, each comprised of a domain state $s \in S$ and a level of autonomy $l \in \mathcal{L}$.
- $\overline{A} = A \times \mathcal{L}$ is a set of factored actions, each comprised of a domain action $a \in A$ and a level of autonomy $l \in \mathcal{L}$.
- $\overline{T} : \overline{S} \times \overline{A} \times \overline{S} \rightarrow [0, 1]$ is a transition function where $\overline{T}(\overline{s}, \overline{a}, \overline{s}')$ represents the probability of transitioning to successor states $\overline{s}' \in \overline{S}$ when taking action $\overline{a} \in \overline{A}$ in state $\overline{s} \in \overline{S}$.
- $\overline{C} = \begin{bmatrix} C & \mu & \rho \end{bmatrix}$ is a vector of cost functions.
- $\overline{s}_0 \in \overline{S}$ is the initial state where $\overline{s}_0 = \langle s_0, l \rangle$ for some $l \in \mathcal{L}$.
- $\overline{G} \subset \overline{S}$ is the set of goal states.

A CAS state $\overline{s} \in \overline{S}$ represents the current domain state s and the level of autonomy, l, that the CAS performed its last action in. The purpose of including the previous level of autonomy in the state representation is to capture the fact that human feedback can vary depending on the level of autonomy that the agent was just operating in (for instance, a human may be less likely to override the system if they were previously engaged in supervising the system); additionally, we may want to discourage the system from oscillating between levels of autonomy by imposing a small cost every time the system changes levels. Note that, one can set $\overline{G} = \hat{S} \times \mathcal{L}$ for some $\hat{S} \subseteq S$ to indicate that the level of autonomy does not impact the goal condition or state, for instance setting $\overline{G} = G \times \mathcal{L}$.

A CAS action $\overline{a} \in \overline{A}$ represents a domain action a to be performed at a given level of autonomy l, which may alter both the mechanics of how the action is executed, the form and degree of involvement by the human authority in the execution of the action, and the types of feedback that the agent can receive from the human authority.

$\overline{T}$ is a transition function that represents the probability distribution over both how the state will change and which feedback signal, if any, the agent will receive from the human when performing an action conditioned on the level the action is being performed in, the current state, and the previous level that the agent had operated in (i.e., the timestep prior to the current one). For example, the likelihood of a human override may decrease if the system had already been acting under supervision than if they had been acting without supervision, as the human may have a better understanding of what the system is doing.

Example 3. Given $\mathcal{L}$ and Σ, we can specify the **state transition function** of this CAS. Given $\overline{s} = (s, l)$, $\overline{s'} = (s', l')$, and $\overline{a} = (a, l')$, we define $\overline{T}$ as follows:

$$\overline{T}(\overline{s}, \overline{a}, \overline{s}') = \begin{cases} \tau_{\mathcal{H}}(s, a, s'), & \text{if } l = l_0, \\ \lambda(\oplus|\overline{s}, \overline{a})\overline{T}(\overline{s}, (a, l_2), \overline{s'}) + \lambda(\ominus|\overline{s}, \overline{a})[s = s'], & \text{if } l = l_1, \\ \lambda(\emptyset|\overline{s}, \overline{a})T(s, a, s') + \lambda(\oslash|\overline{s}, \overline{a})\tau_{\mathcal{H}}(s, a, s'), & \text{if } l \in \{l_2, l_3\}, \end{cases} \tag{3.1}$$

where $[\cdot]$ denotes Iverson brackets. Intuitively, Equation 7 states that when the agent operates in l_0, it follows the transition dynamics of the human who takes control. When operating in l_1, the probability it arrives in state s' is the probability it is approved to take the action times the probability of the state change following $\overline{T}$ under level l_2, plus the probability that it is disapproved and the state is the same. In levels l_2 and l_3, the probability it arrives in state s' is the probability it succeeds following T without any human intervention plus the probability that the human overrides it and takes it to that state. In general, we expect the probability of an override to be lower (or even 0) in l_3 as supervision is not required.

A solution to a given CAS is a policy π that maps states and levels $\overline{s} \in \overline{S}$ to actions and levels $\overline{a} \in \overline{A}$. Multi-objective decision making has been well-studied [92], and for our purposes we assume a scalarized approach [92] with a scalarization function f parameterized by a weight vector $\mathbf{w}$. In particular, we consider a linear scalarization as defined in Section 2.2.1 when solving for the optimal policy; however, in some cases a linearization may be inappropriate, and in fact a well-calibrated scalarization function may be necessary to accurately reflect the relative impact of each cost function in any circumstance. With some modifications, the CAS model could also be extended to handle both lexicographic models [123] and constrained models [2]. However, the properties that we derive for the scalarized model may not necessarily hold for arbitrary multi-objective models, and would need to be re-examined in those contexts.

Additionally, we restrict the CAS to only consider policies that are allowed under the autonomy profile κ in the following way.

Definition 11. Let $\overline{a} = \langle a, l \rangle$. Given $\overline{s} = \langle s, l' \rangle \in \overline{S}$, we say that $(\overline{s}, \overline{a})$ is *allowed* if $l \in \kappa(s, a)$, and a policy π is *allowed* if for every $\overline{s} \in \overline{S}$, $(\overline{s}, \pi(\overline{s}))$ is allowed.

We denote the set of allowable policies given κ as Π_κ and require that the policy returned by solving the CAS, π^*, is always taken from $\text{argmin}_{\pi \in \Pi_\kappa} V^\pi(s_0)$. An illustration of how different autonomy profiles can constrain the full policy space, Π, can be seen in Figure 3.4.

In general, a competence-aware system planning model is not guaranteed to be a valid stochastic shortest path problem (see Proposition 1) due to the possible effects

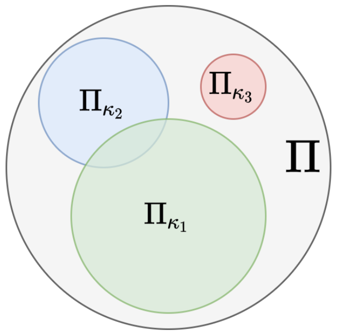


Figure 3.4: Illustration of a policy space, Π, constrained by three different autonomy profiles, κ_1, κ_2, and κ_3.

that κ and λ can have on the existence of a proper policy, although in some cases they may only induce dead-ends away from the initial state for which there is existing work on how to handle [60]. However, one can ensure that there is a proper policy with the inclusion of a level of autonomy with a property similar to level l_0 in Table 3.1 that allows for (at potentially high cost) the deterministic completion of any action or task, guaranteeing the existence of a proper policy. Note that we do not need to worry about the possibility of ρ or μ inducing zero-cost cycles as they are non-negative cost functions, and the domain model is, by assumption, a valid SSP.

3.2 Properties of a Competence-Aware System

In this section, we will discuss the central properties of a CAS that will allow us to prove several key results of competence-aware systems. Henceforth, we will assume that there exists a singular human authority that the semi-autonomous system in a CAS interacts with, and we will use the notation $\mathcal{H}$ to refer to them.

Definition 12. The *human authority*, $\mathcal{H}$ is represented by the tuple $\langle F^{\mathcal{H}}, \lambda^{\mathcal{H}}, \kappa^{H} \rangle$ where:

- $F^{\mathcal{H}}$ is the set of features used by $\mathcal{H}$ when providing feedback,
- $\lambda^{\mathcal{H}} : \overline{S} \times \overline{A} \rightarrow \Delta^{|\Sigma|}$ is a stationary distribution of feedback signals that $\mathcal{H}$ follows, and

- $\kappa^{\mathcal{H}} : \overline{S} \times A \rightarrow \mathcal{P}(\mathcal{L})$ is the fixed mapping from state-action pairs to sets of autonomy levels that $\mathcal{H}$ will allow the autonomous agent to operate in with nonzero probability.

Intuitively, $\kappa^{\mathcal{H}}$ represents the human authority's belief of the agent's competence; by definition any level not contained in the image of $\kappa^{\mathcal{H}}$ will never be allowed by $\mathcal{H}$.

First, we begin with a simple proof that a CAS model is, in general, not guaranteed to be a valid stochastic shortest path problem due to the lack of a proper policy.

Proposition 1. There exists a competence-aware system $\mathcal{S}$ that does not admit a proper policy.

Proof. Let $\mathcal{S}$ be a CAS with exactly one level of autonomy, l, where the level of autonomy works as follows: when the agent attempts to execute action a, they must first query the human to obtain a binary `yes` or `no` feedback signal. If the signal is `yes` then the agent may attempt to execute the action according to its model. If the signal is `no` then the agent may not attempt to execute the action in its current state. Let $(s_0, l) \in \overline{S}$ denote the initial state and assume $(s_0, l) \notin \overline{G}$, where $\overline{S}$ is the state space of $\mathcal{S}$ and $\overline{G}$ is the set of goals. Let $\lambda^{\mathcal{H}}(\texttt{yes}|(s_0, l), (a, l)) = 0.0$ for every action $a \in A$ (where A is the action set). As the agent will never be able to transition out of its state, which is not a goal state by assumption, it is clear that there exists no proper policy. $\square$

Second, a fundamental component of the CAS model is the ability to adjust its autonomy profile over time using what it has learned to optimize its autonomy by reducing unnecessary reliance on human assistance. However, before operating in a new level of autonomy, the system may have no knowledge of how the human will interact with it in that level, i.e., the feedback profile in that new level may be initialized by default to some baseline distribution. As a result it is necessary that the system *explore* levels of autonomy that it predicts are more cost effective than its current allowed levels, so that it may learn whether or not it is competent to act in those levels.

Allowing the system to alter its own autonomy profile, however, can lead to severe consequences in the real world if not done carefully, mitigating the risk-awareness we aim to endow via the competence modeling. Therefore, we propose two notions to ensure a measure of safety and risk-sensitivity in a competence-aware system. The first is *level-safety*, which is a notion of the safety of the level of autonomy that the system is using and is conditioned on both the agent and the human; intuitively, a CAS is level-safe if it cannot act in levels that the human authority would not allow. Second is *gated exploration*, which is a simple extension to standard exploration methods used in reinforcement learning in which the system must obtain permission from a human before exploring a new (disallowed) level of autonomy, ensuring that level-safety is never violated.

Example 4. An autonomous vehicle is initialized to only use levels $\{l_0, l_1, l_2\}$ when executing the overtaking maneuver, but learns that there is a very low likelihood of an override by the human authority during the day with clear visibility and sparse traffic. Hence, it expects based on estimated costs that its competence is in fact l_3, which is initially disallowed to ensure safety at initial deployment. It therefore queries the human to approve it to update its autonomy profile κ by adding level l_3 under the stated conditions.

Definition 13. A CAS $\mathcal{S}$ is *level-safe* under κ if $\kappa(\overline{s}, a) \subseteq \kappa^{\mathcal{H}}(\overline{s}, a)$ for every $(\overline{s}, a) \in \overline{S} \times A$.

Definition 14. We define the *gated-exploration* strategy for $(\overline{s}, a) \in \overline{S} \times A$ as follows: let $\text{adj}(l, l') = 1$ if $l = l'$ or l and l' are adjacent in $\mathcal{L}$ and 0 otherwise, and let $\text{adj}(\kappa(\overline{s}, a), l') = 1$ if $l' \in \kappa(\overline{s}, a)$ or $\text{adj}(l, l') == 1$ for some $l \in \kappa(\overline{s}, a)$. Let $P_l(\mathcal{L})$ be a distribution over $\mathcal{L}$ such that $P_l(l') = 0$ if $adj(l, l') == 0$, and let $l^* \sim P_l(\mathcal{L})$. If

$l^* \in \kappa(\overline{s}, a)$ do nothing, otherwise, query the human authority $\mathcal{H}$ to allow for the level exploration. If the query returns a positive response, set $\kappa(\overline{s}, a) \leftarrow \kappa(\overline{s}, a) \cup \{l^*\}$, and otherwise do nothing.

Proposition 2. Let $\mathcal{S}$ be a CAS with initial autonomy profile κ_0. If $\mathcal{S}$ is level-safe under κ_0 and follows the gated-exploration strategy, then $\mathcal{S}$ will be level-safe under κ_t for any $t \geq 0$.

Proof. This is straightforward to observe by applications of the definitions. If $\mathcal{S}$ is level-safe under κ_0, then for all $(\overline{s}, a) \in \overline{S} \times A$, $\kappa_0(\overline{s}, a) \subseteq \kappa^{\mathcal{H}}(\overline{s}, a)$ by definition. If there exists $t > 0$ for which $\kappa_t(\overline{s}, a) \neq \kappa_0(\overline{s}, a)$ for some $(\overline{s}, a) \in \overline{S} \times A$, then there is some $l^* \in \kappa_t(\overline{s}, a) \setminus \kappa_0(\overline{s}, a)$. By the definition of gated exploration and $\kappa^{\mathcal{H}}$, it must be that $l^* \in \kappa^{\mathcal{H}}(\overline{s}, a)$, and hence $\kappa_t(\overline{s}, a) \subseteq \kappa^{\mathcal{H}}(\overline{s}, a)$. As $(\overline{s}, a)$ is arbitrary, this holds for all $(\overline{s}, a) \in \overline{S} \times A$, and hence $\mathcal{S}$ is level-safe. $\square$

Next, we introduce a notion of *feedback consistency*, which is a property of how consistent the human authority is in providing the same feedback given the same query by the acting agent.

Definition 15. Let $F^{\mathcal{H}} = \{F_1^{\mathcal{H}}, ..., F_n^{\mathcal{H}}\}$ be the set of features used by the human authority, $\mathcal{H}$, and let $\overline{S}_{\mathcal{H}} = F_1^{\mathcal{H}} \times \cdots \times F_n^{\mathcal{H}} \times \mathcal{L}$. The **ground truth feedback function** is a deterministic mapping $f : \overline{S}_{\mathcal{H}} \times \overline{A} \to \Sigma$. $\mathcal{H}$ is **perfectly consistent** if $\lambda^{\mathcal{H}}(f(\overline{s}, \overline{a})|\overline{s}, \overline{a}) = 1 \; \forall \overline{s} \in \overline{S}, \overline{a} \in \overline{A}$. If $\lambda^{\mathcal{H}}(f(\overline{s}, \overline{a})|\overline{s}, \overline{a}) \geq \epsilon$ for $\epsilon \in (0, 1) \; \forall \overline{s} \in \overline{S}, \overline{a} \in \overline{A}$, then $\mathcal{H}$ is ϵ-**consistent**.

Unless otherwise stated, we assume that the human authority is ϵ–consistent henceforth. We now define three central properties of a CAS.

Definition 16. Let $\lambda^{\mathcal{H}}$ be the stationary distribution of feedback signals that the human authority follows. The **competence** of CAS $\mathcal{S}$, denoted $\chi_{\mathcal{S}}$, is a mapping from $\overline{S} \times A$ to the optimal (least-cost) level of autonomy given perfect knowledge of $\lambda^{\mathcal{H}}$. Formally:

$$\chi_{\mathcal{S}}(\overline{s}, a) = \operatorname*{argmin}_{l \in \mathcal{L}} q^*(\overline{s}, (a, l); \lambda^{\mathcal{H}}) \tag{3.2}$$

where $q^*(\overline{s}, (a, l); \lambda^{\mathcal{H}})$ is the cumulative expected cost under the optimal policy π^* when taking action $\overline{a} = (a, l)$ in state $\overline{s}$ conditioned on the human authority's feedback distribution, $\lambda^{\mathcal{H}}$.

Fundamentally, the system's competence for executing action a in state $\overline{s}$, $\chi_{\mathcal{S}}(\overline{s}, a)$, is the most beneficial (e.g., cost effective) level of autonomy were it to know the true human feedback distribution. When $\mathcal{L}$ is an ordered set, we expect this to generally be the highest level of autonomy *allowed* by the human; however, this need not be the case. In principle, the highest allowed level of autonomy could require more frequent human interventions, e.g., due to lower levels of trust by the human in the system [52], that may render it less efficient overall relative to a lower level of autonomy.

It is important to emphasize that this definition of competence relies on $\lambda^{\mathcal{H}}$—the human's mental model of the agent's capabilities—and hence is a definition of competence on *the overall human-agent system*, and is explicitly not just a measure of the underlying agent's technical capabilities (i.e., $\mathcal{D}$). A corollary of this fact is that the CAS is as competent as the human authority believes, and allows, it to be. A human authority that has a poor understanding of the system's abilities could lead to the system having a different competence than a human authority that knows perfectly the limitations and capabilities of the system, by either under-utilizing or over-utilizing the system in an autonomous capacity. In this work, we treat the human as an expert with a good understanding of the system's capabilities, but future work will look to extend the CAS to the setting where the human authority may have little, or even no, initial technical understanding of the system and its abilities. One reason for modeling *competence* in this manner is to avoid relying on arbitrary thresholding based on evaluative metrics to determine when a system is competent or not.

We say that a CAS $\mathcal{S}$ is λ-stationary if, in expectation, any new feedback drawn from the true distribution $\lambda^{\mathcal{H}}$ will not affect λ enough to change the optimal level of autonomy for any $\overline{s} \in \overline{S}$ and $a \in A$. We show below that, under standard assumptions, $\mathcal{S}$ will converge to λ-stationarity.

Definition 17. Let $\mathcal{S}$ be a CAS and let $U(\lambda)$ be the q-value of $(\overline{s}, a)$ under the optimal policy given λ where $\mathcal{S}$ executed the action a in level l in state $\overline{s}$. We define the expected value of sample information (EVSI) on $\sigma \in \Sigma$ for $(\overline{s}, a)$ to be:

$$\sum_{\sigma \in \Sigma} \max_{l \in L} \int_{\Lambda} U(l, \lambda)\lambda(\sigma|\overline{s}, a, l)p(\lambda)d\lambda - \max_{l \in L} \int_{\Lambda} U(l, \lambda)p(\lambda)d\lambda. \tag{3.3}$$

Definition 18. Let $\mathcal{S}$ be a CAS. $\mathcal{S}$ is **λ-stationary** if for every state $\overline{s} = (s, l) \in \overline{S}$, and every action $a \in A$, the expected value of sample information on $\sigma \in \Sigma$ for $(\overline{s}, a)$ (Eq. 3.3) is less than ϵ for any ϵ greater than 0.

Proposition 3. Let $\lambda_t^{\overline{s},a}$ be the random variable representing $\lambda(\overline{s}, a)$ after having received t feedback signals for $(\overline{s}, a)$ where each signal is sampled from the true distribution $\lambda^{\mathcal{H}}(\overline{s}, a)$. Then, as $t \to \infty$, the sequence $\{\lambda_t^{\overline{s},a}\}$ converges in distribution to $\lambda_{\mathcal{H}}^{\overline{s},a} = \mathbb{E}[\lambda^{\mathcal{H}}(\overline{s}, a)]$.

Proof. As each signal is drawn from $\lambda^{\mathcal{H}}(\overline{s}, a)$ i.i.d, then by a straightforward application of the law of large numbers the sequence will converge in probability to $\lambda_{\mathcal{H}}^{\overline{s},a}$, which directly implies the claim. $\square$

Theorem 1. Let $\mathcal{S}$ be a CAS, and let $\lambda_t^{\overline{s},a}$ be the random variable representing $\lambda(\overline{s}, a)$ after having received t feedback signals for $(\overline{s}, a)$ where each signal is sampled from the true distribution $\lambda^{\mathcal{H}}(\overline{s}, a)$. As $t \to \infty$, if no $(\overline{s}, a)$ is starved, $\mathcal{S}$ will converge to λ-stationarity.

Proof. Let $\overline{s} \in \overline{S}$ and $a \in A$. As $\overline{s}$ and a are arbitrary and we assume that no $(\overline{s}, a)$ is starved, it is sufficient to show convergence to stationarity for $(\overline{s}, a)$ as $t \to \infty$.

By Proposition 3, $\{\lambda_t^{\overline{s},a}\}$ will converge to $\lambda_{\mathcal{H}}^{\overline{s},a}$ in distribution given our assumptions. Because $\{\lambda_t^{\overline{s},a}\}$ converges in distribution, $\lim_{t\to\infty} Pr(|\lambda_t^{\overline{s},a} - \lambda_{\mathcal{H}}^{\overline{s},a}| > \epsilon) = 0 \ \forall \epsilon > 0$. Therefore, in the limit the probability that $\lambda = \lambda_{\mathcal{H}}^{\overline{s},a}$ after t steps, $p_t(\lambda)$, defines a Dirac delta function with point mass centered at $\lambda^{\mathcal{H}}$. Hence we get that, $\lim_{t\to\infty}$ EVSI (Eq. 3.3)

$$
\begin{aligned}
&= \Big(\lim_{t\to\infty} \sum_{\sigma\in\Sigma} \max_{l\in L} \int_\Lambda U(\lambda,l)\lambda(\sigma|s,\emptyset,a,l)p_t(\lambda)d\lambda\Big) - \Big(\lim_{t\to\infty} \max_{l\in L} \int_\Lambda U(\lambda,l)p_t(\lambda)d\lambda\Big) \\
&= \Big(\sum_{\sigma\in\sigma} \max_{l\in L} U(\lambda^{\mathcal{H}},l)\lambda^{\mathcal{H}}(\sigma|s,\emptyset,a,l)\Big) - \Big(\max_{l\in L} U(\lambda^{\mathcal{H}},l)\Big) \\
&= \sum_{\sigma\in\Sigma} \max_{l\in L} U(\lambda^{\mathcal{H}},l)(1-\lambda^{\mathcal{H}}(\sigma|s,\emptyset,a,l)) \\
&= \max_{l\in L} U(\lambda^{\mathcal{H}},l)\Big(1-\sum_{\sigma\in\Sigma}\lambda^{\mathcal{H}}(\sigma|s,\emptyset,a,l)\Big) \\
&= \max_{l\in L} U(\lambda^{\mathcal{H}},l)(1-1) \\
&= 0.
\end{aligned}
$$

□

Second, we say that a CAS $\mathcal{S}$ is *level-optimal* in some state if, under its current optimal policy, the action it takes in that state is performed at its competence for that state-action pair.

Definition 19. Let $\mathcal{S}$ be a CAS. $\mathcal{S}$ is **level-optimal in state $\overline{s}$** if

$$\pi^*(\overline{s}) = (a, \chi_{\mathcal{S}}(\overline{s},a)) \tag{3.4}$$

If this holds for all states we say that $\mathcal{S}$ is **level-optimal**. Similarly, $\mathcal{S}$ is **γ-level-optimal** if this holds in $\gamma|\overline{S}|$ states for $\gamma \in (0,1)$.

The primary goal of a competence-aware system is to *reach level-optimality while maintaining level-safety.* As we have already shown that a CAS will maintain level-safety under the gated-exploration strategy (given an initial, level-safe autonomy profile), we therefore want to show that under certain conditions, a competence-aware system $\mathcal{S}$ will be guaranteed to reach level-optimality. In other words, that the system is guaranteed to reach a point where it operates at its competence in all situations.

To prove that a competence-aware system will reach level-optimality, we rely on the notion of *gated exploration* as detailed in Definition 14. However, we also require the following *exploitation* approach: if $\mathcal{S}$ has reached λ-stationarity then it no longer explores under the exploration strategy and instead exploits its knowledge by deterministically selecting the optimal level of autonomy at that point, i.e., for any given $(\overline{s}, a) \in \overline{S} \times A$, the system will use a level $l \in \operatorname{argmin}_{l \in \kappa(\overline{s},a)} q(\overline{s}, (a, l); \hat{\lambda})$. However, as the theory only proves convergence to λ-stationarity (that is, an expected value of sample information of 0 over all $\sigma \in \Sigma$ for every $(\overline{s}, a) \in \overline{S} \times A$) in the *limit*, we instead simply require that for any fixed $z \in \mathbb{R}^+$, sufficiently small, the system will switch to exploitation once the expected value of sample information falls below z everywhere, which will happen in finite time. We will refer to this below as *exploitation under stationarity.*

Definition 20. Let $\mathcal{S}$ be a CAS, and let κ_t represent the autonomy profile κ at time t. Given $\overline{s} \in \overline{S}$ and $a \in A$, we say that $l \in \mathcal{L}$ is reachable from κ_t for $(\overline{s}, a)$ if there exists at least one path from $\kappa_t(\overline{s}, a)$ to $l \in \mathcal{L}$, where all levels along the path are in $\kappa^{\mathcal{H}}(\overline{s}, a)$.

In the following text, let κ_t refer to the autonomy profile, κ, after the t^{th} feedback signal has been received.

Theorem 2. Let $\mathcal{S}$ be a CAS that follows the gated exploration strategy and performs exploitation under stationarity, where $\chi_{\mathcal{S}}(\overline{s}, a)$ is reachable from κ_0 for all $(\overline{s}, a) \in \overline{S} \times A$. Then if no $(\overline{s}, a)$ is starved, as $t \to \infty$, $\mathcal{S}$ will converge to level-optimality.

Proof. Fix $\overline{s} \in \overline{S}$ and threshold $z \ll 1 \in R^+$. We need to show that in the limit, $\pi^*(\overline{s}) = (a, \chi_{\mathcal{S}}(\overline{s}, a))$. By Proposition 1, $\mathcal{S}$ will converge to λ-stationarity for $(\overline{s}, a)$ for all $a \in A$. Hence there is a finite point t at which the expected value of information on Σ falls below z for $(\overline{s}, a)$ for every $a \in A$ and $\mathcal{S}$ will *exploit under stationarity* for $\overline{s}$. That is, at such time, $\pi^*(\overline{s}) = (a, \text{argmin}_{l \in \kappa_t(\overline{s}, a)}(q^*(\overline{s}, (a, l))$. By Proposition 3, this value is exactly the definition of $\chi_{\mathcal{S}}(\overline{s}, a)$ provided that $\chi_{\mathcal{S}}(\overline{s}, a) \in \kappa_t(\overline{s}, a)$. By assumption, $\chi_{\mathcal{S}}(\overline{s}, a)$ is reachable from $\kappa_0(\overline{s}, a) \subseteq \kappa^{\mathcal{H}}(\overline{s}, a)$, so given that under the gated exploration strategy, there is a nonzero probability of reaching $\chi_{\mathcal{S}}(\overline{s}, a)$, and as $\overline{s}$ is arbitrary, we are done. □

3.3 Evaluation

To test the competence-aware system, we implemented the CAS model in two simulated autonomous vehicle domains at different levels of decision-making abstraction. The first domain is a high-level navigation problem in which an autonomous vehicle must plan (and execute) the optimal route to take between two locations conditioned on its knowledge about different intersections and streets and its own competence in performing different maneuvers at the various locations. The second takes a more fine-grained look at one of the maneuvers that can be performed in the first domain, namely passing an obstacle that is blocking its lane, and is modeled after the domain depicted in Example 1.

Gated Exploration

In all experiments, we used the gated exploration strategy as defined in Definition 14. While a variety of different distributions could be used for the exploration strategy, we use an extension of the standard Boltzmann softmax distribution [57] over q-values in the adjacency set of $l \in \mathcal{L}$:

$$P(l') = adj(\kappa(\overline{s}, a), l') \frac{\exp(-q(\overline{s}, (a, l'); \hat{\lambda}))}{\sum_{l'' \in \mathcal{L}} adj(\kappa(\overline{s}, a), l'') \exp(-q(\overline{s}, (a, l''); \hat{\lambda}))} \tag{3.5}$$

where $q(\overline{s}, (a, l); \hat{\lambda}) = \overline{C}(\overline{s}, (a, l)) + \sum_{\overline{s}' \in \overline{S}} \overline{T}(\overline{s}, (a, l), \overline{s}') V(\overline{s}'; \hat{\lambda})$ is the expected cumulative reward when taking action $(a, l) \in \overline{A}$ in state $\overline{s} \in \overline{S}$ conditioned on the current feedback profile $\hat{\lambda}$.

To improve exploration efficiency, we introduce a potential-based mechanism in our experiments in which, for each $\overline{s} \in \overline{S}$ and $a \in A$, we maintain a *potential* for each level $l \in \mathcal{L}$, $\gamma_{\overline{s},a,l}$, which is updated at each level-exploration step, defined as

$$\gamma^{t+1}_{\overline{s},a,l} \leftarrow \begin{cases} 0 & l' \text{ is chosen} \\ \min\left(\gamma^{t}_{\overline{s},a,l} + P(l), 1\right) & \text{otherwise} \end{cases} \tag{3.6}$$

where γ^t_l is the potential at time t and $P(l)$ is defined in Equation 3.5. For readability purposes, define $\gamma^t(\overline{s}, a, l) := \gamma^t_{\overline{s},a,l}$; given this potential function we can slightly alter Equation 3.5 to be

$$\hat{P}(l') = adj(l, l') \frac{\exp(\gamma^t(\overline{s}, a, l'))}{\sum_{l'' \in \mathcal{L}} adj(l, l'') \exp(\gamma^t(s, a, l''))} \tag{3.7}$$

which defines a new distribution from which to sample new levels of autonomy to explore.

In our experiments, a potential matrix was initialized for the CAS model and updated each time the autonomy profile was updated via gated exploration. Gated exploration was implemented by sampling from the above distribution to update the autonomy profile for each $(\overline{s}, a)$ input by including the sampled level if not in $\kappa(\overline{s}, a)$ already, and otherwise doing nothing. The "gated" element was simulated in all experiments by observing the likelihood of an override, and adding the highest level (the only level disallowed initially) if sampled if the likelihood is below 0.15 for the AV navigation domain or below 0.05 for the AV obstacle passing domain.

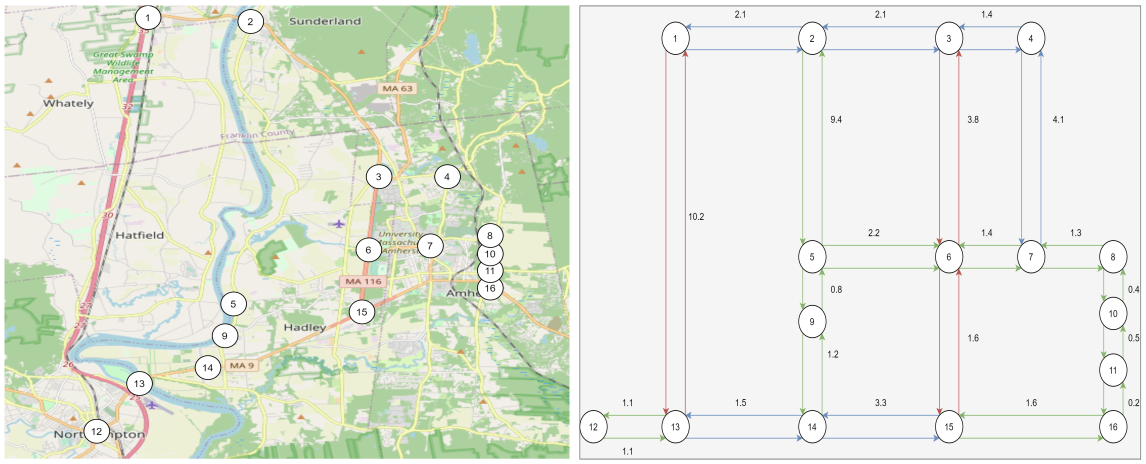


Figure 3.5: *Left*: the map of the region with actual locations from OpenStreetMap. *Right*: the abstracted representation of the navigation graph.

Autonomous Vehicle Navigation

Domain Description

In this domain, an autonomous vehicle operates in a known map represented by a directed graph $G = (V, E)$ where each vertex $v \in V$ represents an intersection and each edge $e \in E$ represents a road; the graph used can be seen in Figure 3.5 and is modeled after locations in the area of Amherst, Massachusetts. The autonomous vehicle is tasked with navigating the map safely from a start vertex to a goal vertex.

Each vertex (intersection) state is represented by an ID for the vertex, a boolean indicator of the presence of pedestrians, a boolean indicator of the presence of an occlusion limiting or blocking visibility, the number of other vehicles at the intersection (0-4), and the vehicle's heading. Each edge (road) state is represented by a start vertex ID, a destination vertex ID, the number of drivable lanes on the current road segment, the direction of travel, and a boolean indicator of the presence of an obstruction blocking the agent's lane. Additionally, each edge is associated with a known length and speed of travel. Model parameters dictating the probabilities of each state variable (e.g., the probability of a pedestrian being at a given intersection upon reaching it) are assumed to be known offline and given as part of the model input. In vertex states, the agent can either `Go Straight`, `Turn Right`, `Turn Left`, `U-Turn`, each of which has a cost of 10.0, or `Wait`, which has a cost of 1.0. All maneuvers succeed deterministically. In edge states, the agent can either `Continue` or `Overtake` an obstruction, each with unit cost. `Overtake` is assumed to succeed with probabilities $[0.2, 0.5, 0.8]$ depending on the number of lanes. `Continue` fails deterministically in the presence of an obstruction, and if there is no obstruction transitions the agent to the end-vertex of the edge with probability $p \propto$ *speed*(e) / *length(e)* or otherwise to the same edge with some probability of an obstruction occurring. We model the expected duration as part of the transition function, rather than the cost function, to allow for the development of an obstruction in the AV's lane while traversing an edge segment, which may be very long in real life.

We consider the following levels of autonomy, $\mathcal{L} = \{l_0, l_1, l_2, l_3\}$ where l_3 does not require any involvement from the human at all (i.e., we assume the probability of an override is 0), l_2 allows the agent to execute an action under supervision, during which the human may override the action if they deem it unsafe, l_1 requires explicit approval from the human for an action prior to its execution during which, if approval is received, the agent may attempt to execute the action under supervision, and if the action is disapproved by the human the agent must select a different action to perform, and l_0 requires full transfer of control to the human to complete the action.

The autonomy profile, κ, is initialized to $\mathcal{L}$ in edge states without an obstruction and otherwise to $\{l_0, l_1, l_2\}$. The feedback profile, λ, is initialized to be uniformly random over the possible feedback signals. There is an associated cost of 10.0 to the human for operating in l_0, as the human is required to manually control the vehicle, a cost of 2.0 for operating in l_1, a cost of 1.0 in level l_2, and no additional cost to the human when operating in l_3. The system incurs a cost of 3.0 when receiving a negative response in l_1 and a cost of 10.0 when receiving an override in l_2 as we assume that the human completes the intended action.

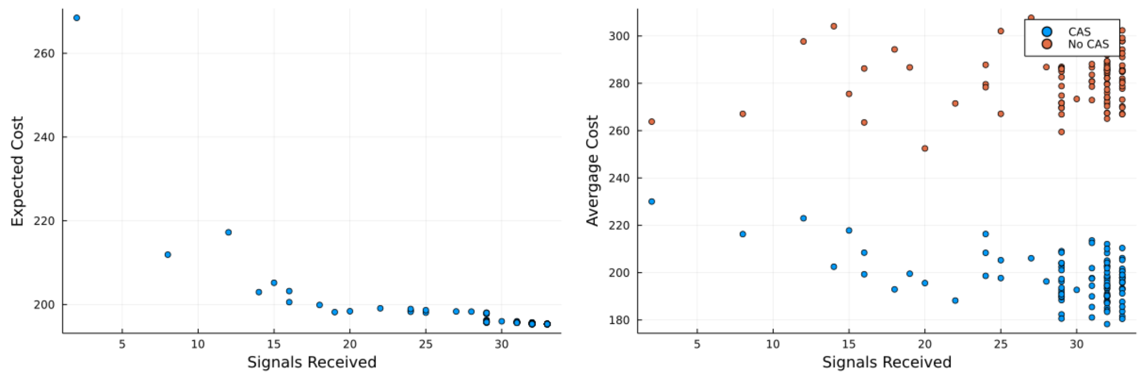
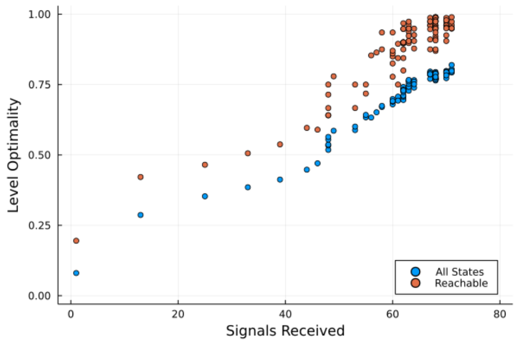


Figure 3.6: Empirical results from simulations of a fixed route (12 $\rightarrow$ 7) showing the expected cost (left) to goal of a CAS and the average cost (right) over 100 trials with a CAS (blue) and without a CAS (red) as a function of the number of signals received.

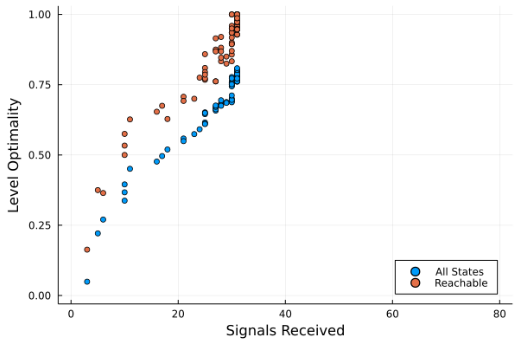
Results

To validate the CAS model in the AV navigation domain, we randomly selected a start node and a goal node each episode to ensure that the system had the ability to visit the entirety of the graph. We repeated this for four different human authorities where we varied their consistency: 0.8, 0.9, 1.0 (i.e., perfectly consistent), and, in the final case, a human who starts with a very low consistency (0.6) to reflect their poor understanding of the capabilities of the system, but increases their consistency by a small amount (0.01) each episode to reflect their improved understanding of the capabilities of the system over time as they interact with it. Figures 3.6, 3.7 and 3.8 report the results from the experiment conducted in the autonomous vehicle navigation domain.

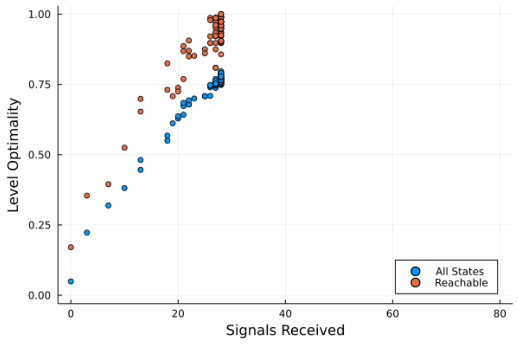
Figure 3.6 depicts the results on a fixed route (node 12 to node 7 in Figure 3.5). The left graph shows the expected cost of the route and the right graph shows the actual mean cost (averaged over 100 simulations) of the CAS (blue) compared against an agent just using the domain model agnostic to its competence, with a human overriding as necessary (i.e., effectively always operating in level l_2) (red). These results demonstrate that by learning an accurate competence model and incorporating that into the planning model, a CAS can efficiently (< 40 feedback signals) improve both its average performance and expected performance, significantly outperforming a system that is agnostic to its competence and the dynamics of human interaction. These experiments were taken from the human with consistency $\epsilon = 0.9$ but we note that very similar results were obtained in all cases. Figure 3.7 depicts in the left column the convergence of the level-optimality of the competence-aware system as a function of the number of feedback signals received, and in the right column the number of signals received over the course of 100 episodes (where each episode is a random route) for a system with a CAS (blue) and a system without a CAS (red). Each row corresponds to a human authority with a different consistency, ϵ, as detailed above. In all cases, the level optimality reaches 100% over all reachable states in the domain. Interestingly, in Figure 3.7d, the results are more comparable to a human with a fixed consistency of 0.9 or 1.0 in both level-optimality convergence rate and the rate at which feedback signals are received, than they are to a human with a fixed consistency of 0.8, which requires roughly twice as many feedback signals to converge to level-optimality. This demonstrates that even a CAS with a human who starts with an initial poor understanding of the system's capabilities, and consequently low consistency, can efficiently reach level-optimality if the human's understanding and consistency improves at a consistent rate. Because we assume that the CAS model captures all features used by the human in determining their feedback, as long as the feedback is not provided randomly, convergence will still occur in the limit, albeit more slowly the greater their inconsistency. In Chapter 4 we investigate the case where the CAS does not capture all features used by the human *a priori*. The figures in the right column illustrate that without a CAS the number of feedback signals provided by the human grows linearly, demonstrating the significant disparity in burden placed upon the human in a system without a CAS model compared to a system with a CAS model. Overall these results demonstrate the primary goal of the CAS model, which is that it enables a system to efficiently reach level-optimality, optimizing the trade-off



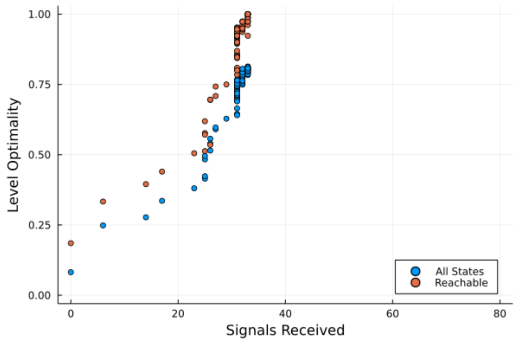
(a) $\epsilon = 0.8$



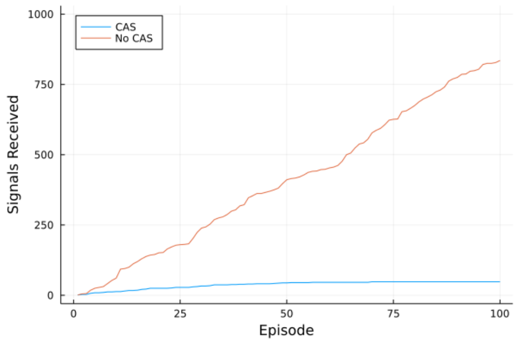
(b) $\epsilon = 0.9$



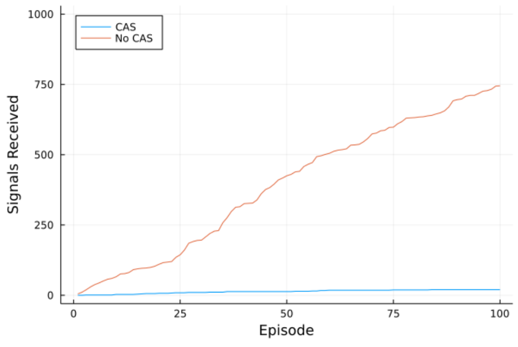
(c) $\epsilon = 1.0$



(d) $\epsilon = 0.6{:}1.0$



(e) $\epsilon = 0.8$



(f) $\epsilon = 1.0$

Figure 3.7: Empirical results from the autonomous vehicle navigation domain with varying levels of human consistency showing the level-optimality as a function of the number of feedback signals received (3.7a – 3.7d) and the number of feedback signals received over the first 100 routes executed (3.7e - 3.7f). In Figure 3.7d, the human consistency increases after each route is executed, mimicking a human whose consistency improves the more it interacts with the system.

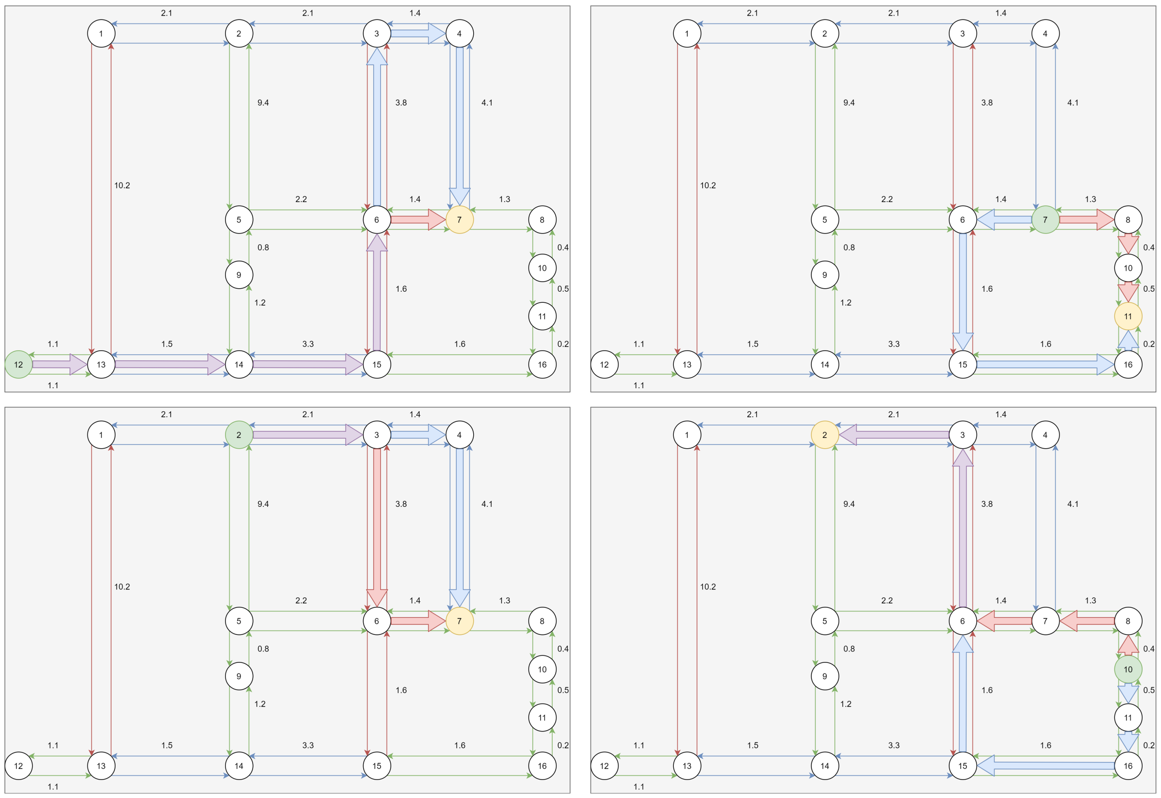


Figure 3.8: Comparison of routes taken before and after the CAS learns its competence. Purple indicates shared route, red indicates route taken by starting model alone, blue indicates route taken by ending model alone. The green node represents the starting node, and the yellow node represents the goal node.

between autonomous performance and human assistance, thereby reducing the net burden placed on the human over the course of the system's operation.

Figure 3.8 depicts the change in routes taken between the first episode and the 100th episode for the CAS model for four fixed routes. Here, purple denotes parts of the route taken that are the same, red denotes parts of the route that are taken in the first episode but not the 100th, and blue denotes parts of the route that are taken in the 100th episode but not the first. The purpose of this figure is to illustrate the *macro* policy changes made as the CAS learns its competence — namely altering its route to avoid states or trajectories of low competence, which would require excessive human assistance — in addition to the *micro* changes of selecting which level of autonomy to use in any given situation. In general, we find that the AV's behavior changes

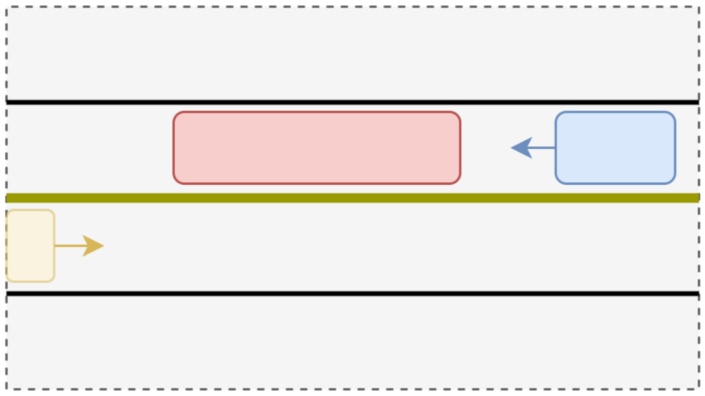


Figure 3.9: Illustration of the AV obstacle passing domain.

to avoid areas densely populated with pedestrians, occlusions, and single lane roads, such as downtown Amherst (nodes 8-11) and UMass Amherst (nodes 6-8).

Autonomous Vehicle Obstacle Passing

Domain Description

In this domain, modeled after the problem depicted in Example 1 and depicted in Figure 3.9, an autonomous vehicle must overtake an obstacle that is blocking its lane on a one-lane road. Importantly, this maneuver required that the AV drive into the oncoming traffic's lane to overtake the obstacle, a potentially dangerous maneuver.

Each state is represented by the vehicle's position (0-4), the position of an oncoming vehicle (0-3, or unknown), and whether the oncoming vehicle has given priority to the AV to attempt its overtake. Model parameters dictating the behavior of oncoming vehicles is assumed to be known offline and given as part of the model input.

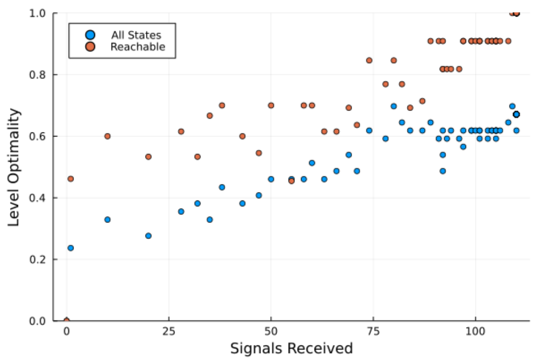
The autonomous vehicle can perform the following actions: `Wait`, `Edge`, and `Go`. `Edge` provides visibility of oncoming traffic to the AV if unknown and otherwise advances the AV's position with probability 0.5. `Go` deterministically advances the AV's position, which results in a crash if the AV and an oncoming vehicle share the same position. `Stop` holds the AV's position, during which time the oncoming vehicles position may change (or become empty), or the oncoming vehicle may give priority to the AV. If the AV has priority it is assumed that the oncoming traffic will stay stopped until the AV has finished its overtake. All actions have unit cost, and crashing incurs a very high cost.

We consider the following levels of autonomy, $\mathcal{L} = \{l_0, l_1, l_2\}$ where l_2 does not involve the human at all, l_1 allows the agent to execute an action under supervision, during which the human may override the action if they deem it unsafe, and l_0 requires full transfer of control to the human to complete the action. Note that we do not include the level l_1 from the prior domain (referred to earlier as "verified autonomy" in Table 3.1) due to the second-to-second nature of decision making in this safety-critical domain, where prompting the human for explicit approval before every action may be impractical or even dangerous.

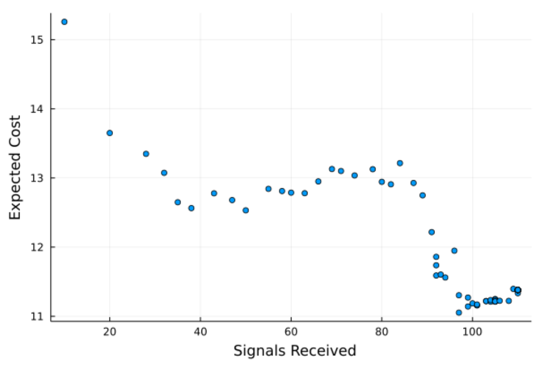
The autonomy profile, κ, is initialized to $\{l_0, l_1\}$ in all cases; i.e., in such a safety critical domain it is expected that, initially, the human is always aware and ready to override the system. As above, the feedback profile λ is initialized to be uniformly random. The human incurs a cost of 10.0 when the CAS operates in l_0 but is assumed to complete the maneuver successfully (i.e., the human does not give back control part way through passing the obstacle), a cost of 1.0 when supervising in l_2, and no cost in l_3. The system receives a penalty of 10.0 when being overridden by the human.

Results

In the AV obstacle passing domain, the problem—i.e., the initial state and goal state—stayed fixed each episode. Figures 3.10a and 3.10b report the results from the experiment conducted in the autonomous vehicle obstacle passing domain. Figure 3.10a shows the level-optimality of the CAS over all states in the domain and all reachable states (each episode) plotted against the number of feedback signals received from the human, in this case consisting only of overrides. The figure illustrates



(a) Autonomous Vehicle Obstacle Passing Domain Level-Optimality



(b) Autonomous Vehicle Obstacle Passing Average Cost

Figure 3.10: Empirical results from the autonomous vehicle obstacle passing domain. Left: depicts the level-optimality over reachable states (red) and all states (blue) as a function of number of received feedback signals. Right: depicts the average task cost (1000 simulations) as a function of number of received feedback signals.

that the CAS is able to converge to level-optimality on all reachable states in the domain with slightly more than 100 feedback signals. The slower convergence rate is due to a stricter requirement on gated exploration due to the more safety-critical nature of the domain. 100% Level-optimality is not reached on the whole state space due to the absence of a portion of the state space ever being visited (or even reachable), preventing the human authority from providing any feedback for actions taken in those states. Figure 3.10b reports the expected cost of overtaking the obstacle and illustrates that the expected cost decreases as the level-optimality increases, corroborating the results from the previous domain. This also demonstrates that, in certain domains, performance may be improved to near optimal performance without even needing to converge to full level-optimality across the entire state space due to variations in state reachability trends.

3.4 Partially-Observable CAS

Throughout this chapter, we have thus far only considered the application of competence-aware systems to fully-observable domains, modeled as either an MDP or an SSP. However, many interesting domains in the open world are *not* fully observable, and include various sources of uncertainty over the system's state at each time step. For example, perceptual information can be noisy, providing only a distribution over certain state factors; inference of state information from perceptual information can itself be imperfect, leading to inferred states of the world that are incorrect, with some probability; or state information may simply be unavailable to the system at certain times, such as if there is an occlusion that blocks a system's vision entirely. It is therefore important, in the pursuit of developing competence-aware system in the open world, that we consider these cases, and the question of how best to model competence when a system is unaware of its exact state. However,, there are two challenging factors that must be addressed; first, how to associate feedback from the human with states in a belief state for the purpose of learning competence (and, implicitly therefore, what to assume about what the human themselves observes); and second, how to best define competence in belief states in a way that still ensures the core properties we wish to preserve.

3.4.1 Preliminaries

Recall from Chapter 2 that in a POMDP, instead of observing its true state, the agent receives an observation $\omega \in \Omega$ in belief state $b \in \Delta^{|S|}$ after taking action $a \in A$, updating their belief state to $b' \in \Delta^{|S|}$ given both the observation function $O : A \times S \to \Delta^{|\Omega|}$ and $T : S \times A \times A \to [0, 1]$ according to the *belief update equation* (Equation 2.17). We can now redefine the CAS model for POMDPs as follows:

Definition 21. A *partially observable competence-aware system* (POCAS) is represented by the tuple $\langle \overline{S}, \Omega, \overline{A}, \overline{T}, O, \overline{C}, \gamma \rangle$ where:

- $\overline{S} = S \times \mathcal{L}$ is a finite set of states comprised of a domain state $s \in S$ and a level of autonomy $l \in \mathcal{L}$.

- Ω is a finite set of observations comprised of a domain observation $\omega \in \Omega$ and a level of autonomy $l \in \mathcal{L}$.

- $\overline{A} = A \times \mathcal{L}$ is a finite set of actions comprised of a domain action $a \in A$ and a level of autonomy $l \in \mathcal{L}$.

- $\overline{T} : \overline{S} \times \overline{A} \to \Delta^{|\overline{S}|}$ is a transition function representing the distribution over successor states $\overline{s}' \in \overline{S}$ given that the agent performed action $\overline{a} \in \overline{A}$ in state $\overline{s} \in \overline{S}$.

- $O : \overline{A} \times \overline{S} \to \Delta^{|\Omega|}$ is an observation function representing the distribution over observations $\omega \in \Omega$ given that the agent is in state $\overline{s} \in \overline{S}$ having just performed action $\overline{a} \in \overline{A}$.

- $\overline{C} = \begin{bmatrix} C & \mu & \rho \end{bmatrix}$ is a vector of cost functions.

- $\gamma \in (0, 1]$ is a discount factor.

We can express the POCAS as an equivalent belief-state MDP (Definition 6) with belief-states $\overline{B} = \Delta^{|\overline{S}|}$, where a solution is a policy, $\pi : \overline{B} \to \overline{A}$. As with the fully-observable CAS model, we assume a standard linear scalarization approach to the multiple objectives with weight vector $\mathbf{w}$, such that the objective is to find the policy that minimizes the function $\mathbf{w}^T\overline{\mathbf{V}}^\pi(\overline{b})$ over all $\overline{b} \in \overline{B}$. Here, $\overline{\mathbf{V}}^\pi(\overline{b}) = [\overline{V}_1^\pi(\overline{b}) \cdots \overline{V}_k^\pi(\overline{b})]$ where each $\overline{V}_i^\pi(\overline{b})$ is defined as

$$\overline{V}_i^\pi(\overline{b}) = \overline{c}_i(\overline{b}, \pi(\overline{b})) + \gamma \sum_{\overline{b}' \in \overline{B}} \overline{\tau}(\overline{b}, \pi(\overline{b}), \overline{b}')\overline{V}^\pi(\overline{b}') \tag{3.8}$$

such that

$$\overline{\tau}(\overline{b}, \overline{a}, \overline{b}') = \sum_{\omega \in \Omega} Pr(\overline{b}'|\overline{b}, \overline{a}, \omega) \sum_{\overline{s}' \in \overline{S}} O(\omega|\overline{a}, \overline{s}') \sum_{\overline{s} \in \overline{S}} \overline{T}(\overline{s}, \overline{a}, \overline{s}')\overline{b}(\overline{s}) \tag{3.9}$$

and

$$\overline{c}_i(\overline{b}, \overline{a}) = \sum_{\overline{s} \in \overline{S}} \overline{b}(\overline{s}) \mathbf{w} \overline{C}_i(\overline{s}, \overline{a}) \tag{3.10}$$

are the belief-state transition and cost function(s) respectively.

3.4.2 Constraining Autonomy in Belief States

Recall that a fully-observable CAS can constrain its policy space, Π, by κ to only allow policies that satisfies the condition that if $\pi(\overline{s}) = (a, l)$, then $l \in \kappa(s, a)$. Unfortunately, this approach does not naturally extend in the same manner, i.e., by constraining the policy on belief states according to a distributional extension of κ. To see why, consider the following "norm-based" methods for κ defined on a belief state $b \in \Delta^{|S|}$ and action $a \in A$:

$$\begin{aligned}
\texttt{Optimistic:} &\quad \cup_{s \in S | b(s|\overline{s}) > 0}\, \kappa(s, a) \\
\texttt{Conservative:} &\quad \cap_{s \in S | b(s|\overline{s}) > 0}\, \kappa(s, a) \\
\texttt{Max-Density:} &\quad \operatorname*{argmax}_{\kappa \in \mathcal{P}(\mathcal{L})} \sum_{s \in S} b(s|\overline{s}) \big[\kappa(s, a) == \kappa\big] \\
\texttt{Max-Likelihood:} &\quad \kappa\Big(\operatorname*{argmax}_{s \in S} b(s|\overline{s}), a\Big)
\end{aligned}$$

The potential issue with this approach – that is, selecting and adhering to a "norm" when determining which level of autonomy to us – is that the selected level of autonomy may not be *allowed* in the autonomy profile on the true underlying domain state in all cases except for `Conservative`, where the intersection may instead be the empty set meaning that there is no level of autonomy that is allowed for a given action, i.e., the action is not allowed at all. Although in practice this may be unlikely to occur, it is simple to create a problem where this leads to arbitrarily bad performance.

In general, competence may not be dependent on all state features, and consequently only those features that are necessary for modeling competence need to be observable. However, we can also not guarantee that this requirement is satisfied for arbitrary problem settings either. Instead, we propose that instead of conditioning the autonomy profile on the belief state which may be highly uncertain, we can instead condition it on the most recent *observation* made by the system, which is always fully observed by definition and contains all fully-observable state information. Consequently, we can redefine the POCAS state space to be $\overline{S} = S \times \Omega \times \mathcal{L}$ to ensure that both the autonomy profile, κ, and the competence, χ, are both defined on the fully-observable portion of the state (in the worst case, only the last observation) as desired. This is also an attractive solution as the belief space is, in general, infinite, whereas there are only a finite, discrete number of observations in the observation set over which the POCAS will learn its competence.

3.4.3 Evaluation

To evaluate the partially-observable CAS, we consider a modified, partially-observable version of the obstacle passing domain (Figure 3.9). Here, the autonomous vehicle cannot directly observe the position (or existence) of oncoming and trailing vehicles which are a causal factor in their competence. Instead, the vehicle can observe whether there is oncoming or trailing light detected which may indicate the presence of an oncoming or trailing vehicle, respectively. The likelihoods are determined by environmental factors such as time of day and weather conditions. However, the other elements of the CAS model such as the levels of autonomy and feedback signal set are unchanged. We provide the competence-learning results in Figure 3.11. Here, we plot the level-optimality as a function of feedback signals over all state-action pairs in the domain (blue) and all state-action pairs visited by the agent during simulation (red). Although the level-optimality over all state-action pairs plateaus around 62.5% as the system, acting conservatively due to the cost of failure and not encountering a large portion of the state-action space while learning, the agent quickly reaches near 100%

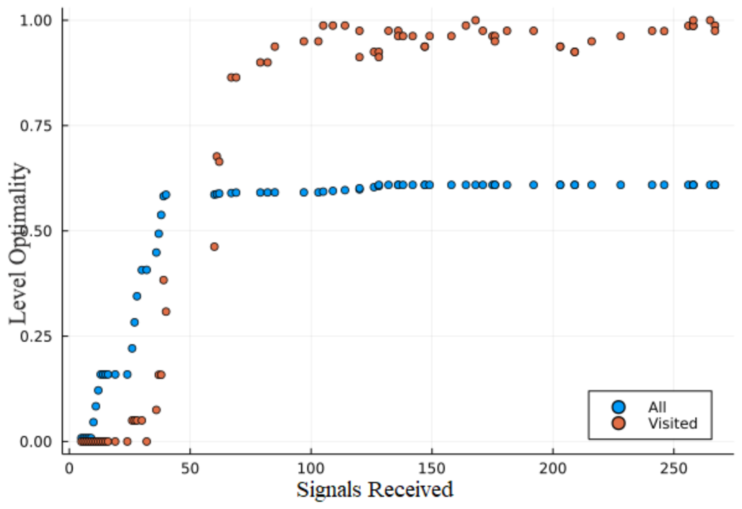


Figure 3.11: Partially-Observable AV Obstacle Passing Domain Level-Optimality

level-optimality on its actual operative performance, demonstrating that a CAS can still effectively operate at its competence in a partially-observable domain when we condition competence on only the most recent observation.

3.5 Discussion

3.5.1 Autonomy Profile Initialization

Because we restrict the system to choose policies from Π_κ, if the autonomy profile κ is altered, so too is the space of allowed policies. Hence, there is a trade-off when setting the initial constraints on the allowed autonomy of the system, i.e., κ. One can take a conservative approach and constrain the system significantly, for instance setting $|\kappa(s,a)| = 1$ so that a single level is deterministically selected for every $(s,a) \in S \times A$, reducing the problem complexity to solving the underlying domain model. However, doing so risks a globally sub-optimal policy with respect to $\mathcal{L}$ and may, depending on the initial κ, make reaching the globally optimal policy impossible. On the other extreme, one can take a risky approach and not constrain the system at all *a priori*, leaving the decision of choosing the level of autonomy completely up to the system when solving its model. This approach, while necessarily containing the optimal policy (subject to the agent's model) is naturally slower due to the larger policy space and inherently less safe as the agent can take actions in undesirable levels. Figure 3.4 illustrates different partitionings of the policy space under different autonomy profiles.

We propose that in practice, the desired initialization is somewhere in the middle; κ should be less constraining in situations where the expected cost of failure is relatively low, and more constraining in situations where it is high. While the model makes no such requirements, in many practical settings such information may be at least partially known *a priori* for a specific domain. For instance, in an autonomous vehicle, κ should be more constraining initially in situations involving pedestrians, poor visibility, or chaotic environments such as large intersections with multiple vehicles; however, initial testing may indicate that driving along a highway is low-risk and may not require a highly constraining κ.

3.5.2 Model Assumptions

We now discuss the practical considerations of the main assumptions made in Section 3.1.4: (1) the human authority, $\mathcal{H}$, provides consistent feedback, (2) the human authority's feedback comes from a stationary, Markovian distribution, and (3) the environment the agent operates in is stationary.

Implicit in Assumption (1) is that humans respond appropriately to each situation, possibly with some noise representing the likelihood of human error. However, because of the limited scope of the system's domain model, it could be that perfectly consistent feedback from $\mathcal{H}$'s perspective is *perceived* to be random by the system, particularly when it is not aware of the domain features that explain the human feedback. As an example, consider a robot that can open 'push' doors and cannot open 'pull' doors, but does not model this discriminating feature. If the robot cannot discriminate between these types of doors, consistent and correct human feedback (approving autonomously opening ‘push’ doors only) may be perceived by the robot to be arbitrary or random. Although in practice one may wish to avoid such situations, we emphasize that the system *will still converge to its competence for the state features it uses*—possibly a low competence—when the feedback distribution appears to be random.

Assumption (2)—that the human feedback distribution $\lambda^{\mathcal{H}}$ is stationary and Markovian from the start—implies that the human has good knowledge of the system from the start. That may not be realistic in certain domains. It is more likely that the feedback signals may vary based upon the observed performance of the system over time. However, as the human authority observes the system’s performance, it is reasonable to assume that their feedback distribution will eventually reach a stationary point as long as the system’s underlying capabilities stay fixed. Therefore, even if there are erroneous feedback signals provided early in this process, in the limit the system will still converge to a fixed competence. Two possible means of expediting this is to introduce a training phase at the beginning of the system’s deployment to allow the human to observe the system’s performance and develop accurate expectations regarding the system’s capabilities, and to introduce standardized feedback criteria that is made known to the human *a priori*.

Finally, while the assumption of environmental stationarity may not always hold in general, for many real-world problems it is reasonable that the systems’ are designed with respect to the general environment that the system is expected to handle, and that, consequently, the system’s domain model should capture the any environmental variability it may face. For instance, an autonomous vehicle built and deployed in the spring should reasonably be expected to have a model for snowy conditions if deployed in an environment where it snows during the winter. However, it is also the case that online model updates can be used to ameliorate scenarios where this *is not* the case, as is often done in many real-world systems; in such cases, a system’s designer can conservatively err on the side of restarting the competence-learning from scratch, but investigating ways to bootstrap a system's competence model in such situations is an interesting direction for future work on competence-aware systems.

3.5.3 Complexity of Partial Observability in CAS

In this chapter we showed that the competence-aware system framework can be generalized from fully-observable MDPs to partially-observable MDPs by augmenting the state representation with the observation set, and conditioning competence on the most recent observation. However, a limiting factor is the computational tractability of solving POMPDs as the size of the state space increases, as our approach increases the state space multiplicatively by the size of the observation set. Our experiments considered a simple observability condition with only 4 unique observations for the non-fully observable state factors, yet still relied on online POMDP solvers as offline solvers did not find a policy within a reasonable amount of time. Extending the competence-aware framework to POMDPs with larger and more complex observation spaces may therefore become intractable to solve completely.

Consequently, an important area for future work will be to reduce the computational complexity in larger, more complex settings. We propose that one approach is to leverage the recently proposed *semi-observable Markov decision process* [9] which was shown to have significant computational benefits over existing state-of-the-art POMDP solvers, and often led to higher quality solutions as well. One shortcoming of this approach is the reliance on the addition of a *reveal* action which deterministically reveals the agent's state on demand (at high cost); we propose that the competence-aware system, where there is an existing reliance on human assistance, may naturally provide a setting where such an action is feasible.

3.6 Conclusion

In this chapter, we introduce a new framework for representing, learning, and reasoning with self-competence models in semi-autonomous systems. Competence in our approach represents the level of autonomy that the system can handle reliably based on human feedback. More precisely, we define competence as the *optimal level of autonomy* in any given situation, consistent with perfect human feedback. We present a novel decision-making framework, *competence-aware systems*, that enables a semi-autonomous system to learn its own competence over time through interactions with a human authority. The result is a system that can handle risky scenarios by relying on the human authority to compensate for limitations or constraints on its autonomous abilities, while simultaneously optimizing its autonomous operation to reduce *unnecessary* reliance on humans.

We illustrate the operation of a competence-aware system with a running example and prove several theoretical properties of the CAS model. In particular, we prove that under standard convergence assumptions the model will converge to *level-optimality*, guaranteeing that the system consistently operates at its competence. We test the efficacy of our model empirically on two simulated autonomous vehicle domains, at different levels of reasoning abstraction, and demonstrate that the competence-aware system can efficiently reach high level-optimality, optimizing the trade-off between its own autonomous operation and human assistance, and leading to less burden on the human and a more cost-effective overall plan.

Finally, we show that our approach extends to the partially-observable setting in a well-defined way by conditioning competence on the most recent observation, rather than the underlying state or current belief state, and provide empirical evidence from simulation that demonstrates that a partially-observable CAS can efficiently learn to operate at its competence nearly 100% of the time.

CHAPTER 4

IMPROVING COMPETENCE VIA ITERATIVE STATE SPACE REFINEMENT

In Chapter 3, we introduced the *competence-aware system* (CAS) as a planning framework for semi-autonomous systems to reduce unnecessary reliance on human assistance by learning their own competence and accounting for it during planning. However, while the CAS model enables a semi-autonomous system to optimize its autonomy over time, it is limited by the features in its fixed model. As discussed previously in this thesis, many problems in the open world are too complex to fully specify *a priori* all features that will be relevant over the course of the system's deployment, even with expert knowledge of the domain. This is particularly prevalent with features that may not directly impact the technical functionality of the autonomous agent (e.g., its domain model) but rather are factors that influence the human's feedback, which may encompass additional features that affect other elements such as comfort or social behavior [13, 74].

Preliminary analysis of override data collected on a real autonomous vehicle prototype from two different safety drivers corroborates this claim. Here, the AV could either be in *supervised autonomy*, or could defer full control to the human; overrides corresponded to braking or accelerating registered by the human driver while the AV was operating in supervised autonomy. The results of this analysis can be seen in Figure 4.1 and Table 4.1 where we provide the correlation matrix for each type of override with every feature used by the CAS model implemented on the AV for each human safety driver. These results demonstrate two important facts. First, the difference in correlation matrices between Human 1 and Human 2 illustrate that

	Human 1			Human 2		
$\hat{\mathbf{F}}$	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
$\mathbf{f_1}$	0.171	-0.146	-0.055	0.222	0.255	-0.410
$\mathbf{f_2}$	0.293	-0.158	-0.209	-0.037	-0.109	0.111
$\mathbf{f_3}$	-0.399	0.267	0.220	-0.212	-0.197	0.361
$\mathbf{f_4}$	0.375	-0.335	-0.103	0.384	-0.170	-0.313
$\mathbf{f_5}$	-0.379	0.257	0.205	-0.372	0.311	0.208
$\mathbf{f_6}$	0.064	0.043	-0.141	0.045	-0.183	0.069
$\mathbf{f_7}$	-0.030	0.118	-0.104	0.044	-0.019	-0.036
$\mathbf{f_8}$	0.179	-0.110	-0.112	0.044	-0.019	-0.036
$\mathbf{f_9}$	0.085	-0.093	-0.002	-0.062	0.027	0.051
$\mathbf{f_{10}}$	0.108	-0.151	0.038	-0.237	0.104	0.193
$\mathbf{f_{11}}$	0.175	-0.059	-0.168	0.325	0.295	-0.549

Table 4.1: The correlation matrices of each override signal with each feature.

feedback, and the features that determine that feedback, can vary significantly between humans, meaning there is no "one-size-fits-all" feedback model. Second, the lack of any feature having a correlation coefficient greater that ± 0.4 indicates that it is challenging, even with expert input, to capture all of the causal features used by all humans *a priori*. If the CAS model does not represent certain features in its model that are used by the human in deciding their feedback signals (either explicitly or implicitly), the human's feedback may appear inconsistent or even random, leading to low competence and a potentially high degree of improper reliance on the human stemming from an underspecified model. Consequently, for these systems to be most effective in the real world it is important that they are equipped with a means of updating their model online to better align with the human's model so that they can better predict the correct feedback likelihoods.

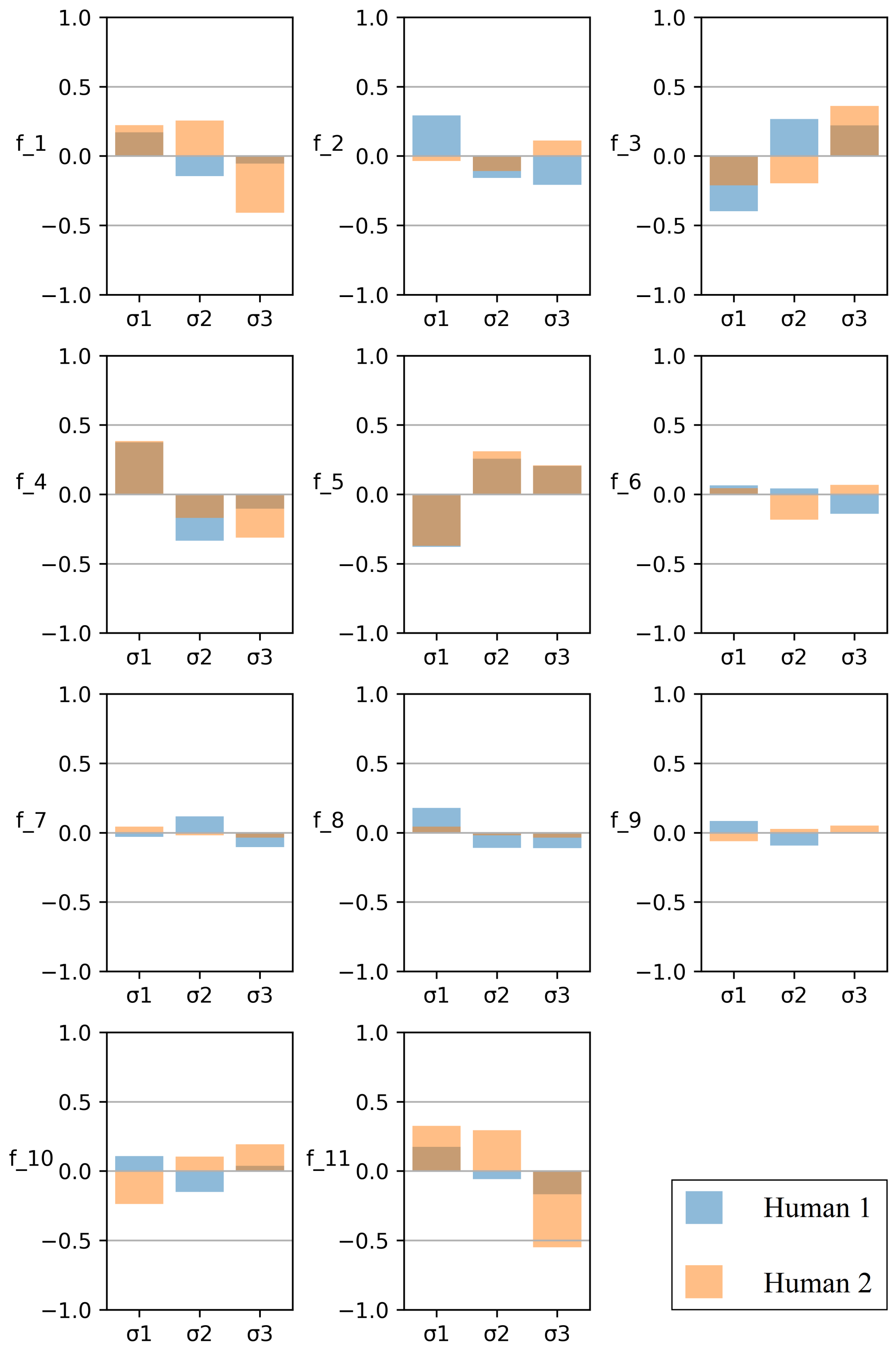


Figure 4.1: The correlation of each override signal (σ_i) and each feature (f_j) for Human 1 (blue) and Human 2 (orange).

4.1 Improving Competence Online

To address this shortcoming, we propose in this chapter a method for providing a CAS the ability to improve its competence over time by increasing the granularity of its state representation through online model updates [8]. The approach works by identifying states that are deemed *indiscriminate* under the system's current feedback profile, i.e., unable to predict human feedback with high confidence, and attempts to find the feature, or set of features, that is available to the system but currently unused that best discriminates human feedback, leading to a more nuanced drawing of the boundaries between regions of the state space with different levels of competence. An example of this process can be viewed in Figure 4.3. The approach leverages the CAS model in two critical ways. First, it exploits existing information available in a standard CAS model in the form of human feedback to identify where new features should be added, adding no additional work to the human. Second, it exploits key properties of the human-agent interaction to avoid needing to directly alter the transition function or reward function, modifying only the state space directly. Consequently, the entire process can be performed online and fully autonomously.

Example 5. Recall the scenario in our running example, where the AV (blue) must overtake an obstacle blocking its lane (red) by driving into the oncoming traffic's lane (yellow). Now, consider the existence of a trailing vehicle (or vehicles) waiting behind the AV (green); the existence of trailing vehicles may not be included in the state representation of the domain model as they do not affect the decision making of the AV from a technical perspective (that is, they do not influence the success or failure probabilities of each action, do not influence the safety of the actions, and short of rear-ending the AV do not directly alter the AV's state), and serve only to increase the state space of the planner. However, it may be the case that the human in the AV is actually *more likely* to override safe behavior, such as waiting if there is an

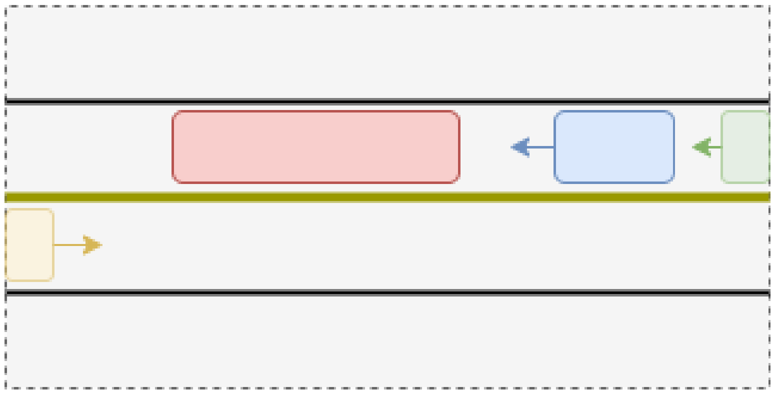


Figure 4.2: Illustration of Example 5

oncoming vehicle, and take manual control of the vehicle due to the social pressure exerted by the trailing vehicle's existence.

4.1.1 Indiscriminate States

Let $\mathcal{S}$ be a competence-aware system. In practice, when a robotic system is deployed into the open world, both the exact environment the system will operate in, and the human authority it will interact with, may not be known *a priori*. Naively including all possible features available to the system from perception or external sources in its planning model may make planning intractable without benefit in the case of many of the features that do not add useful information for decision-making and serve only to increase the number of states. Hence, we assume that $\mathcal{S}$ has available to it a *complete feature space* that can be partitioned into an *active feature space* that is used by $\mathcal{S}$ and an *inactive feature space* that is not yet used by $\mathcal{S}$ in its planning model. However, as $\mathcal{S}$ receives additional feedback over time, $\mathcal{S}$ will learn to exploit some of the inactive features, adding them to its state representation to more effectively align its features with those used by the human authority.

Definition 22. Given the **complete feature space** $F = \{F_1, F_2, ..., F_n\}$ available to $\mathcal{S}$, the **active feature space** is denoted as $\hat{F} \subseteq F$, and the **inactive feature space** as $\breve{F} = F \setminus \hat{F}$.

We say that a state $\overline{s} \in \overline{S}$ is *indiscriminate* if, intuitively, the active feature space is missing features needed to properly discriminate the feedback received from the human for the state $\overline{s}$. The condition states that for at least one action there must be no feedback signal that, under the system's current feedback profile, can be predicted with high probability. The intuition is that, under the assumption of ϵ-consistency and a ground truth feedback, situations where the agent cannot predict feedback with high probability indicate that a feature may be missing from its state representation causing the probability mass to be normalized over the remaining features in its active feature space. We formalize this below.

Definition 23. Let the human authority $\mathcal{H}$ be ϵ-consistent for $\epsilon > \frac{1}{|\Sigma|}$. A state $\overline{s} \in \overline{S}$ is **indiscriminate** if there exists at least one action, $\overline{a} \in \overline{A}$, where for every feedback signal $\sigma \in \Sigma$, we have the following:

$$\lambda(\sigma \mid \overline{s}, \overline{a}) \leq 1 - \delta \quad \delta \in (1 - \epsilon, 1 - \frac{1}{|\Sigma|}) \tag{4.1}$$

Here, δ is referred to as the *discrimination slack*, and determines the predictive confidence needed for a state to be declared indiscriminate; the lower the slack is set, the higher the confidence needed. The discrimination slack serves to provide a formal trade-off mechanism between increasing the complexity of the underlying planning model, and the completeness of the competence-aware model. The determination of how to set δ may be done via expert knowledge, offline evaluations, or could even be tuned online in a dynamic fashion. To avoid considering states that have a very small amount of data (and hence may be deemed "indiscriminate" due to chance), we consider only states for which the system has collected a sufficient amount of data (determined simply via a fixed threshold, or based on some statistical analysis).

4.1.2 Feedback Discriminators

Given the notion of an indiscriminate state, we can now define the central concept of this approach. A *discriminator* is, intuitively, any *subset* of the inactive feature space that could help the agent to better discriminate feedback from $\mathcal{H}$ for an indiscriminate state. For example, consider the autonomous vehicle agent in Running Example 5 that initially does not consider the existing of a trailing vehicle in its active feature set. Suppose that the human always overrides the vehicle and takes manual control when there is a trailing vehicle if the AV waits for too long before proceeding around the obstruction to maintain safe operation. Without this additional feature in its model, the agent may perceive having received "noisy", or even seemingly random, feedback from the human authority, leading to a feedback profile with low predictive capabilities and a poor competence model, resulting in the AV conservatively transferring control to the human when performing an overtake in situations where it was actually competent to act autonomously. By providing the agent with the ability to add these features to its active feature space, the agent's new feedback profile will be able to predict the correct feedback signal in more situations with higher probability.

Definition 24. A **discriminator** is any subset of $\breve{F}$ that, if added to $\hat{F}$, will improve the performance of λ by at least α, for some $\alpha \in (0, 1)$.

The larger that α is set, the stricter the requirement is on including a new feature. Determining α can be as simple as setting it to be a fixed threshold, or can be via more sophisticated means such as based on the value of information or other information-theoretic metrics. The methodology for selecting discriminators is well explored in the feature selection literature and not the focus of this contribution; standard approaches include mRMR [83], JMI [22], and correlation-based methods [106].

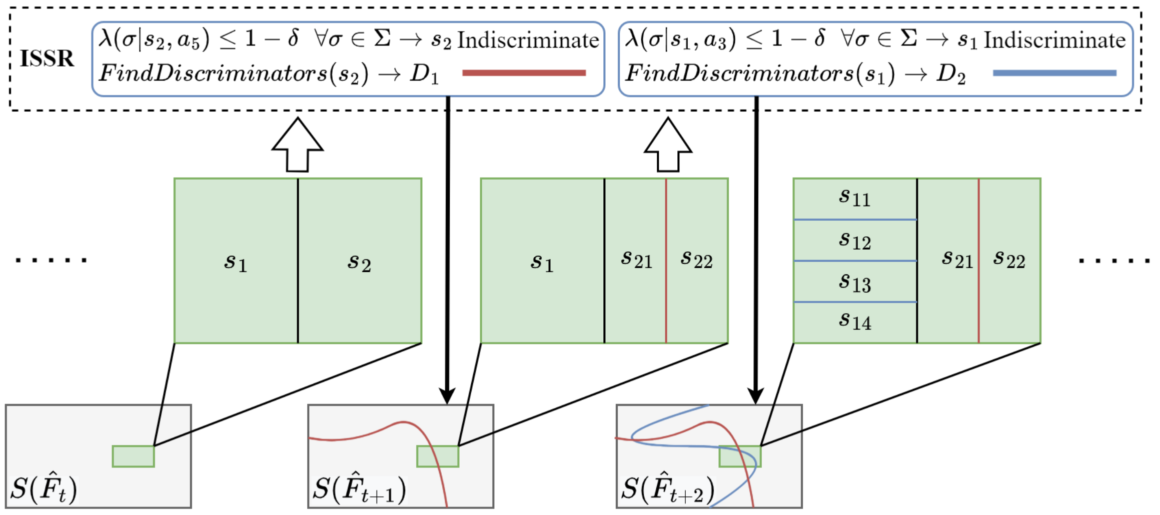

ISSR
$\lambda(\sigma|s_2, a_5) \leq 1 - \delta \;\; \forall \sigma \in \Sigma \rightarrow s_2$ Indiscriminate
$FindDiscriminators(s_2) \rightarrow D_1$
$\lambda(\sigma|s_1, a_3) \leq 1 - \delta \;\; \forall \sigma \in \Sigma \rightarrow s_1$ Indiscriminate
$FindDiscriminators(s_1) \rightarrow D_2$
s_1
s_2
s_1
s_{21}
s_{22}
s_{11}
s_{12}
s_{13}
s_{14}
s_{21}
s_{22}
$S(\hat{F}_t)$
$S(\hat{F}_{t+1})$
$S(\hat{F}_{t+2})$

Figure 4.3: An illustration of our *iterative state space refinement* approach in a simple navigation task. Colored blocks represent different colored doors. Each colored path corresponds to the optimal path under a different granularity of the state space representation. An abstract representation of the state space at different granularities of refinement can be seen in the middle row. As features are identified and added to the system's state space representation, the system can better learn its true competence in a larger portion of the state space, as depicted in the bottom row, enabling it to take paths that better exploit the trade-off between autonomous operation and human assistance.

We define a discriminator as a subset because there may be causal features which if added individually do not help to discriminate the human's feedback, but when added together do (i.e., they are only meaningful in the context of each other). The size of feature subsets to consider when selecting potential discriminators is therefore an important parameter of the approach.

4.2 Iterative State Space Refinement

In this section, we present the main methodological contribution of this chapter, which is a general algorithm we call *iterative-state space refinement*.

Algorithm 1 presents the pseudocode of one iteration of our approach for improving the competence of a CAS via iterative partitioning of the state space by adding new features to the state representation over time. The algorithm first identifies the

```
Algorithm 1: Single–Step State Space Refinement
  Input: A CAS $\mathcal{S}$, dataset $\mathcal{D}$, slack $\delta$, and threshold $M$
  Result: An updated CAS $\mathcal{S}$
1  $\overline{S}^* \leftarrow \{\}$
2  for $\overline{s} \in \mathcal{S}.GetStates()$ do
3      for $\overline{a} \in \mathcal{S}.GetActions()$ do
4          if $\max_{\sigma\in\Sigma} \lambda(\sigma|\overline{s},\overline{a}) \leq 1-\delta$ and
5          $\max_{\sigma\in\Sigma} \Pr[Obs(\mathcal{D}(\overline{s},\overline{a}))|\sigma$ is ground truth$] < p_\epsilon$
6              $\overline{S}^* \leftarrow \overline{S}^* \cup \{\overline{s}\}$
7  if $\overline{S}^* = \emptyset$
8      return $\mathcal{S}$
9  $\overline{s}^* \sim \overline{S}^*$
10 $\mathcal{D}_{train}, \mathcal{D}_{val} \leftarrow Split(\mathcal{D})$
11 $D \leftarrow FindDiscriminators(\mathcal{D}_{train}, \breve{F}, \overline{s})$
12 for $d \in D$ do
13     $\lambda_d \leftarrow train(\hat{F}_1 \times \cdots \times \hat{F}_{|\hat{F}|} \times d, \mathcal{D}_{train})$
14 $d^* = \text{argmax}_{d\in D}\ Evaluate(\lambda_d,\ \mathcal{D}_{val})$
15 if $Validate(d^*, \mathcal{S})$ is $True$
16     $\hat{F} \leftarrow \hat{F} \cup d^*$
17     $\mathcal{S}' \leftarrow Update(\mathcal{S})$
18 return $\mathcal{S}'$
```

current set of indiscriminate states (Lines 1-6). To avoid labeling sparsely sampled state-action pairs as indiscriminate through chance, we limit the process to only consider certain state-action pairs. In particular, only those where the probability of having observed all labeled instances of that element in the existing dataset $\mathcal{D}$, referred to in Algorithm 1 as $Obs(\mathcal{D}(\overline{s}, \overline{a}))$, is at least some threshold p_ϵ conditioned on the assumption that there exists a true correct feedback signal returned with probability at least ϵ by the human for every state-action pair (Line 5). Next, the algorithm samples an indiscriminate state from the set (Line 9) and identifies the most likely discriminators for that state using any standard feature selection technique (in our case, we used mRMR [83] with the FCQ methodology [126]) (Line 11). For each potential discriminator, a new feedback profile is trained using a portion of the full dataset with the discriminator temporarily added to the active feature set (Lines 12–13). The discriminator that leads to the best performing feedback profile, in our case the highest Matthews correlation coefficient, is selected for validation (Line 14). If validation is successful, the discriminator is added to the active feature set and the system is updated (Lines 15–17).

A natural question is whether in the process of adding a discriminator so as to make some indiscriminate states discriminate, we will, as an unintended by-product, make some discriminate state indiscriminate.

Remark 1. Adding a discriminator will never cause a discriminate state to become indiscriminate.

While possibly not obvious *a priori*, this remark is trivially true. Observe that any given discriminate state will either be affected by the discriminator or it will not. If it is not affected, the feedback profile for the state will not change. If the state is affected, then the initial state in question by definition no longer exists. More importantly, we want to ensure that every state is eventually properly discriminated given a sufficient set of features.

The following proposition states that if every feature that the human uses to determine their feedback is available to the robot, then there must be a point in time at which the robot has fully discriminated all states, and no state will become indiscriminate past that point.

Proposition 4. Let I_t be the number of indiscriminate states at time t, and let $\lambda_t^{\overline{s},a}$ be the random variable representing $\lambda(\overline{s}, a)$ after having received t feedback signals for $(\overline{s}, a)$ where each signal is sampled from the true distribution $\lambda^{\mathcal{H}}(\overline{s}, a)$. If $F^{\mathcal{H}} \subseteq F$, $\mathcal{H}$ is ϵ-consistent, $\delta > 0$ and no $(\overline{s}, \overline{a}) \in \overline{S} \times \overline{A}$ is starved, then there exists some $t^* > 0$ for which $I_{t'} = 0$ for all $t' > t^*$.

Proof. First, observe that as $F^{\mathcal{H}} \subseteq F$, if there is a point at which $F^{\mathcal{H}} \subseteq \hat{F}$, then because the sequence $\{\lambda_t^{\overline{s},a}\}$ converges in distribution by Proposition 3, $\lim_{t\to\infty} \Pr(|\lambda_t^{\overline{s},a} - \lambda_{\mathcal{H}}^{\overline{s},a}| > \gamma) = 0 \ \forall \gamma > 0, (\overline{s}, a) \in \overline{A} \times A$. Hence, there exists some $t^* > 0$ for which $\Pr(|\lambda_t^{\overline{s},a} - \lambda_{\mathcal{H}}^{\overline{s},a}| > \delta) = 0$ at which point it is clear that no state will be indiscriminate under δ. Consequently, for the claim to not hold, it must be the case that for every $t > 0$, $F^{\mathcal{H}} \setminus (F^{\mathcal{H}} \cap \hat{F}) \neq \emptyset$. Pick such a t, sufficiently large, for which there is an indiscriminate state $\overline{s} \in \overline{S}$. There is some subset, $G \subseteq F^{\mathcal{H}} \setminus (F^{\mathcal{H}} \cap \hat{F})$, which is a discriminator of $\overline{s}$. As this holds for all $t > 0$ and $\overline{s} \in \overline{S}$, we either reach a satisficing t^* where $F^{\mathcal{H}} \setminus (F^{\mathcal{H}} \cap \hat{F}) \neq \emptyset$, and hence are done, or where $F^{\mathcal{H}} \subseteq \hat{F}$, which contradicts our assumption. $\square$

When using Algorithm 1, each time a new discriminator, d, is added to the active feature set, the state space of the CAS is partitioned as seen in Figure 4.3, increasing the total number of states by a factor of $|d|$. If the CAS's *true* underlying transition function is independent on the discriminator, d, i.e., the features only affect the human's feedback function λ and transition function $\tau_{\mathcal{H}}$, then the process can be run without any additional information or intervention as λ and $\tau_{\mathcal{H}}$ are computed directly from observed data, and are parameters of the function $\overline{T}$. In other words, $\overline{T}$ will not have to be updated directly at any point, and the algorithm can be run in a fully unsupervised capacity, while ensuring that the transition function is correct on all new states in the new partitioned state space. However, if the true transition function is dependent on the discriminator, d, for some states $d(s)$ (denoting the set of states derived from partitioning the state s with discriminator d), the algorithm can still be run, but the conditional affect of d on T for each state in $d(s)$ will be marginalized out. In such a case, the transition function is *no worse*; rather, the model simply did not have the feature expressivity prior to adding the discriminator d to properly express the model's transition imprecision in certain scenarios. Moreover, it is possible to use our approach in an offline capacity as a means of identifying the missing features for system designers, and subsequently improving the system's competence by directly updating the transition and cost functions (e.g., via software updates).

4.3 Evaluation

We evaluated our iterative state space refinement approach (Algorithm 1) on both of the domains described in Chapter 3, where the key difference is that, here, the CAS model is missing features in its initial active feature space that do not impact its transition model (that is, what it is technically capable of doing), but impact the human's feedback signal likelihoods regardless. We test our approach for multiple different simulated humans, each of whom uses different auxiliary features in determining their feedback.

To evaluate the *iterative state space refinement* method, we implemented Algorithm 1 and compared the performance of a CAS with Algorithm 1 and a CAS without it on both of the domains defined above (autonomous vehicle navigation and autonomous vehicle obstacle passing). In both experiments we considered different human users of the autonomous vehicle system, each of whose feedback was conditioned not just on the features already used by the CAS model that directly impacted the CAS's technical performance (i.e., the existence of a pedestrian, an occlusion, etc.) but additionally on auxiliary features, which are tracked by the autonomous vehicle but not included in its *a priori* planning model, as the features in question are different for each person, and do not (directly) impact the transition and cost dynamics of the system.

In the AV navigation domain, the inactive feature set included the following features: whether the AV has a trailing vehicle, a vehicle to its left, or a vehicle to its right, whether the AV has been "waiting" to move, whether it is daytime or nighttime, and whether it is sunny, rainy, or snowy. In the AV obstacle passing domain, we consider the same inactive features except whether there is a vehicle to the AV's left or right, as the problem is for single lane roads.

In the AV navigation domain, we consider two "people" implemented as software agents: the first person is cautious with low trust in letting the AV operate in challenging environmental conditions (even though they do not impact the AV in simulation), for instance taking over control when the system attempts an overtake on a road segment when it is either snowing or rainy and night time. The intuition here is that the weather conditions impacts the human's ability to fully assess the situation and hence the veracity of the AV's actions, prompting them to take control of the vehicle themselves. We refer to them as "Cautious". The second person is motivated by more social factors, and is more likely to take control of the vehicle when there is a trailing vehicle the AV is blocking, and or when the AV has been stopped for too long (either on a road segment behind an obstruction, or at an intersection). We refer to them as "Conscientious".

In the AV obstacle passing domain, we consider three "people" implemented as software agents: the first is motivated by the same features as the first person above; we again refer to them as "Cautious". The second person is motivated by whether there is a trailing vehicle that they are blocking, prompting them to take control if the AV waits to long to attempt its overtake; we also refer to them as "Conscientious". The third person is in a rush and takes over control if the AV is waiting too long or doesn't go when it has priority; we refer to them as "Rushed". Each simulated person is perfectly consistent up to some fixed noise ϵ, within which they return uniformly random feedback.

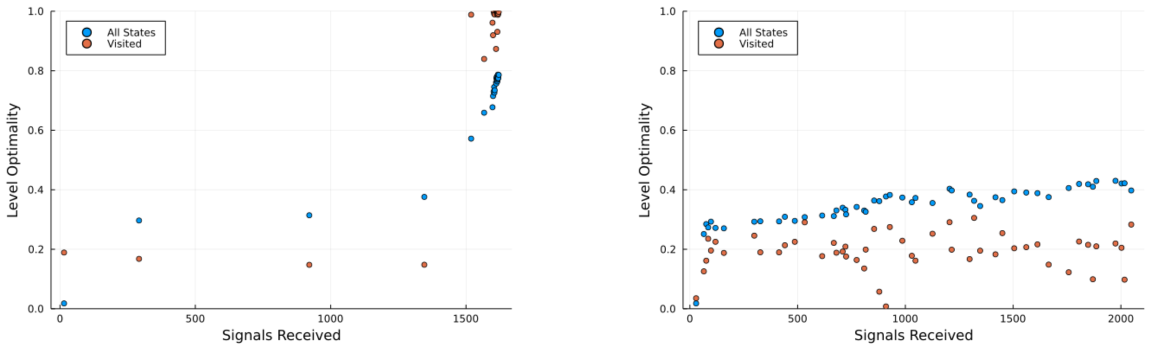
We note that in both domains, some inactive features are never used by any of the humans simulated, and hence we aim to show that our approach does not simply "pick all features" in the inactive feature space. Additionally, one important distinction between the two domains is that the additional inactive features may change at each new state in the AV navigation domain, but are fixed in the AV obstacle passing domain at the beginning of each episode due to the short time horizon of the problem.

Results

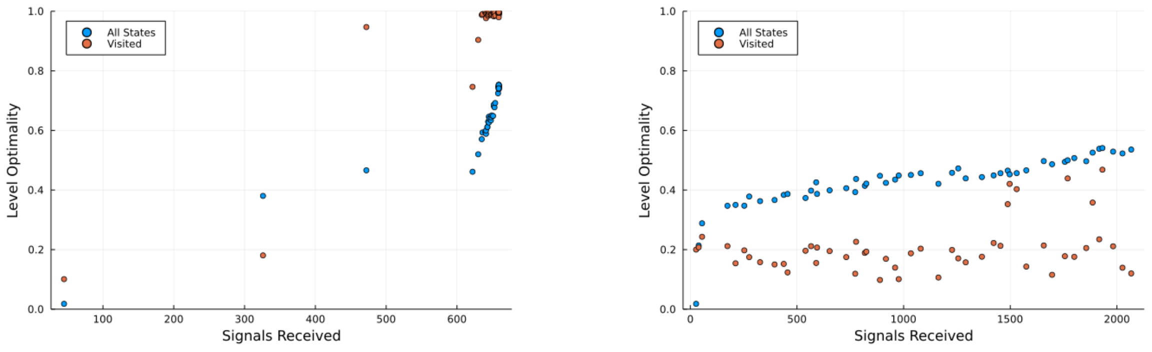
Figure 4.4 shows the results of our experiment, comparing the performance of a CAS with and without the iterative state space refinement (ISSR) approach (Algorithm 1) implemented, on the AV navigation domain with random routes each episode. Figure 4.5 shows the results for the AV obstacle passing domain. In Figure 4.4, we can see that the CAS with the ISSR implemented converges to higher level-optimality on all state in the domain, and 100% level-optimality on all states visited each episode, leading to far fewer feedback signals from the human, for both human authorities. Additionally, in both cases, the only features added to the active feature space where the features in the inactive feature space that were actually used by the humans in determining their feedback.

Figure 4.5 shows the results for the AV obstacle passing domain. Note that we include results on all reachable states here because the additional features stay fixed through each episode, whereas in the AV Navigation domain, they can change throughout an episode and the transition dynamics are (by design) not modeled by the agent.

There are several key takeaways from these graphs. First, if we consider the level-optimality over all states in the domain, it is higher for the ISSR-CAS in the cases of all three human authorities, than for the CAS without ISSR active, indicating that our approach is enabling the CAS to generalize its competence model to a larger portion of the (unvisited) state space. We remark that by adding features in order to refine the state space, the number of states increases multiplicatively with each feature added, meaning that not only is the ISSR-CAS level-optimal in a larger portion of the state space, that directly translates to being level-optimal in a larger number

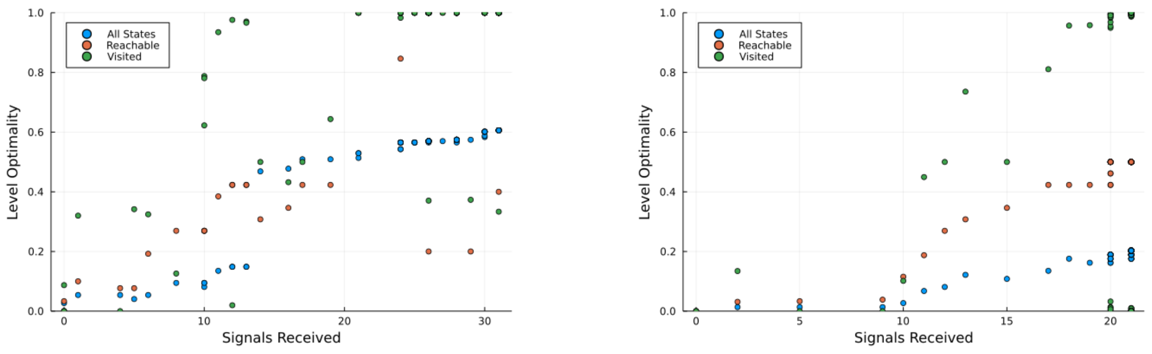


(a) Person 1 (Cautious)

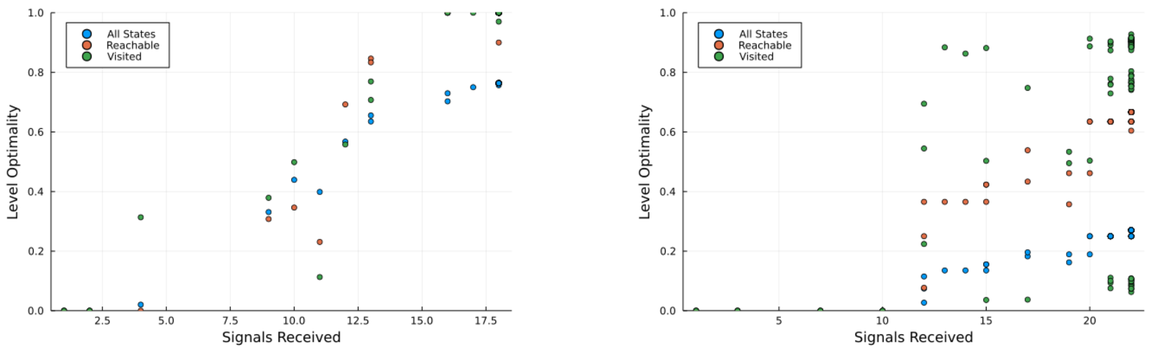


(b) Person 2 (Conscientious)

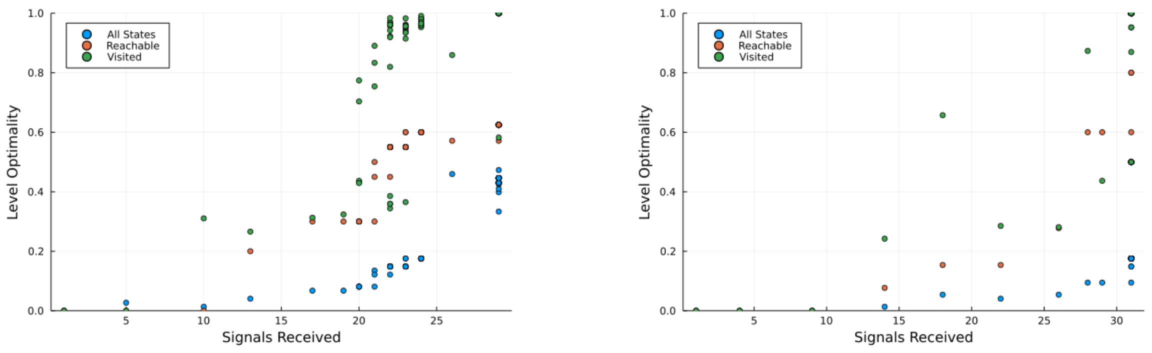
Figure 4.4: Iterative state space refinement results for two human authorities in the autonomous vehicle navigation domain, showing the level optimality after each episode as a function of the number of feedback signals with (left) and without (right) Algorithm 1 implemented. Colors indicate the level-optimality over states visited during each episode (green) and the full state space (blue).



(a) Person 1 (Cautious)



(b) Person 2 (Conscientious)



(c) Person 3 (Rushed)

Figure 4.5: Iterative state space refinement results for three human authorities in the autonomous vehicle obstacle passing domain, showing the level optimality after each episode as a function of the number of feedback signals with (left) and without (right) Algorithm 1 implemented. Colors indicate the level-optimality over states visited during each episode (green), all reachable states each episode (red), and the full state space (blue).

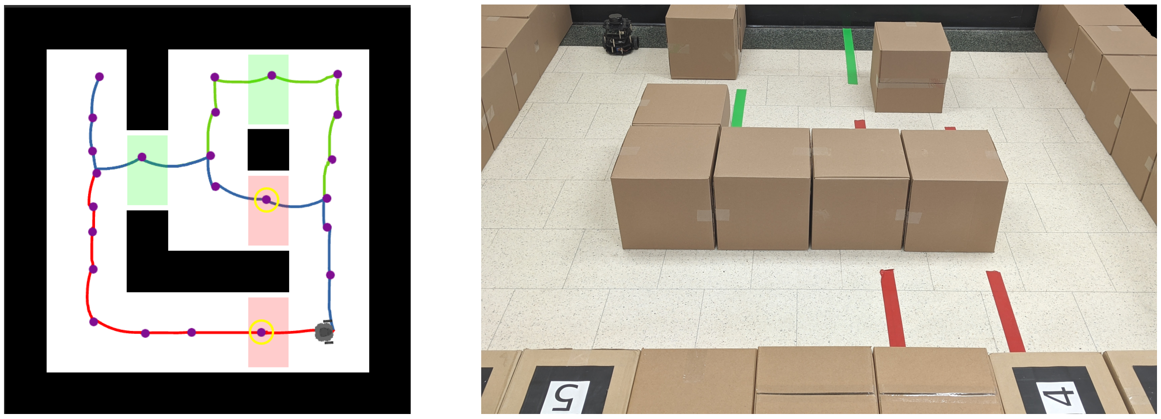


Figure 4.6: *Left*: The three paths taken by the robot. *Right*: The robot and the physical domain.

of unique situations. More important are the results depicting the level-optimality over all visited states each episode; here, we see that this reaches 100%, or near 100%, for all 3 human authorities with fewer than 50 feedback signals. However, we observe an interesting phenomenon for the CAS without ISSR active; namely, we see *several* clusters of green at the far right (at which point no additional feedback signals were received). This phenomenon is due to the fact that the CAS learns to operate in l_0, that is, full human control, in a large portion of the statespace because it cannot properly discriminate the feedback received from the human conditioned on features in the inactive feature space, which is correct for certain settings of these features (which, to reiterate, are set and fixed at the start of each episode), but not for others. However, because the state space is not refined enough to consider these decision boundaries, the CAS learns to operate at the incorrect level of autonomy (relative to the full feature space) in certain conditions. These results demonstrate that the ISSR method is effective at enabling a competence-aware system to improve its competence online when missing from its active feature space features used by its human authority.

Additionally, we implemented our approach on a TurtleBot3 mobile robot [91] in a small grid-like domain with two types of door depicted by the green and red tape.

The results can be seen in Figure 4.6 where we depict the three distinct paths taken by the robot over time. The *red* line represents the first path taken by the robot—the robot, knowing nothing about the differences between doors, takes the shortest path to the goal requiring it to request human aid to open a pull door. The *blue* line represents the path taken by the robot after introducing the new feature `doortype`—the robot now travels through the first push door, knowing that the bottom pull door is not openable by it, and attempts to open door 3 as it has not determined with high confidence if `doortype` is the determining factor, or if it simply is disallowed from opening door 4 specifically. After receiving a disapproval, it goes up to door 1, a push door, which it is allowed to open. The *green* line represents the final path—the robot has now identified that `doortype` is the determining factor, and takes the path that only interacts with push doors, despite being longer than the other two.

4.4 Conclusion

Preliminary internal testing on an autonomous vehicle prototype suggests that designing a perfectly specified competence-aware system for real-world, highly-unstructured domains is a non-trivial task. Even with expert domain knowledge, an initial model may be missing features used by the human in determining their feedback for the CAS. To avoid solving this naively with the inclusion of all possible system features in the CAS's domain model (many of which would serve no functional purpose but would cause the state space to explode and render planning intractable), we devise the *iterative state space refinement* approach. Described in Algorithm 1, the approach provides a competence-aware system the means to gradually refine its state representation online, enabling it to better identify the boundaries between state-action pairs with difference competences. This ability is particularly relevant in the context of systems deployed in the real world where human feedback may be conditioned on features that are unspecified or unknown *a priori*. Such features may not impact the original stated objectives of the system, but could influence unstated human preferences, trust, safety, and social conscientiousness. We prove that, when possible, this approach is guaranteed to reach a point where all states are discriminated, and demonstrate empirically that a CAS with this approach implemented far outperforms a CAS without it when the CAS cannot properly learn from human feedback due to missing state features. In particular, the modified CAS requires both fewer total feedback signals from the human, placing less burden on the human, and is more sample efficient with the feedback it receives in learning its competence, leading to a higher level-optimality for the CAS.

CHAPTER 5

CONTEXTUAL COMPETENCE AND HETEROGENEOUS HUMAN OPERATORS

In Chapter 3, we introduced the competence-aware system framework conditioned on several strong assumptions about what we referred to as the “human authority”. Namely, (1) that there is only a single human authority; (2) that the human is perfectly consistent; and (3) that the human is perfectly safe with invariant performance. However, humans are not perfectly consistent and their performance may vary with both skill, state, and context. Consequently, in real-world domains where these assumptions do not hold, the applicability of the base CAS framework may be diminished, prompting a need for extending the framework in such a way that it can handle relaxing the stated assumptions. In fact, accounting for the human’s state and performance is particularly important in the pursuit of competence modeling given that we specifically measure competence as a function of the human-agent system as a whole, rather than simply as a function of the autonomous agent’s underlying technical capacity, and is hence intrinsically conditioned on the capacity of the agent to interact with the human and vice-versa.

Consequently, in this chapter we consider a relaxed version of the problem setting where there may be more than one human operator and where each operator maintains their own (possibly unique) internal state that affects their performance and interactions with the agent and changes stochastically over time conditioned on both the behavior of the agent as well global contextual features that are independent of the agent’s behavior [69]. Our work is motivated by insights from Costen et al. [32] who proposed what they call a *stochastic-operator semi-autonomous system* (SO-SAS) which is a direct extension of the SAS model [124] wherein each operator (assumed, but not required, by Costen et al. to all be human but one, the autonomous agent) maintains a partially observed internal state that changes stochastically according to a known Markov model. Notably, Costen et al. focused specifically on the shared-autonomy problem of selecting the optimal operator at each timestep to be in control at each timestep. However, the proposed SO-SAS model does not consider several features that are integral to the CAS framework. First, the authors do not consider multiple levels of autonomy, proposing an all-or-nothing operative control structure; second, each operator behaves according to a fixed policy when in control; and third, the reward function is independent of the operator (up to transition dynamics).

We propose an extension of the base CAS model that naturally handles the relaxed problem setting with multiple stochastic human operators and contextual competence dependence, while still maintaining the same convergence guarantees and demonstrating significant empirical improvements over the standard CAS model in the given problem setting. In fact, we show that the CoCAS not only fully generalized the CAS model, but also the SO-SAS model when each operator is fully observable.

5.1 Contextual Competence

In Chapter 3, we introduced the formal competence-aware system model for an agent that had a domain model, $\mathcal{D}$, an autonomy model, $\mathcal{A}$, and a feedback model, $\mathcal{F}$, that was based on a single human authority, $\mathcal{H}$. In this chapter, we do not make any changes to either the domain model or the autonomy model, but instead extend the feedback model in the following ways.

Let $\mathcal{H} = \{H_1, ..., H_N\}$, for $N \geq 1$, be a set of human operators. Each human operator, H_i, is parameterized at each timestep by an operator state θ_i^j drawn from a set of possible operator states $\Theta_{H_i} = \{\theta_i^1, ..., \theta_i^m\}$, with $m \geq 1$. In general, each Θ_{H_i} may be unique; however, for the sake of notational clarity, we will henceforth simply write Θ_H by observing that we can define $\Theta_H = \bigcup_{i=1:N} \Theta_{H_i}$ in the case where the Θ_{H_i} are non-identical. Here, Θ_H encodes local information about each individual human operator, such as whether they are tired or active, their current capabilities and performance as a tele-operator or authority, or the quality of their connection to the autonomous agent when tele-operating.

In addition to the the human state information, we augment the feedback model with an additional parameter $\theta_C \in \Theta_C$ which encodes additional *contextual* information about the world that is relevant to the operators' feedback model. In general, Θ_C represents global information about the world or the set of operators as a whole, as well as other external contextual information that may impact the overall system's competence. For instance, the current active load by a fleet of competence-aware systems on the active human operators, which operators are busy or available, the global weather or environmental conditions, etc.

Example 6. A fleet of semi-autonomous vehicles are used to taxi passengers around a city and are supported by a group of remote human operators. The semi-autonomous vehicles may each require tele-operative support in different capacities and for different durations to handle challenging, hazardous, or socially ambiguous situations that constrain the capacity of their autonomous behavior. For example, fully tele-operative control by a remote human may be necessary to complete a challenging maneuver such as overtaking a lane obstruction in a narrow road, or to to approve re-routing when the passenger does not appear to be around to be picked up. However, each tele-operator's availability and internal state will vary with time and context such as their current workload or time spent working, and their feedback may vary with global contextual information such as the weather or time of day which may impact their trust in the vehicle and risk-tolerance.

5.1.1 Extending the Feedback Model

We represent the *contextual operator model*, $\mathcal{F}^+$, by the tuple $\langle S_H, T_H, \Sigma, \lambda, \rho, \tau_{\mathcal{H}} \rangle$ where

- $S_H = \Theta_H^N \times \Theta_C$ is a finite set of contextual-operator states represented by a human-operator state vector, $\langle \theta_H^1, ..., \theta_H^N \rangle \in \Theta_H^N$, representing the state of each human operator, and a contextual state parameter, $\theta_C \in \Theta_C$, representing any additional global information,

- $T_H : S_H \times S \times \mathcal{L} \times A \times \mathcal{L} \times S_H \to [0, 1]$ is a transition function representing the probability of transitioning from contextual-operator state s_H to contextual-operator state s'_H conditioned on taking action $a \in A$ at level $l \in \mathcal{L}$ in state $s \in S$ having just operated in level $l' \in \mathcal{L}$,

- $\Sigma = \{\sigma_1, ..., \sigma_k\}$ is a finite set of feedback signals that can be received from any of the N operators,

- $\lambda : S_H \times S \times \mathcal{L} \times A \times \mathcal{L} \to \Delta^{|\Sigma|}$ is a feedback model that represents that probability distribution over feedback signals that can be received when performing action $a \in A$ at level $l \in \mathcal{L}$ in state $s \in S$ having just operated in level $l' \in \mathcal{L}$, conditioned on contextual-operator state s_H,

- $\rho : S_H \times S \times \mathcal{L} \times A \times \mathcal{L} \to \mathbb{R}^+$ is a *human cost function* that represents the non-negative cost to the current active human operator of performing action $a \in A$ at level $l \in \mathcal{L}$ in state $s \in S$ having just operated in level $l' \in \mathcal{L}$ conditioned on the contextual-operator state $s_H \in S_H$, and

- $\tau_{\mathcal{H}} : S_H \times S \times A \times S \to [0, 1]$ is the *human transition function* that represents the probability of transitioning from domain state $s \in S$ to CAS state $s' \in S$ when performing domain action $a \in A$ with the current human operator in control of the system, conditioned on the current contextual-operator state $s_H \in S_H$.

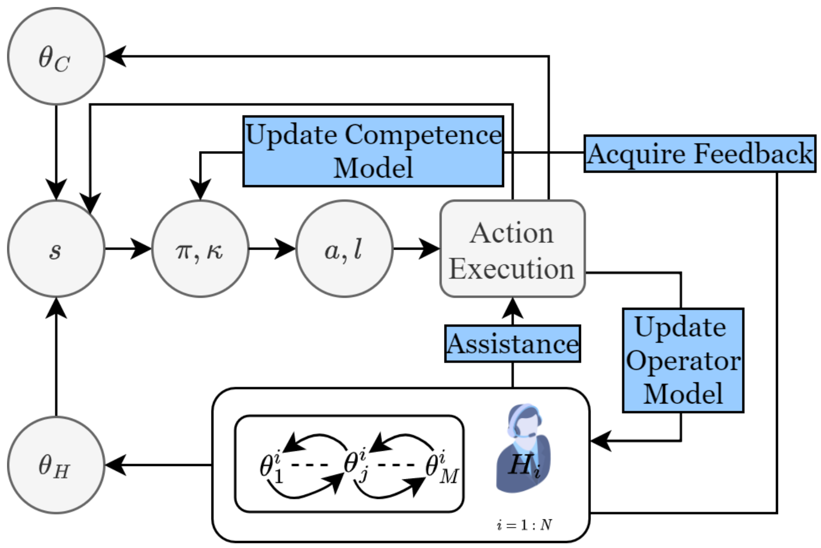


Figure 5.1: Illustration of the information flow in a CoCAS.

A *contextual competence-aware system* (CoCAS) is the natural extension to the CAS model as defined in Chapter 3, where the feedback model, $\mathcal{F}$, is replaced with the contextual operator model, $\mathcal{F}^+$. There are several notable extensions in the representational power of the CoCAS model over the base CAS model. Unlike the CAS model which assumes a single, fixed, stationary human who is always available, the CoCAS model allows for us to represent multiple, stochastic operators, with dynamic control states when computing the competence of the overall system. By incorporating the contextual-operator states into the state representation, a CoCAS can also proactively plan in a way that globally optimizes (in expectation) its interactions with each operator over the course of executing its task. We formally define a CoCAS below.

Definition 25. A *contextual competence-aware system* (CoCAS), $\mathcal{C}$, is represented by the tuple $\langle \overline{S}, \overline{A}, \overline{T}, \overline{R}, \gamma \rangle$, where

- $\overline{S} = S_H \times S \times \mathcal{L}$ is a set of factored states such that S is the set of domain states and $\mathcal{L}$ is the levels of autonomy,
- $\overline{A} = A \times \mathcal{L}$ is a set of factored actions such that A is the set of domain actions and $\mathcal{L}$ is the levels of autonomy,

- $\overline{T} : \overline{S} \times \overline{A} \times \overline{S} \rightarrow [0,1]$ is a transition function representing the probability of transitioning to state $\overline{s}' \in \overline{S}$ having taken action $\overline{a} \in \overline{A}$ in state $\overline{s} \in \overline{S}$,
- $\overline{R} : \overline{S} \times \overline{A} \rightarrow \mathbb{R}$ is a reward function representing the immediate reward for taking action $\overline{a} \in \overline{A}$ in state $\overline{s} \in \overline{S}$, and
- $\gamma \in (0,1]$ is the discount factor.

Example 7. For example, the transition function from Chapter 3, Example 3 would be modified in the following way:

$$\overline{T}(\overline{s}, \overline{a}, \overline{s}') = \begin{cases} \tau_{\mathcal{H}}((s, s_H), a, s'), & \text{if } l = l_0, \\ \Big(\lambda(\oplus|\overline{s}, \overline{a}) T(s, a, s') + \lambda(\ominus)[s = s']\Big) * \mathbf{T_H}(\bar{\mathbf{s}}, \bar{\mathbf{a}}, \mathbf{s'_H}), & \text{if } l = l_1, \\ \Big(\lambda(\emptyset|\overline{s}, \overline{a}) T(s, a, s') + \lambda(\oslash|\overline{s}, \overline{a}) \tau_{\mathcal{H}}(s, a, s')\Big) * \mathbf{T_H}(\bar{\mathbf{s}}, \bar{\mathbf{a}}, \mathbf{s'_H}), & \text{if } l = l_2, \\ T(s, a, s') * \mathbf{T_H}(\bar{\mathbf{s}}, \bar{\mathbf{a}}, \mathbf{s'_H}), & \text{if } l = l_3, \end{cases}$$

where bold text here is used for emphasis and clarity.

5.1.2 Properties

Definition 26. Let $\lambda^{\mathcal{H}} : \overline{S} \times \overline{A} \rightarrow \Delta^{|\Sigma|}$ be the stationary distribution of feedback signals for the N operators, $\mathcal{H}$. The **competence** of CoCAS $\mathcal{S}$, denoted $\chi_{\mathcal{S}}$, is a mapping from $\overline{S} \times A$ to the *reward-maximizing* level of autonomy given perfect knowledge of $\lambda^{\mathcal{H}}$. Formally:

$$\chi_{\mathcal{S}}(\overline{s}, a) = \operatorname*{argmax}_{l \in \mathcal{L}} q^*(\overline{s}, (a, l); \lambda^{\mathcal{H}}) \tag{5.1}$$

where $q^*(\overline{s}, (a, l); \lambda^{\mathcal{H}})$ is the cumulative expected reward under the optimal policy π^* when taking action $\overline{a} = (a, l)$ in state $\overline{s}$ conditioned on the feedback distribution $\lambda^{\mathcal{H}}$.

While the CoCAS model offers additional representational power, it is important to show that the same theoretical properties of a normal CAS model hold for a CoCAS as well. Indeed, the key observation to show that the convergence guarantees do hold for a CoCAS is that the additional model information (i.e., the addition of multiple stochastic operators who may each have their own feedback profiles and cost functions) is captured by S_H, which is included in the state representation $\overline{S}$. Hence, under the assumption that every $(\overline{s}, \overline{a}) \in \overline{S} \times \overline{A}$ (with the redefined CoCAS sets) is visited sufficiently in the limit, the system will still learn the optimal level of autonomy for each operator in each contextual-operator state.

Proposition 5. Let $\mathcal{S}$ be a CoCAS and let $\lambda_t^{\overline{s},a}$ be the random variable representing $\lambda(\overline{s}, a)$ after having received t feedback signals for $(\overline{s}, a)$ where each signal is sampled from the true distribution $\lambda^{\mathcal{H}}(\overline{s}, a)$. As $t \to \infty$, if no $(\overline{s}, a)$ is starved, $\mathcal{S}$ will converge to λ-stationarity (see Definition 18).

Proof. Let the expected value of sample information (EVSI) on $\sigma \in \Sigma$ for $(\overline{s}, a)$ to be defined as in Definition 17. Fix $(\overline{s}, a)$. As each feedback signal for $(\overline{s}, a)$ is drawn from the true distribution $\lambda_{\mathcal{H}}(\overline{s}, a)$ i.i.d, then by a straightforward application of the law of large numbers, it follows that the sequence $\{\lambda_t^{(\overline{s},a)}\}$ will converge in distribution to $\lambda_{\mathcal{H}}^{(\overline{s},a)} = \mathbb{E}[\lambda_{\mathcal{H}}(\overline{s}, a)]$. Hence, it follows that $\lim_{t\to\infty} Pr[|\lambda_t^{\overline{s},a} - \lambda_{\mathcal{H}}^{\overline{s},a}]| > \epsilon] = 0 \ \forall \epsilon > 0$. Consequently, as $t \to \infty$, the probability that $\lambda_t^{\overline{s},a} = \lambda_{\mathcal{H}}^{\overline{s},a}$ defines a Dirac delta function with point mass centered at $\lambda_{\mathcal{H}}^{\overline{s},a}$. The rest of the proof follows the proof of Proposition 1. $\square$

Proposition 6. Let $\mathcal{S}$ be a CoCAS that follows any level-exploration strategy that ensures a non-zero probability that all reachable levels of autonomy are explored and switches to exploitation once λ-stationarity has been reached, and where $\chi_{\mathcal{S}}(\overline{s}, a)$ is reachable from κ_0 for all $(\overline{s}, a) \in \overline{S} \times A$. Then if no $(\overline{s}, a)$ is starved, as $t \to \infty$, $\mathcal{S}$ will converge to level-optimality.

Proof. The proof follows that of the proof of Theorem 2. By Proposition 5, $\mathcal{S}$ will reach λ-stationarity, which ensures that for any $\overline{s} \in \overline{S}$ and action $a \in A$ the optimal level of autonomy $l^* \in \mathcal{L}$ is known in the limit, and hence we can determine in the limit the competence $\chi(\overline{s}, a)$ for all $\overline{s} \in \overline{S}$ and $a \in A$ so long as that level is explorable by the system (that is, both allowed under the human's true autonomy profile and reachable from the initial autonomy profile) with non-zero probability, which is true by assumption. □

Remark 2. Any CAS $\mathcal{S}$ can be modeled as a COCAS $\mathcal{C}$.

This remark is straightforward to observe simply based on the definition of the contextual operator model, $\mathcal{F}^*$, by fixing $|\Theta_{\mathcal{H}}| = 1$, $|\Theta_C| = 1$, and $N = 1$.

Proposition 7. Any fully-observable SO-SAS can be modeled as a CoCAS.

Proof. Let $M = \langle S^E, X, s_0, b_0, A, \Omega, T, O, R, \gamma \rangle$ be a fully- observable SO-SAS, i.e., so $\Omega = S^E \times X$ where S^E is the environment state space and $X = X_1 \times \cdots \times X_N$ is the operator profile state space, and the rest is as defined in [32]. Now, let $\mathcal{L} = \{l_1, ..., l_N\}$ be the levels of autonomy where l_i represents full control by the i^{th} operator, where the first operator is the autonomous agent, and let the CoCAS domain action set be $\emptyset$ such that $\overline{A} = \mathcal{L} \cong A$ i.e., the selection of which operator to be in control. Let $S = S^E$ and $S_H = X$, so that $\overline{S} = S_H \times S \times \mathcal{L} = X \times S^E \times \mathcal{L}$, and assume by construction that $\mathcal{L}$ does not impact either $\overline{T}$ or $\overline{R}$, so that we can simplify that $\overline{S} = S_H \times S$. Then we have that $\overline{T} : \overline{S} \times \overline{A} \to \Delta^{\overline{S}} \cong T : (S^E \times X) \times A \to \Delta^{(S^E \times X)} \cong T : (S^E \times X) \times A \times (S^E \times X) \to [0, 1]$, and $\overline{R} : \overline{S} \times \overline{A} \to \mathbb{R} \cong R : (S^E \times X) \times A \to \mathbb{R}$. □

5.2 Evaluation

To test the contextual competence-aware system, we implement anmodified version of the autonomous vehicle navigation domain used earlier in this thesis. In this domain, an autonomous vehicle (AV) is tasked with picking up and dropping

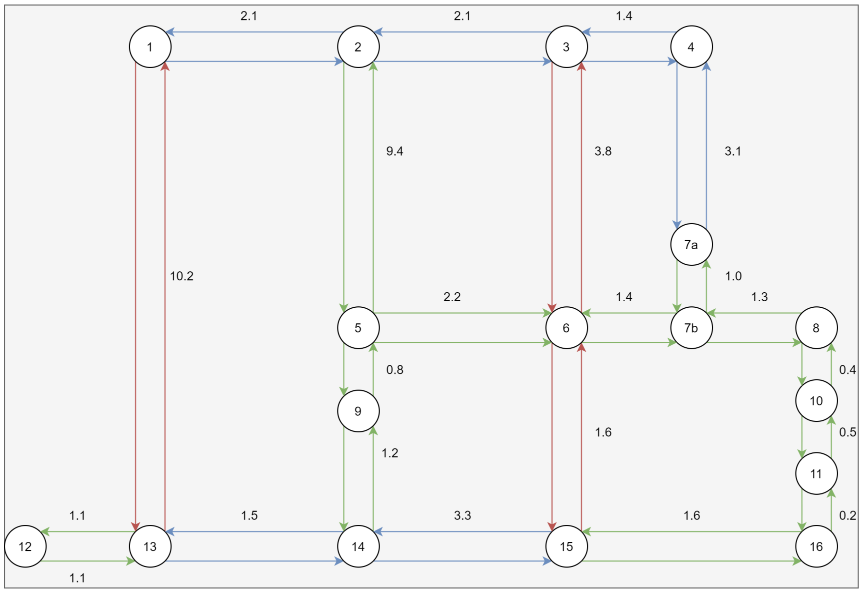


Figure 5.2: Illustration of the abstracted map for the autonomous vehicle delivery service domain.

off goods around an area, where its objective is to reach its destination in the cost cost-effective manner. There are two types of state: node states representing intersections, and edge states representing road segments. Node states are represented by the tuple $\langle ID, p, o, v, \theta \rangle$, and edge states by the tuple $\langle u, v, \theta, o, l, r \rangle$. Here, ID is the ID of the node, p is 1 if there are pedestrians or else 0, v is the number of relevant other vehicles, θ is the AV's heading, u and v are the start and end node IDs, o is 1 if there is an obstruction in the AV's lane or else 0, l is the number of lanes on the road, and r denotes road restrictions if any.

The AV drives autonomously (l_0) but may seek assistance from human teleoperators: requesting approval for certain actions (l_1) or requesting full tele-operative control of the vehicle (l_0) (which may be denied). We model the existence of two human operators: a local operator, $\mathcal{H}_l$, and a global operator, $\mathcal{H}_g$ The local operator's connection is always stable allowing them to provide support for the AV in any capacity requested; however the local operator may become busy assisting another vehicle in the fleet when the demand is high. A global operator, however, is always available, but their connection may be either stable or unstable. We additionally consider contextual information, $\theta_C = \langle d, t, w \rangle \in \Theta_C$ where d is the current AV demand, t is the time of day, and w is the current weather condition. Hence we get that $S_H = \{\theta_{busy}, \theta_{active}\} \times \{\theta_{stable}, \theta_{unstable}\} \times \{\mathcal{H}_l, \mathcal{H}_g\} \times \Theta_C$. When stable, the global operator can provide verification for actions with consistency 0.8 consistency and full tele-operation in certain non-challenging conditions; when unstable, the global operator can only provide verification for actions with consistency 0.7. We also require that the AV transfer control to a tele-operator when driving on roads that are demarcated as *pedestrian zones* or *school zones.*

We define T_H as follows: $\mathcal{H}_l$ can become busy or available with probabilities $1 - 0.5^d$ and 0.5^d at each step respectively. Note that, when there is zero demand, $\mathcal{H}_l$ is always available. At each step, $\mathcal{H}_g$'s connection quality changes with probability 0.25 and otherwise remains the same. We model the human cost function, ρ, to be based off of the expected opportunity cost incurred when the operator helps the AV and hence is unavailable to help another vehicle in the fleet. We apply this cost only to the local operator whose capacity is limited, and scale by the current fleet demand, d, but apply no such cost to the global operator, as we assume there is a sufficient supply to cover all demand at the lowered level of assistance. However, there is always a small baseline cost of requesting a transfer of control.

Results

In our experiments, we simulate the performance of five different models to compare as baselines against the CoCAS model: (1) *Operator*, which allows for no autonomy throughout execution and is instead performed entirely by a human operator who can change during the task's execution; (2) R-SO-SAS, which randomly selects the operator (one of which is the autonomous agent) uniformly at each time step, where each operator acts on a fixed, preo-computed policy; (3) SO-SAS, which follows the

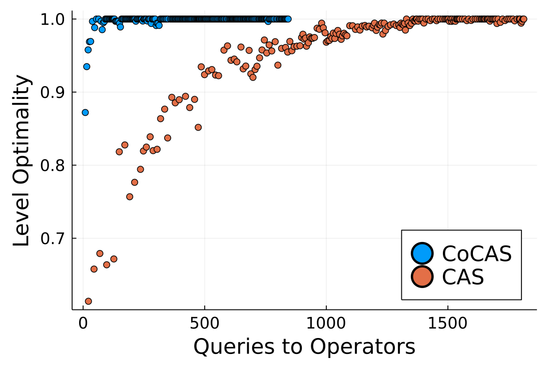
	Operator	R-SO-SAS	SO-SAS	C-SO-SAS	CAS	CoCAS
Route 1	67.09(±11.15)	52.20(±43.73)	42.08(±6.20)	44.51(±4.59)	41.40(±6.62)	**39.71**(±**4.00**)
Route 2	207.31(±23.59)	223.07(±54.83)	168.76(±21.42)	155.65(±13.72)	176.00(±25.62)	**142.85**(±**10.81**)
Route 3	118.86(±17.69)	130.96(±45.69)	87.71(±9.51)	84.61(±8.52)	84.70(±77.90)	**77.93**(±**7.56**)
Route 4	169.39(±19.96)	166.83(±42.61)	127.44(±10.09)	118.18(±7.67)	129.00(±15.30)	**112.87**(±**7.39**)
Route 5	110.75(±17.83)	115.43(±42.86)	76.06(±9.32)	75.63(±6.35)	69.40(±4.79)	**68.44**(±**3.89**)
Route 6	182.83(±22.87)	202.17(±59.32)	136.38(±12.29)	127.65(±9.87)	132.00(±11.98)	**119.98**(±**9.71**)
Route 7	160.37(±21.78)	191.84(±62.82)	111.31(±12.42)	105.61(±11.80)	121.40(±24.77)	**99.80**(±**9.55**)
Route 8	96.91(±16.67)	193.44(±44.71)	72.96(±10.91)	68.29(±9.75)	65.80(±**6.05**)	**61.35**(±6.85)

Table 5.1: Results Summary from AV Delivery Domain.

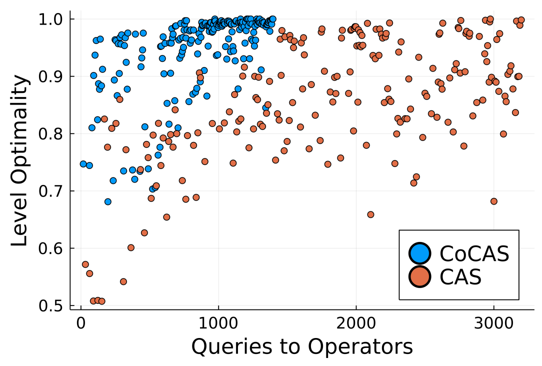
model from Costen et al. [32], except that the operator state is fully observable at each; (4) C-SO-SAS is the same as SO-SAS except that we add the cost function ρ to the planning model; (5) the standard CAS model.

Table 5.1 shows the results of our model (CoCAS) against the same 5 baseline models on the AV delivery service domain across 8 different routes. Unlike the UAV domain, this is a purely cost-minimizing domain and, in the interest of space, we only include the cumulative cost. In particular, we can see that the CoCAS outperforms all other approaches in average cost across all 8 domains, and the lowest standard deviation in all but one case as well (Route 8).

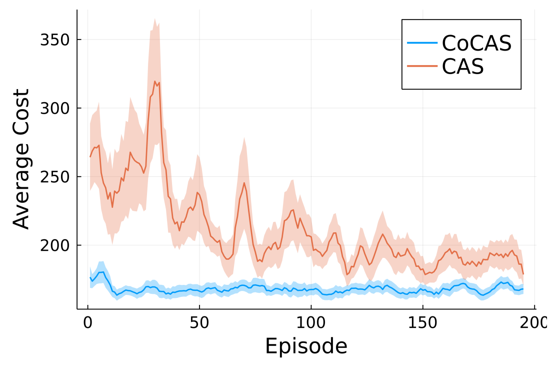
Figure 5.3 depicts the results from the learning simulation for both the CAS and CoCAS over 200 episodes. In the left column of figures, the contextual information for time of day and the weather conditions is fixed throughout all 200 episodes, whereas the latter set the contextual information varies across episodes; in both cases, the current AV demand changes dynamically during the episode at each time step. Figures 5.3a and 5.3b depicts the level optimality of *actions taken by the system* as a function of the number of queries made to the human. When the contextual information is fixed, we see that the CAS is able to converge to 100% level-optimality, although at a much slower rate than the COCAS, which reaches it very efficiently.



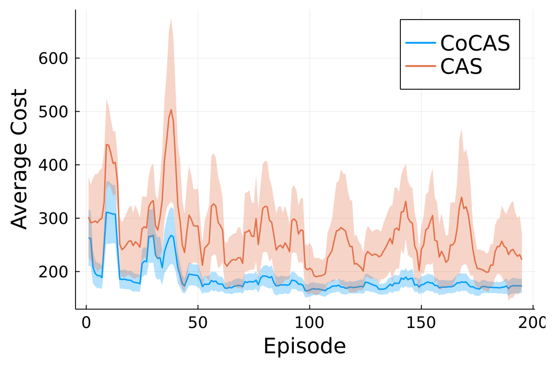
(a)



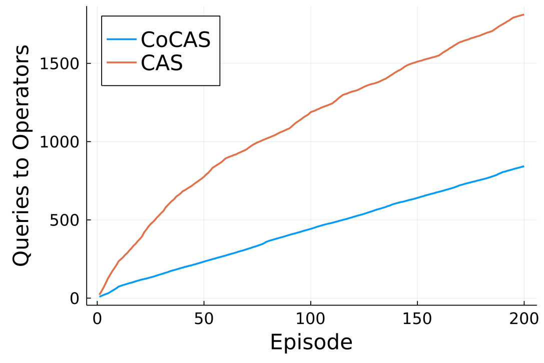
(b)



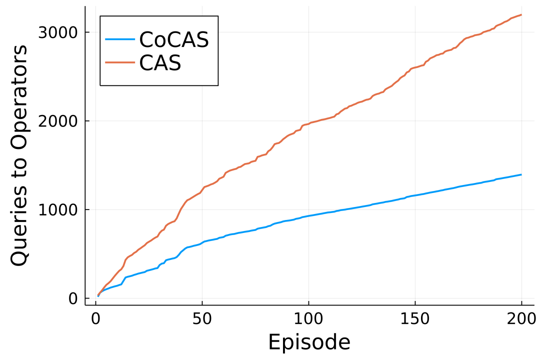
(c)



(d)



(e)



(f)

Figure 5.3: Comparison of the performance of the CAS and CoCAS over 200 episodes in the AV Delivery Service domain with fixed contextual parameters (left) and dynamic contextual parameters (right).

When it is not fixed, the CoCAS can still reach 100% level-optimality over actions taken but the level-optimality of the CAS remains highly variable, and as low as 70% due to its inability to appropriately learn across unmodeled contextual information that alters the dynamics of the operators' feedback and interactions.

Figures 5.3c and 5.3d depict the mean and standard deviation of the incurred cost for both the CoCAS and CAS over the 200 episodes. Notably, the CoCAS quickly minimizes the incurred cost in both scenarios, and even in the fixed-context case where the CAS reaches 100% level-optimality in actions taken, the CoCAS is still more cost-effective in its operation. When the contextual information varies, where the CAS is unable to sufficiently learn its competence, its performance is both less cost-effective than the CoCAS and significantly more noisy. From figures 5.3e and 5.3f we see that the CAS queries the operators at roughly twice the rate as the CoCAS.

5.3 Conclusion

In this chapter, we introduce an extension of the CAS model, called a *contextual CAS*, designed to handle multiple, heterogeneous stochastic human operators and contextually-dependent competence modeling. We prove that the CoCAS is general enough to capture both the CAS model and the fully-observable SO-SAS model [32] (although the premise of each problem is distinct), but still retains the same theoretical convergence guarantees as the standard CAS model. Additionally, we evaluate the model empirically on a modified version of the autonomous vehicle navigation domain used in prior chapters, demonstrating empirically that the CoCAS model outperformed five different baseline models across all scenarios tested – the CAS model, the SO-SAS model and 2 variants, and a human-operator-only baseline. We show that not only does the CoCAS have lower average cost (and lower variance in performance in all cases but one), but also reaches a higher level-optimality more efficiently than the standard CAS model when competence is conditioned on contextual information and the operator state.

Although we focus specifically on a generalization of the *human feedback model*, by integrating contextual information and stochastic operators into the competence-modeling framework, we propose that a similar generalization could be made to the *autonomy model*, most obviously by extending constraints on autonomy to depend on contextual information and the state of the operators. Additionally, natural directions for future research include extending the CoCAS to the partially-observable setting both in the domain model as well as the operator model, learning the operator model parameters online, and formulating the fleet-wide human-to-agent matching problem to optimize fleet-wide operation.

CHAPTER 6

LEARNING COMPETENCE FROM PROACTIVE FEEDBACK

In prior chapters, we have implicitly assumed that all feedback generated by the human from which a competence-aware system learns its competence model is generated *reactively* by the human. By reactive, we mean that the feedback is conditioned on (or caused by) the most recent state-action pair in the agent's trace. In fact, prior work in learning from human feedback has always assumed that feedback is generated either reactively, or retroactively [5, 110, 67, 65], but has ignored the possibility of *proactive* feedback. That is, feedback that is conditioned on the possibility of inferred future behavior of the agent given the human's understanding of the agent's behavior.

However, growing evidence suggests that humans' cognitive control processes are generally either reactive in response to rapid or unexpected changes in their external contexts, or proactive as a means of strategically optimizing behavior resulting from anticipated goal-relevant interference that they aim to ameliorate before occurring [18, 3]. Consequently, we believe that it is important to consider human interventions that can be proactive to ensure that the agent learns the intended behavior.

In this chapter, we consider a simplified version of the CAS model where there are only two levels of autonomy: *no autonomy* and *full autonomy.* As before, we assume that the CAS has a known domain model with a well-specified reward function that captures the nominal task objective, but no *a priori* knowledge of its autonomy profile, which represents the state-action pairs that the CAS is disallowed from performing autonomously. This can be viewed more simply as a set of *constraints* that may capture more nuanced aspects of the domain such as preferences or safety concerns that may be challenging to specify exactly when designing the system.

We propose a constraint learning method for the agent that infers the true constraint set from human feedback in the form of sparse interventions, rather than more costly, verbose forms of feedback like full or partial demonstrations. We consider this supervise-and-intervene setting as interventions are a natural teaching framework for a human, are easy to provide, and provide useful information to the agent as an indication of a potential constraint violation assuming a reasonable level of understanding by the human [110].

We consider interventions rather than demonstrations because learning from demonstrations limits the applicable set of domains to those where the human a human can manually generate demonstrations for the robot. Furthermore, even when demonstrations are available, it is possible for the agent to incorrectly infer the intended behavior of the demonstration provider [5]. Likewise, a query-for-information framework may be inefficient and onerous to the human if the agent frequently queries for the existence of a non-existent constraint, or unsafe if the agent fails to query in the presence of possible constraint violations.

The proposed approach makes minimal assumptions about the human, but enables the agent to iteratively learn a constraint model and intervention model from intervention data using a data aggregation approach [95]. Additionally, we use uncertainty-based incentives to balance exploration and exploitation during training time to improve convergence. We show that our method enables an agent in a domain with an accurate nominal task model to efficiently learn a set of unspecified constraints from sparse, proactive feedback that generalizes to novel environments in the domain. Furthermore, we show that failing to account for proactivity in human feedback can lead an agent to infer an incorrect constraint set and perform poorly in new environments.

6.1 Constraint Learning Formulation

Let $M = \langle S, A, T, R \rangle$ be an MDP that represents the nominal domain of the agent, which, for reward R, induces what we refer to as the *nominal objective* (o_n) that the agent aims to maximize. In other words, the nominal objective aims to find the policy, π that maximizes the cumulative reward under R,

$$\underset{\pi \in \Pi}{\operatorname{argmax}} \mathbb{E}\Big[\sum_{s \sim d_\pi} R(s, \pi(s))\Big]. \tag{6.1}$$

However, there exists a set, $C \subset S \times A$, of *constraints* that is unknown to the agent *a priori* that describe the state-action pairs that the agent is disallowed from performing (or, more generally, are undesirable). The constraint set induces an additional objective (o_c), which aims to find the policy that minimizes the expected number of constraint violations,

$$\underset{\pi \in \Pi}{\operatorname{argmin}} \mathbb{E}\Big[\sum_{s \sim d_\pi} C(s, \pi(s))\Big], \tag{6.2}$$

where, for clarity, we let $C : (s, a) \to \mathbb{I}[(s, a) \in C]$.

Because the constraint set is unknown *a priori*, the agent must learn the constraint set from proactive feedback provided by a human expert. The human is modeled by the tuple, $\mathcal{H} = \langle \epsilon, h, \tau \rangle$ where:

- $\epsilon \in (0, 1]$ models the degree of the human's understanding of the agent's behavior,
- $h \in \mathbb{Z}^+$ is a temporal horizon, and
- β is an intervention temperature parameter.

Under ϵ, we can compute a policy-like function that we called the *corrupted observer policy*, $\hat{\pi}$, which models the human's lookahead belief over the agent's behavior $i \geq 1$ steps into the future from state s_t, as

$$\hat{\pi}(s_{t+i}) = \begin{cases} \pi(s_{t+i}) & \text{w.p. } 1-\epsilon \\ \sim U(A \setminus \{\pi(s_{t+1})\}) & \text{w.p. } \epsilon \end{cases} \tag{6.3}$$

For the current time step, where $i = 0$, we assume that the human knows exactly the the action that the agent is performing, ensuring that they will be able to intervene in any constraint violation while supervising the agent. Intuitively, this represents the human's error in their model of the agent's behavior.

Given $\mathcal{H}$ and $\hat{\pi}$, we define the human intervention function $I(s_t|h, \beta, \hat{\pi}, C) \rightarrow \{0, 1\}$, which represents the binary decision of the human to override the system, or not, in state s_t given horizon h, temperature parameter β, corrupted observer policy $\hat{\pi}$ and the true constraint set C. In general, it is expected that it will be proportional to the expected number of constraint violations in the h-term future horizon starting at state s_t given policy $\hat{\pi}$:

$$I(s_t|h, \beta, \hat{\pi}, C) \propto \mathbb{E}_{s \sim \tau(s_t, \hat{\pi}, h)} \Big[C(s, \hat{\pi}(s)) \Big] \tag{6.4}$$

where $\tau(s_t, \hat{\pi}, h)$ represents the full set of h-step trajectories under $\hat{\pi}$ starting from state s_t.

More generally, we may only have access to some distribution over β, D_β, and distribution over h, D_h; in which case, we get that

$$I(s_t|D_h, D_\beta, \hat{\pi}, C) = \int_{\beta \sim D_\beta} \int_{h \sim D_h} I(s_t|h, \beta, \hat{\pi}). \tag{6.5}$$

While the agent is still learning C, it maintains an estimate, $\hat{C} : S \times A \rightarrow [0, 1]$, which represents the current likelihood estimate of $(s, a) \notin C$. Given $\hat{C}$, we can compute an approximate intervention function,

$$\hat{I}(s_t|h, \beta, \hat{\pi}, \hat{C}) \propto \mathbb{E}_{s \sim \tau(s_t, \hat{\pi}, h)} \left[\hat{C}(s, \hat{\pi}(s)) \right] \tag{6.6}$$

and, correspondingly,

$$\hat{I}(s_t|D_h, D_\beta, \hat{\pi}, C) = \int_{\beta \sim D_\beta} \int_{h \sim D_h} \hat{I}(s_t|h, \beta, \hat{\pi}). \tag{6.7}$$

6.1.1 Learning Setting

Unlike prior chapters where all learning was performed online, we consider in this chapter a *train-then-deploy* setting where the agent has a fixed amount of training time available during which it must learn the constraint set to the best of its ability. Once the training time is over, the agent is deployed into an unsupervised setting where it must optimize its nominal task while adhering to its learned constraint set. Extending the proactive feedback model to the learning-on-the-go setting introduces challenging complexities that are beyond the scope of this chapter, but a discussion of them can be found in Section 6.4.

Training

During training, we add a secondary reward, E, for interventions that is based off of the information to provide an exploration utility to the agent for learning more of its constraint set. Consequently, during training we model the problem as a *multi-objective MDP* (see Definition 3) where the primary objective is the agent's nominal task objective, and the secondary objective is to learn its constraint set. In general, one can include a penalty, I, for interventions as well; however, we associate no cost to an intervention during training as it is the job of the human to supervise and train the agent. Formally, we define $M^T = \langle S, A, T, \begin{bmatrix} R & I & E \end{bmatrix}^T, \begin{bmatrix} w_R & w_I & w_E \end{bmatrix} \rangle$.

Deployment

During deployment, the agent does not have a human that it can rely on to provide it safety. Hence, we model the problem as an *lexicographic MDP* (see Definition 4) where the primary objective is to minimize the probability of violating a constraint, and the secondary objective is to maximize the nominal task reward. Formally, define $M^D = \langle S, A, T, [-C \quad R]^T, \langle \delta \rangle, [o_C, o_R] \rangle$ where $\delta \in \mathbb{R}^+$ is the maximum slack allowed from the minimal probability of constraint violation.

6.2 Methodology

In this section, we describe our proposed method for learning constraints from spare binary feedback signals in the form of proactive interventions. We start by describing the agent's training loop using a fixed known horizon $\mathbf{h}$ for ease of understanding. Later, we relax this requirement and provide a more general solution.

6.2.1 Training Loop

The algorithm starts by initializing the constraint model, C_{θ_1}, and the intervention model, I_{θ_2}, randomly (line $1-2$). Here, models are represented by two neural networks parameterized by θ_1 and θ_2 respectively. Next, the agent's policy, π_0, is initialized using the policy under the nominal reward model R (line 3). Lastly, the dataset $\mathcal{D}$ that is used for training C_{θ_1} and I_{θ_2} is initialized to $\emptyset$. After initialization, C_{θ_1} and I_{θ_2} are iteratively updated through an interactive learning process (line $5 - 12$), as detailed next.

The i-th iteration of the training loop starts by collecting a set of feedback, $\mathcal{F}_i$, using human supervision and the agent's current policy π (line 6). Feedback is provided as sparse, binary signals, which come in the form $\langle s, f \rangle$, where the state s is

```
Algorithm 2: Model-Based Constraint Learning from Proactive Feedback
    Input: Domain Model $D$, Horizon $h$, Sample $K$, Reward Weight $\mathbf{w}$
    Result: Constraint Model $C_{\theta_1}$
1 Initialize Constraint Model $C_{\theta_1}$
2 Initialize Intervention Model $I_{\theta_2}$
3 $\pi_0 \leftarrow ComputePolicy(D/R, R)$
4 $\mathcal{D} \leftarrow \emptyset$
5 for $i = 0 : \infty$ do
6     $\mathcal{F}_i \leftarrow CollectInterventionFeedback(\pi_i, K)$
7     $\mathcal{D} \leftarrow D \cup (\mathcal{F}_i, \pi_i)$
8     $Train(C_{\theta_1}, I_{\theta_2}; \mathcal{D}, h)$
9     $R^T \leftarrow GetTrainingReward(D, \mathbf{w}, C_{\theta_1})$
10    $\pi_{i+1} \leftarrow CalculatePolicy(D/R, R^T)$
11    if Termination Condition is Met
12        BREAK
13 return $C_{\theta_1}$
```

labeled by $f \in \{true, false\}$ indicating an intervention, or no intervention, in state s respectively. The set of feedback signals, $\mathcal{F}_i$, along with current policy, π_i, are added to the existing dataset $\mathcal{D}$ (line 7). $\mathcal{D}$ is then used to jointly train C_{θ_1} and I_{θ_2} using gradient-based optimization (line 8) given the horizon h. Based on the updated C_{θ_1}, a new training reward R^T is calculated using the nominal reward R, the constraint set C, the information-theoretic exploration bonus E, and the linearization weight $\mathbf{w}$ (line 9). Finally, the policy is updated using the new training reward R^T (line 10). The training loop is continued until terminating conditions are met. There are many possible choices for terminating conditions including using a fixed number of iterations, the total number of interventions during feedback collection, or the amount of change in constraint set prediction. Finally, the learned constraint model, C_{θ_1}, is returned.

6.2.2 Constraint Learning

Our core assumption is that the probability of a human intervention in a state s_j is a function of the expected number of constraint violations in the future from the current state s_j. We define this value

$$CV^h_{\pi_i}(s_j) = \mathbb{E}_{\tau\sim\pi_i(s_j)}\left[\sum_{t=0,(s_t,a_t)\in\tau}^{t=h} C(s_t, a_t)\right]. \tag{6.8}$$

To estimate this value, we learn a constraint model, C_{θ_1}, parameterized by θ_1, under which we estimate the probability of a state-action pair (s_j, a_j) being constrained as

$$C^{a_j}_{s_j} = P((s_j, a_j) \in C) \approx \sigma(C_{\theta_1}(s_j, a_j)) \tag{6.9}$$

where $\sigma(\cdot)$ is n sigmoid function. Hence, $CV^h_{\pi_i}(\cdot)$ can be estimated as follows:

$$CV^h_{\pi_i}(s_j) \approx \mathbb{E}_{\tau\sim\pi_i(s)}\left[\sum_{t=0,(s_t,a_t)\in\tau}^{t=h} C^{a_t}_{s_t}\right]. \tag{6.10}$$

Next, we use this value to estimate the probability of intervention at state s_j using a learned function, I_{θ_2}, parameterized by θ_2; specifically:

$$\hat{f}_j = P(f_j = 1|s_j) \approx \sigma(I_{\theta_2}(CV^h_{\pi_i}(s_j))). \tag{6.11}$$

To optimize both θ_1 and θ_2 we use two different loss functions. First, we use a binary cross-entropy loss to train both $C_{\theta 1}$ and $I_{\theta 2}$ using the intervention feedback:

$$\mathcal{L}_f(\theta_1, \theta_2) = -\frac{1}{|D|}\sum_{\langle s_j,f_j\rangle\in D} f_j \log(\hat{f}_j) + (1 - f_j)\log(1 - \hat{f}_j). \tag{6.12}$$

However, as we do not have a ground truth label for constraint classification, in order to ensure that C_θ learns the correct representation, we want to enforce the following constraint:

$$\mathop{\mathbb{E}}_{\langle s_j, f_j\rangle \in D_1}[CV^h_{\pi_i}(s_j)] \quad > \mathop{\mathbb{E}}_{\langle s_j, f_j\rangle \in D_0}[CV^h_{\pi_i}(s_j)], \tag{6.13}$$

where, $D_1 = \{\langle s_j, f_j\rangle \in D | f_j = 1\}$ and $D_0 = \{\langle s_j, f_j\rangle \in D | f_j = 0\}$. Generally, we expect to see more constraint violations after an intervention state. This constraint can naturally be enforced by a negative log_softmax loss. Specifically,

$$\mathcal{L}_c(\theta_1) = -\log\left(\frac{exp(J_1)}{exp(J_1) + exp(J_0)}\right) \tag{6.14}$$

where

$$J_1 = \mathop{\mathbb{E}}_{\langle s_j, f_j\rangle \in D_1}[CV^h_{\pi_i}(s_j)] \tag{6.15}$$

and

$$J_0 = \mathop{\mathbb{E}}_{\langle s_j, f_j\rangle \in D_0}[CV^h_{\pi_i}(s_j)]. \tag{6.16}$$

Intuitively, this loss goes down as I_1 gets larger than I_0.

6.2.3 Ensemble Learning

In order to get a better estimate of the uncertainty of our model, we train an ensemble of multiple models for the constraint classification task [98]. To train each model in the ensemble, in addition to sampling different subsets of the dataset, we also use a different horizon, h, sampled from the distribution $\mathcal{D}_h$. We use a weighted voting scheme to estimate the ensemble. Specifically, we calculate the probability of a constraint violation, $C^{a_j}_{s_j}$, as

$$C^{a_j}_{s_j} = \frac{\sum_k w_k \sigma(C_{\theta^k_1}(s_j, a_j))}{\sum_k w_k}, \tag{6.17}$$

where $w_k \propto Accuracy(C_{\theta_1^k})$. Intuitively this puts more weight on models in the ensembles that better explain the observed data, and also reduces the reliance on knowing the exact horizon h.

Finally, we use an uncertainty measurement to improve constraint learning during the training phase. Specifically, we define an information-theoretic exploration bonus, $E(s_j, a_j)$, as

$$E_{s_j}^{a_j} = E(s_j, a_j) = \mathbf{H}(P(constraint|(s_j, a_j))). \tag{6.18}$$

where $\mathbf{H}$ is the Shannon entropy. The purpose of this exploration bonus is to encourage the agent to explore parts of the state-action space where the agent is uncertain about its constraint prediction. Based on this, we construct a multi-objective reward, $R^T = \begin{bmatrix} R & \hat{I} & E \end{bmatrix}^T$, where $\hat{I}(s,a) = -\mathbb{I}[I_{\theta_2}(s,a) > \mu]$ for some classification threshold $\mu \in [0,1]$ (usually set to 0.5), and solve the problem as a MOMDP with linear weights, $\mathbf{w} = \begin{bmatrix} w_R & w_I & w_E \end{bmatrix}$. Using the weight parameters w_R and w_E we can balance between exploitation of the nominal task reward and exploration (of constraints) during training, particularly in settings where constraint learning is not the only reason for training. The weight w_I, which is assumed to be 0 in our work, can also be tuned to balance the agent's risk-aversion during training.

6.2.4 Deployment Phase

After the training phase is completed, we utilize our learned models in a deployment phase wherein we assume that there is no human that can intervene and protect the robot. We construct a multi-objective reward, $R^D = \begin{bmatrix} -\hat{C} & R \end{bmatrix}$, where $\hat{C}$ and R are as defined above (as the true constraint set is not known); note that we remove the entropy-based exploration bonus during the deployment phase as there is no longer a human who can provide intervention signals. We consider the lexicographic ordering, $\mathbf{o} = [o_C, o_R]$, aiming to optimize first the constraint-based objective, and the nominal task objective second given some slack $\delta \in \mathbb{R}^+$ on objective o_C.

Additionally, we can modify the classification threshold μ to control how conservative the agent should be during deployment about predicting possible constraints. For example, setting μ to be very small will cause the agent to avoid a state-action pair if there is even a small probability that the state-action pair is constrained. Finally, we would like to bound the worst-case performance of our model in deployment given what is learned by the learning algorithm. Below, we show that if our learned constraint set is at most α inaccurate, then the expected performance of the optimal policy computed for the learned constraint set will be at most a $\frac{1+\alpha T}{1-\alpha T}$-factor of the optimal expected value, assuming a finite horizon, T.

Theorem 3. Let $\hat{\pi}^*$ be the optimal policy given the learned constraint set $\hat{C}$ and π^* be the optimal policy given the true constraint set C, and let $T \in \mathbb{N}$ be a maximum horizon for the problem. Denote by V_c^π and $V_{\hat{c}}^\pi$ the value functions induced by the policy π under constraint sets C and $\hat{C}$ respectively for the primary objective (see Eq. 6.2) within the horizon T. If $\hat{C}$ is at least $(1-\alpha)$-accurate, i.e.,

$$\frac{\sum_{(s,a)\in S\times A}[\hat{C}(s,a) == C(s,a)]}{|S||A|} \geq 1-\alpha,$$

then $\max_{s\in S} V_c^{\hat{\pi}^*}(s) \leq \frac{1+\alpha T}{1-\alpha T} V_c^{\pi^*}(s)$.

Proof. For notational clarity, we write, e.g., V_c^π in stead of $V_c^\pi(s)$, understanding that our proof is with respect to the maximal difference in value functions over all states. First, observe that if we fix a policy π, then given that $\hat{C}$ is at most α incorrect, it follows that $|V_c^\pi - V_{\hat{c}}^\pi| \leq \alpha T V_c^\pi$ simply by the definition under Equation 6.2. Consequently, we have that

$$V_c^\pi - V_{\hat{c}}^\pi \leq \alpha T V_c^\pi \tag{6.19}$$

and

$$V_{\hat{c}}^\pi - V_c^\pi \leq \alpha T V_c^\pi \tag{6.20}$$

for any fixed policy π. We can apply Equation 6.20 to π^*, which gives us that

$$V_{\hat{c}}^{\pi^*} \leq (1 + \alpha T) V_c^{\pi^*}.$$

Additionally, by the definition of optimality (as this is a minimization problem), we know that $V_{\hat{c}}^{\hat{\pi}^*} \leq V_{\hat{c}}^{\pi^*}$. Consequently, we get that

$$V_{\hat{c}}^{\hat{\pi}^*} \leq (1 + \alpha T) V_c^{\pi^*}.$$

By applying $\hat{\pi}^*$ to Equation 6.19, we also have that

$$V_{\hat{c}}^{\hat{\pi}^*} \geq V_c^{\hat{\pi}^*} - \alpha T V_c^{\hat{\pi}^*}.$$

Finally, by connecting the inequalities, we get that

$$V_c^{\hat{\pi}^*} - \alpha T V_c^{\hat{\pi}^*} \leq (1 + \alpha T) V_c^{\pi^*} \implies V_c^{\hat{\pi}^*} \leq \frac{1 + \alpha T}{1 - \alpha T} V_c^{\pi^*},$$

which gives us the claim. □

6.3 Evaluation

We test our approach in two different simulated domains as described below. In both domains, we begin by training the agent in one environment within the domain using human intervention, before testing them in one or more different environments within the domain where there is no human to intervene.

Domain Descriptions

Boxpushing

This domain is inspired by the Box-pushing domain presented by Saisubramanian et al. [100] where a robot is asked with pushing a box from a starting position to a goal position. The nominal reward is 0 in a goal state and negative elsewhere, encouraging the agent to reach the goal as efficiently as possible. However, there are rugs on the floor that the human does not want the robot to move over while pushing boxes. Consequently, when the human believes the robot is about to move onto a rug, they intervene and relocate the robot to a location from which there is no chance of a constraint violation within their horizon. States $s \in S$ are represented by the tuple $\langle x_i, y_i, x_g, y_g, \phi \rangle$, where x_i and y_i denotes the current location of the robot, x_g and y_g denotes the robot's goal location, and ϕ denotes if there is a rug in the robot's current location. Actions $a \in A$ represent 8 directional movements. Each of the train and test environments differs in the number of rugs and their positions.

Autonomous Vehicle Navigation

In this domain, an autonomous vehicle (AV) is supervised by a human and must traverse a graph from a start node (intersection) to a goal node (intersection), modulating its speed as necessary. States $s \in S$ are represented by the tuple $\langle id, o, pr, s, x, \theta \rangle$ where id is the intersection ID, o denotes whether the intersection appears obstructed, pr denotes whether the intersection is protected for the AV, s denotes the AV's speed, x denotes the AV's position relative to the intersection, and θ denotes their current heading. Actions $a \in A$ are represented by a direction of travel and a speed modulation (acceleration, soft brake, hard brake, and nothing). The AV is initially only aware of hard legal constraints, such as stopping at an unprotected intersection with a stop sign, but not that it needs to slow down sufficiently early as it approaches an intersection (even if it is protected) if there is a potential obstruction, and cannot hard-brake right as it reaches an intersection for the comfort of the human. The human intervenes by braking when they expect the AV to not slow down, or slow down too late.

Robot Box-Pushing									
		$\mathbf{E_1}$	$\mathbf{E_2}$	$\mathbf{E_5}$	$\mathbf{E_8}$	$\mathbf{E_{10}}$	**Test 1**	**Test 2**	**Test 3**
$\mathbf{H = 1}$	**I Acc.**	55.3	75.2	77.4	78.1	78.99	—	—	—
	C Acc.	78.4	88.1	83.7	82.2	83.5	85.6	86.5	82.7
$\mathbf{H \sim U(1, 4)}$	**I Acc.**	63.0	79.8	87.8	88.7	90.06	—	—	—
	C Acc.	92.01	95.7	96.4	96.7	97.2	97.8	97.6	97.1
$\mathbf{H \sim U(1, 5)}$	**I Acc.**	55.1	73.2	82.8	86.1	86.2	—	—	—
	C Acc.	94.2	96.1	95.4	95.4	97.07	97.9	98.0	97.4
$\mathbf{H \sim U(1, 6)}$	**I Acc.**	55.4	70.5	81.7	83.8	84.83	—	—	—
	C Acc.	89.2	97.1	95.1	96.5	98.0	98.5	98.0	98.3
$\mathbf{H \sim tN(3, 1.0)}$	**I Acc.**	51.3	73.5	81.3	84.1	84.35	—	—	—
	C Acc.	99.6	98.6	98.7	98.6	99.7	99.8	99.7	99.7
$\mathbf{H \sim tN(3, 0.25)}$	**I Acc.**	53.4	72.6	82.4	84.7	85.3	—	—	—
	C Acc.	99.5	99.1	100.0	99.5	99.2	100.0	99.6	99.5
Autonomous Vehicle Navigation									
		$\mathbf{E_1}$	$\mathbf{E_5}$	$\mathbf{E_{10}}$	$\mathbf{E_{15}}$	$\mathbf{E_{20}}$	**Test 1**	**Test 2**	**Test 3**
$\mathbf{H = 1}$	**I Acc.**	95.0	93.4	94.2	95.0	95.5	—	—	—
	C Acc.	55.0	63.0	63.0	62.0	63.0	65.0	64.0	63.0
$\mathbf{H \sim U(1, 4)}$	**I Acc.**	52.5	80.9	80.8	84.0	80.6	—	—	—
	C Acc.	61.0	75.0	77.0	76.0	76.0	75.0	76.0	76.0
$\mathbf{H \sim U(1, 5)}$	**I Acc.**	50.5	83.8	85.3	90.7	83.4	—	—	—
	C Acc.	60.0	78.0	77.0	81.0	78.0	79.0	79.0	79.0
$\mathbf{H \sim U(1, 6)}$	**I Acc.**	63.0	83.5	83.3	83.9	85.9	—	—	—
	C Acc.	61.0	77.0	77.0	80.0	81.0	81.0	81.0	81.0
$\mathbf{H \sim tN(3.5, 2.5)}$	**I Acc.**	60.0	82.6	84.8	85.5	81.5	—	—	—
	C Acc.	65.0	80.0	80.0	83.0	80.0	80.0	80.0	80.0
$\mathbf{H \sim tN(4, 0.5)}$	**I Acc.**	52.0	87.6	90.2	90.0	91.2	—	—	—
	C Acc.	66.0	79.0	81.0	82.0	82.0	83.0	82.0	82.0

Table 6.1: I Acc. and C Acc. denote the accuracies of the intervention and constraint models respectively after each of 5 epochs of training, $\mathbf{E_i}$., and in each of 3 test environments.

Robot Box-Pushing				
	Train Eval	**Test Eval 1**	**Test Eval 2**	**Test Eval 3**
`Ignorant`	$[2.67, -\mathbf{2.09}]$	$[4.42, -\mathbf{2.07}]$	$[5.96, -\mathbf{2.05}]$	$[3.92, -\mathbf{2.05}]$
$\mathbf{H = 1}$	$[0.33, -2.90]$	$[0.62, -3.04]$	$[0.34, -3.15]$	$[0.17, -3.00]$
$\mathbf{H \sim U(1, 4)}$	$[0.13, -2.88]$	$[0.26, -2.92]$	$[0.05, -3.08]$	$[0.02, -2.93]$
$\mathbf{H \sim U(1, 5)}$	$[0.13, -2.84]$	$[0.26, -2.94]$	$[0.04, -3.08]$	$[0.02, -2.94]$
$\mathbf{H \sim U(1, 6)}$	$[0.12, -2.82]$	$[0.26, -2.93]$	$[0.03, -3.07]$	$[0.02, -2.93]$
$\mathbf{H \sim tN(3, 1.0)}$	$[0.10, -2.84]$	$[0.25, -2.92]$	$[0.02, -3.06]$	$[0.01, -2.91]$
$\mathbf{H \sim tN(3, 0.25)}$	$[\mathbf{0.09}, -2.84]$	$[\mathbf{0.24}, -2.92]$	$[\mathbf{0.01}, -3.05]$	$[\mathbf{0.01}, -2.91]$

Autonomous Vehicle Navigation				
	Train Eval	**Test Eval 1**	**Test Eval 2**	**Test Eval 3**
`Ignorant`	$[1.04, -\mathbf{14.07}]$	$[0.66, -\mathbf{9.43}]$	$[0.87, -\mathbf{10.37}]$	$[0.89, -\mathbf{17.13}]$
$\mathbf{H = 1}$	$[0.75, -15.19]$	$[0.68, -9.52]$	$[0.33, -12.83]$	$[0.76, -17.57]$
$\mathbf{H \sim U(1, 4)}$	$[0.39, -16.03]$	$[0.34, -10.31]$	$[0.17, -13.20]$	$[0.36, -18.71]$
$\mathbf{H \sim U(1, 5)}$	$[0.39, -15.77]$	$[0.34, -10.32]$	$[0.27, -12.56]$	$[0.38, -18.63]$
$\mathbf{H \sim U(1, 6)}$	$[0.07, -16.59]$	$[0.10, -10.83]$	$[0.07, -13.04]$	$[0.05, -19.21]$
$\mathbf{H \sim tN(3.5, 2.5)}$	$[0.35, -16.07]$	$[0.33, -10.29]$	$[0.17, -13.19]$	$[0.36, -18.58]$
$\mathbf{H \sim tN(4, 0.5)}$	$[\mathbf{0.02}, -16.62]$	$[\mathbf{0.00}, -10.96]$	$[\mathbf{0.00}, -12.92]$	$[\mathbf{.003}, -19.27]$

Table 6.2: Bracketed values $[x, y]$ denote the sample average constraint violations per randomized episode (x) and sample average nominal reward value per randomized episode (y), over $10,000$ simulations.

Results

Tables 6.1 and 6.2 show the results from our experiment. In particular, we present both the classification accuracy on interventions (i.e., predicting whether the human will intervene for a given state-action pair) and constraints (i.e., predicting whether a given state-action pair belong to the true constraint constraint set) after the 1st, 5th, 10th, 15th, and 20th epoch of training in table 6.1. In table 6.2 we present the average objective value for each of the two objectives – constraint minimization and domain reward maximization – over 10,000 trials for 4 domains: the domain where the agent was trained, and three different test domains. Here, we compare the performance against a reactive agent that assumes that the human always intervenes due to the current state-action pair, and five different proactive agents, to illustrate how the agent's prior knowledge of the human's temporal model affects its learning ability. We tested a proactive agent with three uniform distributions over the horizon H — $\mathbf{U(1,4)}$, $\mathbf{U(1,5)}$, and $\mathbf{U(1,6)}$ — and two truncated normal distributions over H — $\mathbf{tN(3.0,1.0)/tN(3.5,2.5)}$ and $\mathbf{tN(3,0.5)/tN(4,0.5)}$ (for the two domains respectively) truncated from 1-6 — where in all experiments the human had a *fixed* horizon of 3 for the box pushing domain and 4 for the autonomous vehicle domain.

In table 6.1, we see that despite the success of the reactive agent in the autonomous vehicle domain at achieving high intervention accuracy (although, notably not in the box pushing domain), it is never able to break 90% and 70% in the two domains respectively in terms of accuracy on the constraint set in either the train or test domains, significantly under-performing in terms of constraint accuracy when compared to the proactive agents. This illustrates that the agent is learning the wrong measure to apply to future environments where predicting an intervention does not directly predict a constraint. On the other hand, each proactive agent is able to achieve high ($\approx 85-90\%$) accuracy on the intervention set, and nearly 100% accuracy in the box pushing domain, and at least 80% accuracy in the autonomous vehicle domain, for the constraint set.

In table 6.2, we included the performance of an agent, referred to as `Ignorant`, which did not model the constraint set at all and only attempted to maximize the nominal objective, to provide a baseline for the performance on the nominal objective. Notably, the reactive agent did not perform the best in either objective across all environments tested. In the box pushing domain, there was not significant difference in performance between the five proactive agents, but all five significantly outperformed the reactive agent in the primary objective, and even outperformed the reactive agent in the nominal objective as well. These results indicate that in some domains, simply accounting for proactivity is enough to properly learn the constraint set.

In the autonomous vehicle domain, the proactive agent with the truncated normal distribution $\mathbf{tN(3.5, 2.5)}$ performed comparably to the best performing proactive agent with uniform distribution ($\mathbf{U(1, 6)}$), and generally outperformed the other two uniform-distribution agents across the four domains. This indicates that appropriate coverage over the horizon, so that sufficient *weight* is placed on or around the human's true horizon, can be more important than the distribution itself when the prior is fairly uncertain.

However, an agent with almost perfect knowledge of the human's horizon, using distributions $\mathbf{tN(3, 0.5)}$ and $\mathbf{tN(4, 0.5)}$ for the two domains led to the best results over all in both domains. This is particularly notable in the autonomous vehicle domain where the agent incurred a 0.0, or nearly 0.0, sample likelihood of a constraint violation in the three test environments, outperforming the other proactive agents by at least an order of magnitude. These results strongly indicate that a well-informed prior on the human's temporal model can significantly improve the quality of the learned constraint set by the agent in domains where the constraints are sparse, which is particularly important in safety-critical domains like autonomous vehicles.

6.4 Discussion

6.4.1 Learning Setting

In this chapter, we considered a train-then-deploy learning setting in which the agent has a fixed amount of time or resources to learn its objective (in this case, the constraint set) in the context of a human teacher who can provide a safety net for the agent before it is deployed in to its operational environment without such a safety net, but where the constraints will continue to hold. The natural extension is the online, or "learn-on-the-go", setting where the agent's training occurs throughout its operation. The challenge of such a setting is that, generally, human interventions are not free, as we have considered in this chapter; as a consequence, a non-fully cooperative game is induced as the agent's objective is now impacted by the likelihood of an intervention, impacting its own policy, which is known by the human and determines the human's likelihood of intervening.

6.4.2 Constraint Optimization

There are several ways of modeling constraint optimization problems; in this chapter, we considered one that is naturally applicable to the types of open-world, safety-critical domains that motivate this thesis, wherein we aim to minimize the expected number of constraint violations as the primary objective in a lexicographic optimization setting. We briefly discuss some other constraint settings one can easily extend our approach to. First, one may consider a *soft-constraint* setting, where constraints model non-critical elements like preferences or negative side effects [99]; this approach can be naturally formulated as *either* an LMDP, as done here, but where the constraint violations represent the *secondary* objective, rather than the primary, or it can be formulated as a MOMDP [92], as done here in the training phase but which can be extended to the deployment phase as well. Second, one may consider a *budgeted* violation setting in which there is an acceptable number of constraint violations that are allowable, but where the final policy must ensure that the budget itself is not violated. Such an approach can be naturally formulated as a linear program, which is a known approach to compute an optimal policy for a Markov decision process [71].

6.5 Conclusion

In this chapter, we consider the challenge of learning from proactive feedback, i.e., feedback generated by the human operator that is conditioned on their inferred near-term future behavior of the system under an ϵ-noisy model of the agent's policy. While the problem is presented in its most general form, and is not specific to the competence model, we treat the learning problem in a simple competence-aware setting where the system aims to learn a set of constraints on autonomy, which can be viewed as where it has the competence to operate autonomously or not. We specifically focus on learning from sparse interventions in a train-then-deploy setting, rather than other types of feedback such as full or partial demonstrations, agent-driven queries, or action guidance, although extending to such types of feedback, or even multiple of them, is a natural and interesting direction for future work.

We prove that if the learned constraint model is at least $(1-\alpha)$ accuracy, then the expected number of constraint violations under the optimal policy for the learned constraint model will be at most a factor of $\frac{1+\alpha T}{1-\alpha T}$ of the optimal expected number of constraint violations, when operating in a finite horizon setting with horizon $T \in \mathbb{N}$. Additionally we provide rigorous empirical evaluations that demonstrated that even though a reactive agent can learn with high accuracy where the human is going to intervene in the training environment, their learned constraint model has low accuracy and does not generalize well to new environments within the domain, leading to a much higher sample likelihood of a constraint violation during the deployment phase as compared to a proactive agent with even a uniform prior on the human's temporal horizon. As a result, we suggest that in real world settings where the human's feedback may be proactive, it is critical that models for learning from such feedback consider this proactivity to ensure that they learn the correct model. Furthermore, our results strongly indicate that a high quality prior over the human's temporal model can significantly improve results, leading to an almost zero sample likelihood of a constraint violation in almost all test environments considered. Future work will examine how calibration tasks, such as those performed by Schrum et al. [104], may be used to produce high quality priors on the human's temporal model.

CHAPTER 7
CONCLUSION AND FUTURE WORK

This thesis proposes a paradigm for sequential stochastic decision-making in open-world domains based on the notion of *competence modeling*, called *competence-aware autonomy*. Competence-aware autonomy is, intuitively, the ability of an agent to learn and reason about (1) their limitations in executing a task autonomously, (2) the environmental or situational factors that influence these limitations, and (3) the proper form and extent of human assistance to request to optimally compensate for their limitations. We suggest that such an ability is crucial for the long-term success of intelligent robotic systems deployed in the open world for long durations.

To this end, we proposed a formal planning framework called a *competence-aware system* that enables an autonomous system that can operate in multiple levels of autonomy and is supported by external human assistance to model and learn its competence over time through interactions with the human operator. We showeqd that such a system can learn its competence exactly, optimizing its autonomous operation and, by extension, its reliance on human assistance over time as well. We then showed how the CAS model can be extended to partially-observable domains in a well-defined manner that handles the nuances associated with competence when the state is not fully known. In each subsequent chapter of the thesis, we introduced novel models and methodological extensions that could handle challenges introduced by relaxing one or more of the baseline assumptions made in the formulation of the initial competence-aware system.

Notably, we first demonstrated that a CAS can improve its competence online by refining its state space over time by exploiting apparent inconsistencies in observed feedback generated by the human operator, without adding additional work to the human in doing so. We then proposed an extension to the CAS model that provides the representational power to model and handle multiple, heterogeneous human operators with stochastic, dynamic internal states, as well as global and local contextual information on which the system's competence depends. In particular, we showed that under the analogous assumptions (lifted to the extended setting), the contextual CAS model will still maintain the same convergence guarantees as the CAS model. Finally, we relaxed the standard assumption made in learning from human feedback that feedback (i.e., interventions or demonstrations) are causally conditioned on the concurrent behavior of the system, or behavior of the system that had just occurred. Instead, based on recent work in human cognitive control, we suggested that in many real world domains human feedback may instead be generated *proactively*, conditioned on the inferred future behavior of the system by the human under the human's model of the system's behavior. We provided a learning methodology based on proactive feedback in the context of constraint learning that relied on minimal assumptions about the human, but was still able to learn the true constraints in the majority of the constraint space, and significantly outperformed an agent which assumed purely reactive feedback in terms of the sample likelihood of a constraint violation.

7.1 Future Work

This thesis presents an initial effort towards the development of competence-aware autonomy for open-world intelligence. We suggest that the use of competence-modeling in sequential decision-making systems operating in the open world can enable them to maintain safer and more reliable behavior by optimizing their autonomy and reliance on external human assistance. Indeed, we have shown that the models and methodologies presented can be applied even in the context of challenging real-world conditions such as partial-observability, multiple dynamic heterogeneous humans, and proactive human feedback, even when the agent's initial model is incomplete in its feature representation.

However, there remains numerous directions for fruitful future work in the area of competence-aware autonomy for open-world intelligence. Perhaps the most obvious direction is in the application of competence-modeling to *fleets* of systems, which may each share some (or all) of the same technical specifications and domain model, but may interact with only some overlap in human operators, or may even each have a completely unique human operator. This introduces interesting challenges, such as how to best share information, particularly when the variations in underlying assumptions is not known precisely, in order to expedite global competence-learning, or how to bootstrap the competence model of a new system introduced into the fleet while still maintaining system guarantees such as level-safety.

Additionally, in this thesis we have assumed that either the human operator(s) has a good technical understanding of the agent to ensure that safe behavior is maintained, and high quality feedback is provided, or, when that is not the case, that only non-technical (e.g., preferential) feedback is provided by the human(s). A useful direction of future work would be to explore how the agent and human can start with little to no information about each other in a life-long, collaborative learning setting where both the human's model of the system and the domain itself may be non-stationary, and determine if competence can still be learned safely and efficiently.

Finally, we have only considered a limited set of discrete feedback signals that the agent can receive from the human due to their simple yet expressive nature. However, there is a significant amount of work that considers additional forms of feedback such as explicit action recommendations, action set restrictions or guidance, behavioral demonstrations, or even information-seeking actions. Although such types of feedback are not always available, as discussed previously in the thesis, in domains where they are available (and particularly if they are cheap to provide), learning competence from them may be an efficient and effective approach.

BIBLIOGRAPHY

[1] James E. Allen, Curry I. Guinn, and Eric Horvitz. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 1999. doi: 10.1109/5254.796083.

[2] Eitan Altman. *Constrained Markov decision processes: stochastic modeling.* Routledge, 1999.

[3] L. Gregory Appelbaum, C. Nicolas Boehler, Lauren A. Davis, Robert J. Won, and Marty G. Woldorff. The dynamics of proactive and reactive cognitive control processes in the human brain. *Journal of Cognitive Neuroscience*, 26(5): 1021–1038, 2014.

[4] Peter D. Ashworth and Judy Saxton. On 'competence'. *Journal of Further and Higher Education*, 14(2):3–25, 1990. doi: 10.1080/0309877900140201.

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

[6] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[7] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Learning to optimize autonomy in competence-aware systems. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 123–131, 2020.

[8] Connor Basich, Justin Svegliato, Allyson Beach, Kyle H. Wray, Stefan Witwicki, and Shlomo Zilberstein. Improving competence via iterative state space refinement. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1865–1871. IEEE, 2021.

[9] Connor Basich, John Peterson, and Shlomo Zilberstein. Planning with intermittent state observability: Knowing when to act blind. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11657–11664. IEEE, 2022.

[10] Connor Basich, Kyle Hollins Wray, Stefan Witwicki, and Shlomo Zilberstein. Introspective competence modeling for av decision making, April 19 2022. US Patent 11,307,585.

[11] Connor Basich, Justin Svegliato, Kyle H. Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Competence-aware systems. *Artificial Intelligence*, 316:103844, 2023.

[12] Jacob Beal and Miles Rogers. Levels of autonomy in synthetic biology engineering. *Molecular Systems Biology*, 16(12):e10019, 2020. doi: 10.15252/msb.202010019.

[13] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *Science Robotics*, 3 (21):eaat5954, 2018.

[14] Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

[15] David Blackwell. Discrete dynamic programming. *The Annals of Mathematical Statistics*, pages 719–726, 1962.

[16] Blai Bonet and Hector Geffner. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1641–1646. Citeseer, 2009.

[17] Jeffrey M. Bradshaw, Hyuckchul Jung, Shri Kulkarni, Matthew Johnson, Paul Feltovich, James Allen, Larry Bunch, Nathanael Chambers, Lucian Galescu, Renia Jeffers, et al. Kaa: Policy-based explorations of a richer model for adjustable autonomy. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 214–221, 2005.

[18] Todd S. Braver. The variable nature of cognitive control: a dual mechanisms framework. *Trends in cognitive sciences*, 16(2):106–113, 2012.

[19] John Bresina, Ari Jónsson, Paul Morris, and Kanna Rajan. Mixed-initiative activity planning for Mars rovers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1709–1710, 2005.

[20] Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, C. Guarino Lo Bianco, and Aurelio Piazzi. The ARGO autonomous vehicle's vision and control systems. *International Journal of Intelligent Control and Systems*, 3(4):409–441, 1999.

[21] Alberto Broggi, Pietro Cerri, Mirko Felisa, Maria Chiara Laghi, Luca Mazzei, and Pier Paolo Porta. The VisLab intercontinental autonomous challenge: An extensive test for a platoon of intelligent vehicles. *International Journal of Vehicle Autonomous Systems*, 10(3):147–164, 2012. doi: 10.1504/IJVAS.2012.051250.

[22] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning research*, 13:27–66, 2012.

[23] David J. Bruemmer, Douglas A. Few, Ronald L. Boring, Julie L. Marble, Miles C. Walton, and Curtis W. Nielsen. Shared understanding for collaborative control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(4):494–504, 2005.

[24] Roberto Capobianco, Guglielmo Gemignani, Luca Iocchi, Daniele Nardi, Francesco Riccio, and Andrea Vanzo. Contexts for symbiotic autonomy: Semantic mapping, task teaching and social robotics. In *AAAI Workshop on Symbiotic Cognitive Systems*, 2016.

[25] Michael Cashmore, Maria Fox, Tom Larkworthy, Derek Long, and Daniele Magazzeni. AUV mission control via temporal planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6535–6541, 2014.

[26] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009. doi: 10.1613/jair.2584.

[27] Manolis Chiou, Nick Hawes, Rustam Stolkin, Kimron L Shapiro, Jess R Kerlin, and Andrew Clouter. Towards the principled study of variable autonomy in mobile robots. In *IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*, pages 1053–1059. IEEE, 2015.

[28] Manolis Chiou, Nick Hawes, and Rustam Stolkin. Mixed-initiative variable autonomy for remotely operated mobile robots. *ACM Transactions on Human-Robot Interaction (THRI)*, 10(4):1–34, 2021.

[29] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations with grid and parametric representations. *The International Journal of Robotics Research*, 40(10-11):1255–1283, 2021.

[30] Jeffery A. Clouse. *On integrating apprentice learning and reinforcement learning.* PhD thesis, University of Massachusetts, 1996.

[31] Silvia Coradeschi and Alessandro Saffiotti. Symbiotic robotic systems: Humans, robots, and smart environments. *IEEE Intelligent Systems*, 21(3):82–84, 2006. doi: 10.1109/MIS.2006.59.

[32] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Shared autonomy systems with stochastic operator models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4614–4620, 2022.

[33] Ernst Dieter Dickmanns. *Dynamic vision for perception and control of motion.* Springer Science & Business Media, 2007.

[34] Gregory Dorais, R. Peter Bonasso, David Kortenkamp, Barney Pell, and Debra Schreckenghost. Adjustable autonomy for human-centered autonomous systems. In *IJCAI Workshop on Adjustable Autonomy Systems*, pages 16–35, 1999.

[35] David D. Dubois. *The competency casebook: Twelve studies in competency-based performance improvement.* Human Resource Development, 1998.

[36] Lance Eliot. Legal judgment prediction (ljp) amid the advent of autonomous ai legal reasoning. *arXiv preprint arXiv:2009.14620*, 2020.

[37] Lance Eliot. An ontological AI-and-law framework for the autonomous levels of AI legal reasoning. *arXiv preprint arXiv:2008.07328*, 2020.

[38] Eugene A. Feinberg and Adam Shwartz. *Handbook of Markov decision processes: Methods and applications*, volume 40. Springer, 2012.

[39] George Ferguson, James F. Allen, Bradford W. Miller, et al. TRAINS-95: Towards a mixed-initiative planning assistant. In *AAAI International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 70–77, 1996.

[40] Fanny Ficuciello, Guglielmo Tamburrini, Alberto Arezzo, Luigi Villani, and Bruno Siciliano. Autonomy in surgical robots and its meaningful human control. *Journal of Behavioral Robotics*, 10(1):30–43, 2019. doi: 10.1515/pjbr-2019-0002.

[41] Terrence Fong, Charles Thorpe, and Charles Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 50(4): 699–704, 2003.

[42] Yang Gao and Steve Chien. Review on space robotics: Toward top-level science through space exploration. *Science Robotics*, 2(7), 2017. doi: 10.1126/scirobotics.aan5074.

[43] E. Amir M. Ghalamzan, Firas Abi-Farraj, Paolo Robuffo Giordano, and Rustam Stolkin. Human-in-the-loop optimisation: mixed initiative grasping for optimally facilitating post-grasp manipulative actions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3386–3393. IEEE, 2017.

[44] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting.* Cambridge University Press, 2016.

[45] Thomas F. Gilbert. *Human Competence: Engineering Worthy Performance.* 1996.

[46] Ella Glikson and Anita Williams Woolley. Human trust in artificial intelligence: Review of empirical research. *Academy of Management Annals*, 14(2):627–660, 2020.

[47] Nathan A. Greenblatt. Self-driving cars and the law. *IEEE Spectrum*, 53(2): 46–51, 2016.

[48] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*, 26, 2013.

[49] Paul Hager and Andrew Gonczi. What is competence? *Medical Teacher*, 18(1): 15–18, 1996. doi: 10.3109/01421599609040255.

[50] Nick Hawes, Christopher Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrova, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Kortner, et al. The STRANDS project: Long-term autonomy in everyday environments. *IEEE Robotics & Automation Magazine*, 24(3):146–156, 2017. doi: 10.1109/MRA.2016.2636359.

[51] G.D. Hermann and R.J. Kenyon. Competence-based vocational education, project report, 1987.

[52] Charlie Hewitt, Ioannis Politis, Theocharis Amanatidis, and Advait Sarkar. Assessing public perception of self-driving cars: The autonomous vehicle acceptance model. In *International Conference on Intelligent User Interfaces*, pages 518–527, 2019.

[53] Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors*, 57(3):407–434, 2015. doi: 10.1177/0018720814547570.

[54] Jarrett Holtz, Arjun Guha, and Joydeep Biswas. Interactive robot transition repair with smt. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4905–4911, 2018.

[55] Fernando Huenupán, Nestor Becerra Yoma, Carlos Molina, and Claudio Garretón. Confidence based multiple classifier fusion in speaker verification. *Pattern Recognition Letters*, 29(7):957–966, 2008. doi: 10.1016/j.patrec.2008.01.015.

[56] Shu Jiang and Ronald C. Arkin. Mixed-initiative human-robot interaction: definition, taxonomy, and survey. In *IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*, pages 954–961. IEEE, 2015.

[57] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research (JAIR)*, 4:237–285, 1996.

[58] W. Bradley Knox, Cynthia Breazeal, and Peter Stone. Learning from feedback on actions past and intended. In *ACM/IEEE International Conference on Human-Robot Interaction, Late-Breaking Reports Session (HRI)*. Citeseer, 2012.

[59] W. Bradley Knox, Peter Stone, and Cynthia Breazeal. Training a robot via human feedback: A case study. In *International Conference on Social Robotics (ICSR)*, pages 460–470. Springer, 2013.

[60] Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented MDPs with dead ends. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 438–447, 2012.

[61] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley & Sons, 2004.

[62] Clayton Kunz, Chris Murphy, Hanumant Singh, Claire Pontbriand, Robert A. Sohn, Sandipa Singh, Taichi Sato, Chris Roman, Koichi Nakamura, Michael Jakuba, et al. Toward extraplanetary under-ice exploration: Robotic steps in the Arctic. *Journal of Field Robotics*, 26(4):411–429, 2009. doi: 10.1002/rob.20288.

[63] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomáš Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4023–4030, 2018.

[64] John D. Lee and Katrina A. See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.

[65] Quanyi Li, Zhenghao Peng, and Bolei Zhou. Efficient learning of safe driving policy via human-ai copilot optimization. In *International Conference on Learning Representations, (ICLR)*, 2022.

[66] Patrick Lin. Why ethics matters for autonomous cars. In *Autonomous driving*, pages 69–85. Springer, 2016.

[67] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *Computing Research Repository*, abs/2211.08416, 2022.

[68] Rafal Lysiak, Marek Kurzynski, and Tomasz Woloszynski. Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers. *Neurocomputing*, 126:29–35, 2014. doi: 10.1016/j.neucom.2013.01.052.

[69] Saaduddin Mahmud, Connor Basich, and Shlomo Zilberstein. Semi-autonomous systems with contextual competence awareness. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2023.

[70] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 7390–7399, 2021.

[71] Alan S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[72] Markus Maurer, J. Christian Gerdes, Barbara Lenz, and Hermann Winner. *Autonomous driving: technical, legal and social aspects*. Springer, 2016.

[73] David L. McPherson, Kaylene C. Stocking, and S. Shankar Sastry. Maximum likelihood constraint inference from stochastic demonstrations. In *Conference on Control Technology and Applications (CCTA)*, pages 1208–1213. IEEE, 2021.

[74] Emily McQuillin, Nikhil Churamani, and Hatice Gunes. Learning socially appropriate robo-waiter behaviours through real-time user feedback. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 541–550, 2022.

[75] Wim Meeussen, Eitan Marder-Eppstein, Kevin Watts, and Brian P. Gerkey. Long term autonomy in office environments. In *Robotics: Science and Systems (RSS) Alone Workshop*, 2011.

[76] Vera Zaychik Moffitt, Jerry L. Franke, and Meghann Lomas. Mixed-initiative adjustable autonomy in multi-vehicle operations. *Association for Unmanned Vehicle Systems International*, 2006.

[77] Ithan Moreira, Javier Rivas, Francisco Cruz, Richard Dazeley, Angel Ayala, and Bruno Fernandes. Deep reinforcement learning with interactive feedback in a human–robot environment. *Applied Sciences*, 10(16):5574, 2020.

[78] Salama A. Mostafa, Mohd Sharifuddin Ahmad, and Aida Mustapha. Adjustable autonomy: A systematic literature review. *Artificial Intelligence Review*, 51(2): 149–186, 2019. doi: 10.1007/s10462-017-9560-8.

[79] John F. Mustard, David W. Beaty, and Deborah S. Bass. Mars 2020 science rover: Science goals and mission concept. In *AAS Division for Planetary Sciences Annual Meeting*, volume 45, pages 211–217, 2013.

[80] Dimitris Papadimitriou, Usman Anwar, and Daniel S. Brown. Bayesian methods for constraint inference in reinforcement learning. *Transactions on Machine Learning Research*, 2022.

[81] Raja Parasuraman, Thomas B. Sheridan, and Christopher D. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[82] Stephen D. Patek. On partially observed stochastic shortest path problems. In *IEEE Conference on Decision and Control (CDC)*, volume 5, pages 5050–5055. IEEE, 2001.

[83] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005. doi: 10.1109/TPAMI.2005.159.

[84] Giannis Petousakis, Manolis Chiou, Grigoris Nikolaou, and Rustam Stolkin. Human operator cognitive availability aware mixed-initiative control. In *IEEE International Conference on Human-Machine Systems (ICHMS)*, pages 1–4. IEEE, 2020.

[85] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M. Mitchell. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*, 2019.

[86] Sadegh Rabiee and Joydeep Biswas. IVOA: Introspective vision for obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1230–1235, 2019.

[87] Sadegh Rabiee, Connor Basich, Kyle Hollins Wray, Shlomo Zilberstein, and Joydeep Biswas. Competence-aware path planning via introspective perception. *IEEE Robotics and Automation Letters*, 2022. doi: 10.1109/LRA.2022.3145517.

[88] Ramya Ramakrishnan, Ece Kamar, Besmira Nushi, Debadeepta Dey, Julie Shah, and Eric Horvitz. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. In *AAAI conference on Artificial Intelligence (AAAI)*, pages 6137–6145. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33016137.

[89] L. A. Rastrigin and R. H. Erenstein. Method of collective recognition. *Energoizdat*, 595:37, 1981.

[90] Marc Rigter, Bruno Lacerda, and Nick Hawes. A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, 2020. doi: 10.1109/LRA.2020.2970619.

[91] Robotis. TurtleBot3 e-Manual, 2020. URL `https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`.

[92] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[93] Michael T. Rosenstein and Andrew G. Barto. Supervised actor-critic reinforcement learning. In J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*, pages 359–380. IEEE Press, 2004.

[94] Stephanie Rosenthal, Joydeep Biswas, and Manuela M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, volume 10, pages 915–922, 2010.

[95] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics (ICAIS)*, pages 661–668. JMLR, 2010.

[96] SAE On-Road Automated Vehicle Standards Committee. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standards Journal*, 3016:1–16, 2014.

[97] Alessandro Saffiotti, Mathias Broxvall, Marco Gritti, Kevin LeBlanc, Robert Lundh, Jayedur Rashid, BeomSu Seo, and Young-Jo Cho. The PEIS-ecology project: Vision and results. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2329–2335, 2008.

[98] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[99] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. A multi-objective approach to mitigate negative side effects. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[100] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. Avoiding negative side effects of autonomous systems in the open world. *Journal of Artificial Intelligence Research*, 74:143–177, 2022.

[101] Demetrios Sampson and Demetrios Fytros. Competence models in technology-enhanced competence-based learning. In *Handbook on Information Technologies for Education and Training*, pages 155–177. Springer, 2008.

[102] Paul Scerri and Nancy Reed. Designing agents for systems with adjustable autonomy. In *IJCAI Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.

[103] Paul Scerri, David V. Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002. doi: 10.1613/jair.1037.

[104] Mariah L. Schrum, Erin Hedlund-Botti, Nina Moorman, and Matthew C. Gombolay. Mind meld: Personalized meta-learning for robot-centric imitation learning. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 157–165. IEEE, 2022.

[105] Dexter R. R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[106] Baris Senliol, Gokhan Gulgezen, Lei Yu, and Zehra Cataltepe. Fast correlation based filter with a different search strategy. In *IEEE International Symposium on Circuits and Systems (ISCIS)*, pages 1–4, 2008.

[107] Thomas B. Sheridan. *Telerobotics, Automation, and Human supervisory control.* Cambridge, MA: MIT Press, 1992.

[108] Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001. doi: 10.1111/0824-7935.00142.

[109] Alexandre Sousa, Luis Madureira, Jorge Coelho, José Pinto, João Pereira, João Borges Sousa, and Paulo Dias. LAUV: The man-portable autonomous underwater vehicle. *International Federation of Automatic Control*, 45(5):268–274, 2012. doi: 10.3182/20120410-3-PT-4028.00045.

[110] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. Expert intervention learning. *Autonomous Robots*, 46(1):99–113, 2022.

[111] Robert J. Sternberg and J.E. Kolligian Jr. *Competence considered.* Yale University Press, 1990.

[112] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[113] Justin Svegliato, Kyle Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Belief space metareasoning for exception recovery. In *IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, pages 1224–1229, 2019.

[114] Justin Svegliato, Samer B. Nashed, and Shlomo Zilberstein. Ethically compliant sequential decision making. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11657–11665, 2021.

[115] Further Education Unit. Towards a competence-based system: An FEU view, 1984.

[116] B. Vecht. *Adjustable autonomy: Controlling influences on decision making.* Utrecht University, 2009.

[117] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Tom Kollar, Cetin Mericli, Mehdi Samadi, Susana Brandao, and Rodrigo Ventura. Cobots: Collaborative robots servicing multi-floor buildings. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5446–5447, 2012.

[118] Manuela M. Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. CoBots: Robust symbiotic autonomous mobile service robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4423–4429, 2015.

[119] Tomasz Woloszynski and Marek Kurzynski. On a new measure of classifier competence applied to the design of multiclassifier systems. In *International Conference on Image Analysis and Processing (ICIAP)*, pages 995–1004, 2009.

[120] Tomasz Woloszynski and Marek Kurzynski. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, 44(10-11): 2656–2668, 2011. doi: 10.1016/j.patcog.2011.03.020.

[121] Tomasz Woloszynski, Marek Kurzynski, Pawel Podsiadlo, and Gwidon W Stachowiak. A measure of competence based on random classification for dynamic ensemble selection. *Information Fusion*, 13(3):207–213, 2012. doi: 10.1016/j.inffus.2011.03.007.

[122] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997. doi: 10.1109/34.588027.

[123] Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. Multi-objective MDPs with conditional lexicographic reward preferences. In *AAAI conference on Artificial Intelligence (AAAI)*, pages 3418–3424, 2015.

[124] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. Hierarchical approach to transfer of control in semi-autonomous systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 517–523, 2016.

[125] Guang-Zhong Yang, James Cambias, Kevin Cleary, Eric Daimler, James Drake, Pierre E. Dupont, Nobuhiko Hata, Peter Kazanzides, Sylvain Martel, Rajni V. Patel, et al. Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy. *Science Robotics*, 2(4):8638, 2017. doi: 10.1126/scirobotics.aam8638.

[126] Zhenyu Zhao, Radhika Anand, and Mallory Wang. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 442–452. IEEE, 2019.

[127] Stéphane Zieba, Philippe Polet, Frédéric Vanderhaegen, and Serge Debernard. Principles of adjustable autonomy: A framework for resilient human-machine cooperation. *Cognition, Technology & Work*, 12(3):193–203, 2010. doi: 10.1007/s10111-009-0134-7.

[128] Shlomo Zilberstein. Building strong semi-autonomous systems. In *AAAI Conference on Artificial Intelligence*, pages 4088–4092, 2015.