Human-Aware AI Methods for Active Teaming

by

Sachin Grover

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2022 by the
Graduate Supervisory Committee:

Subbarao Kambhampati, Chair
David Smith
Sidhharth Srivastava
Kurt VanLehn

ARIZONA STATE UNIVERSITY

November 2022

ABSTRACT

The future will be replete with Artificial Intelligence (AI) based agents closely collaborating with humans. Although it is challenging to construct such systems for real-world conditions, the Intelligent Tutoring System (ITS) community has proposed several techniques to work closely with students. However, there is a need to extend these systems outside the controlled environment of the classroom. More recently, Human-Aware Planning (HAP) community has developed generalized AI techniques for collaborating with humans and providing personalized support or guidance to the collaborators. In this thesis, the take learning from the ITS community is extend to construct such human-aware systems for real-world domains and evaluate them with real stakeholders. First, the applicability of HAP to ITS is demonstrated, by modeling the behavior in a classroom and a state-of-the-art tutoring system called Dragoon. Then these techniques are extended to provide decision support to a human teammate and evaluate the effectiveness of the framework through ablation studies to support students in constructing their plan of study (iPOS). The results show that these techniques are helpful and can support users in their tasks. In the third section of the thesis, an ITS scenario of asking questions (or problems) in active environments is modeled by constructing questions to elicit a human teammate's model of understanding. The framework is evaluated through a user study, where the results show that the queries can be used for eliciting the human teammate's mental model.

In memory of the strongest person I knew my Grand-mother

ACKNOWLEDGEMENTS

A friend once told me, that no human is on an island. One's achievement is an outcome of one's hardwork and unconditional love and support from many people. In that regard I feel lucky to be raised by a loving family, and have had many friends who have supported me directly or indirectly during my degree. During my graduate studies I believe I have learned the most important lessons of managing work life balance, handle tough situations when results are not positive, stay motivated and navigate mental fitness for working daily and not just in small bursts. Most importantly I have learnt about things that make me happy. Before I thank people I would like to give special thanks to my place in the corner at Yochan lab. It was my hideout from the world and problems, where I felt safe and stayed motivated throughout my graduate years.

Professor (only because he doesn't like us to call that) Subbarao Kambhampati, my advisor has had the biggest impact on my degree. His honest and direct feedback has helped me improve my understanding of the subject and life in general. Over the years I have closely observed him to understand how one critically analyzes a problem from different perspectives, and how the idea of Science is to argue why something will not work, instead of why something will work. As pessimistic it sounds it's important to understand this as I believe one grows only when they can analyze a solution to understand it's limitations.

I came from India, to learn, without a clear idea of what I wanted to learn. Over the years, I worked closely with all my committee members. Dr. Kurt VanLehn supported me in my early years and was my advisor during my Masters degree.The only thing I knew was writing e-commerce websites wasn't what I wanted to keep doing. He had an impact on my understanding of research, what it can lead to and what does it mean to pursue research for a long time. Dr. Siddharth Srivastava

and Dr. David Smith supported me in broadening my perspective in the area of task planning. Dr. Srivastava helped me developed an understanding of various techniques through multi-agent planning problems (which I am still supposed to publish), and Dr. Smith guided me in modeling complex problems and I continue to use these learnings in my recent projects.

Discussions are central part of any Ph.D. degree, and one's labmates play an important role to create an understanding of research problems. When I joined I was able to work closely with Tathagata, Sailik and Sarath. The projects done with them formed some of the initial work that I presented in my thesis. Working with them and several discussions with them and other senior student Anagha shaped my understanding of research in general as well human-aware AI specific problems. They helped me understand the literature better and that helped me manage different research problems. Zahra, Alberto and Sriram joined the lab an year or two after me, and with them I learned to grow as a researcher. They have been my lunch partner for last four years, where we would wait for each other to join us so that we can eat together and catch up on the day's events. Our discussions would usually start with the "how is it going?" and from there it could take any turn under the sun. Post covid times there have been numerous parties which would be deeply missed. From Yantian, I learned the art of being disciplined and stay motivated for your work. I wish Utkarsh, Lin, Karthik and Sidhhant luck for their research, and want to thank them as our lab lunches became all the more interesting because of the happy environment that has persisted in the lab.

Throughout my Ph.D. I have had immense support of my friends and family during those times. Their presence (and in some cases followed by their absence) has played an important role in my life and shape the person I am today. Arun has been a friend who has been around from Day 1 of my Masters degree and his support

has been a constant. Pulkit, Arun, Vignesh and Bahar were not directly involved in any of the projects for my thesis, but I was able to collaborate with them and do some interesting projects. Most of them didn't convert to a publication, however, some of these projects helped broaden my understanding and I learnt to be a better teammate. I went through a tough time during my Ph.D. due to some personal life events, but I was always able to depend on Sailik, Anurag, Rohan, Anagha, Mansooreh, Jayanth, Harshini, Edi, Pulkit, Zahra, Sriram, Anuj, Shrinvas, Kushal, Khimya, Shruti, Prateek, to lend an ear and support me to cope with whatever I was going through. I have been lucky to have them around, as they usually would come by and check up regularly whether I was alright and on many days there support kept me trying harder to be a better person. I want to thank my roommates Abinash, Prashant, Shatadal, Anurag, Aniket, Tushar, Harshini, Jayanth, Aman and Hitesh over the years for putting up with my mood swings and hot-headedness.

Some of my friends are from my childhood, school or even undergraduate days, and they have also played an important role as they motivate me daily to be the best version of myself and strive to do more every moment of my life. I have known Anuj and Kushal from my high-school when we prepared for competitive exams in India. They are still around and I can always dependent on them to hear honest feedback and even a scolding if I need/deserve one. Balli, Vinay, Anupam, Ganesh, Shrinivas, Karan, Vaibhav, Jagdeep, Smriti, Abinash, Shatadal, Ravi, Prashant are some of my friends from my undergraduate college, with whom I have lived in a hostel, studied and had innumerable fun nights. Prashant, Ravi, Abinash, Shatadal and Shrinivas were also around during my ASU days and with them I could continue to closely mingle and having them around gave me a very strong support system for every situation.

I would also like to thank my mother, and my late father, who taught me to be empathetic person and have a never say die attitude. My elder brothers Naveen and Kapil from whom I learned the art of critical analysis and having arguments. My late grand-mother was the beacon of strength, whose love and care nurtured me to be a good person. I consider myself lucky to have Sahiba as my wife, who constantly argues that she doesn't understand my work but yet hasn't made an effort to understand it. As a last effort I have written my ideas in a thesis, let's see if she reads it this time around. I would also like to thank my father and mother in law, who in so little time have shown me so much love and care, that they put me at ease about facing the future as one family. I also would like to thank my cousins Renu, Sushma, Poonam, Sonia, Monika, Nisha, Sunny, Kaittkee for their impact on my life. I want to wish luck to the next generation of the family Pavit, Manas, Advika, Jay, Krisha, Tushar, Kanav, Saksham, Shagun, Anurag.

Finally I would like to thank Kobe Bryant. I grew up watching him play Basketball, and all the stories about his work ethic. I have tried to emulate that single minded work-ethic and pushing the boundaries of what I know and understand. This thesis is only the first step to the future where I plan to grow and keep working hard everyday and dwell into the unknown world of Science and Research.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Artificial Intelligence (AI) is being applied to many of our day-to-day activities such as driving to various places (through automated cars like Waymo), providing personalized assistance (through interactive voice assistants like Alexa or Google Now), and much more Many of these systems have been integrated in industries like manufacturing (automated production lines), finance (checking anomalies in transactions), healthcare (machine learning based diagnosis), education (helping students learn through feedback), and so on. AI techniques have made this possible, as they can learn complex patterns from data for intelligent short-term decision making. For example, a smart car learning to react after detecting other cars based on it's sensory inputs. However, these systems are neither *human-aware* as they do not account for human biases or preferences in their decision making process, nor, they are part of *active teams* as they are not working with active humans or supporting or guiding them in their complex activties. These properties can be particularly helpful in complex working environments such as hospitals, or a classroom, where a robot can be an active teammate with the nurse or a teacher and support them in patient care or guiding students to solve their problems (like a personalized teacher). Through this thesis, we understand the current state of such systems that were designed to work in the active environments especially with students and use them as motivation for designing generalized human-aware AI based systems to work with several different experts in a domain-independent fashion.

Intelligent Tutoring System (Anderson *et al.*, 1985) were the original automated/intelligent systems, earlier called *Computer Assisted Instruction* (Steinberg, 1991).

They have over fifty-year long history of research and development to design some of the first intelligent systems to closely work with humans. The research community has explored to provide support to students and bring in expert tutors wherever necessary, thus reducing the burden on the instructor as well as improving student's learning experience. There are many existing tutoring systems to teach various subjects, such as, physics (Schulze *et al.*, 2000), dynamic systems based on first-order differential equations (VanLehn *et al.*, 2017), SQL (Mitrovic, 2003), etc. There have been some subject independent tutoring systems that are modeled towards a specific method of teaching, such as Q&A interaction-based system (Graesser *et al.*, 2005). These systems have been successful at tracking a students learning (Corbett and Anderson, 1993; Piech *et al.*, 2015), ask them questions (Barla *et al.*, 2010; Zhang and VanLehn, 2016) to improve interaction with humans. These methods have continuously evolved over the years to domain specific tutoring cases. ITS framework has also been able to support more recent development in web-based teaching through MOOCs, where students learn at their own pace. Although these systems have been at the forefront of technology and have been thoroughly tested through human-studies, there is a need for a domain independent framework that can drive different kinds of student oriented learning experience. Moreover, there is also a need to extend the learning from ITS to other domains, while constructing human-aware intelligent systems for other domains.

More recently, *Human-Aware Planning* (HAP) or *Human-in-the-loop Planning* (HILP) or *Human-Aware AI* (HAAI) techniques (Kambhampati and Talamadupula, 2015) have shown promise, as they reason about the model of the human using automated planning techniques. Automated Planning techniques are domain independent techniques that has developed generalized methods to inference using a structured model (Ghallab *et al.*, 2004). Human-aware AI techniques have extended these tech-

niques to incoroporate human's mental model for reasoning about automated system's decisions (Kambhampati and Talamadupula, 2015), synthesize behavior explicable to the human-in-the-loop (Kulkarni *et al.*, 2016), and explain their decision using the human's understanding of the task (Chakraborti *et al.*, 2019b), etc. These techniques have also been applied to many domains, such as urban search and rescue (Talamadupula *et al.*, 2010a), decision support settings (Vadlamudi *et al.*, 2016), or even in commercial settings of warehouses (Liang *et al.*, 2015), restaurants (Berezina *et al.*, 2019). They can also assume many roles, such as the role of a personalized assistant to a human supervisor (Grochow, 2020) or as a teacher managing the dialogue between student and the tutoring system (Rahati and Kabanza, 2010). Although these systems can be applied to diverse settings, evaluating them with human stakeholders has become a central challenge.

To summarize, where HAP techniques can be generalized they have been tested on very few real-world scenarios. Moreover, the learning from tutoring systems also needs to be generalized to other complex domains for working with active human-teammates. Thus, to motivate our work, we look at ways to bring learning from ITS to HAAI and use it for active teaming scenarios where robot can provide support or actively learn the teammate's model for seamless collaboration. In this work, first we present a survey of how tutoring systems have been developed to work with students. Then we discuss how the generalized human-aware planning framework can be used to develop the theoretical background to support students and teachers working with an ITS in a domain-independent fashion by modeling state-of-the-art tutoring system, such as Dragoon (VanLehn *et al.*, 2016). We also show how personalized behavior can be developed to support more recent technologies. For the second part of our work, we model an important scenario of asking questions for tutoring system to an active teaming scenarios, i.e. to ask questions for human-robot teaming. Before we

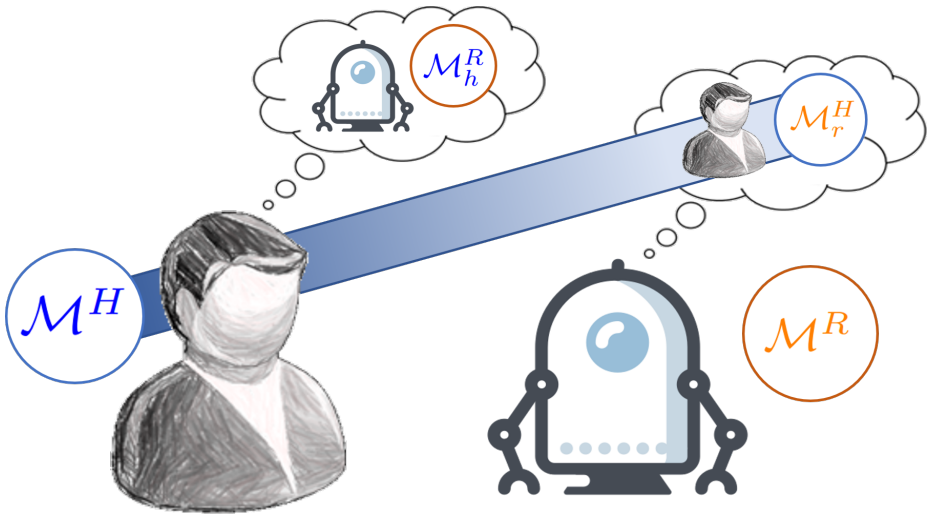


Figure 1.1: Depicts the human-aware AI setting. $\mathcal{M}^H$ is the human's model, $\mathcal{M}^R$ represents the robot's (or planner, software agent's) model and $\mathcal{M}_h^R$ represent human's understanding of the robot's model $\mathcal{M}^R$. Human-Aware AI proposes that automated agent should use $\mathcal{M}_h^R$ while collaborating with the human. This framework can be extended to include robot's understanding of the human's model $\mathcal{M}_h^R$ for collaborating with an active human-teammate.

get into the details of our work, we will describe the specific aspects of Human-Aware Planning techniques and research in Intelligent Tutoring System community that will be useful to understand our work.

1.1 Human-Aware Planning

Human-Aware Planning (HAP) framework proposes improved collaboration between human and robot by using human's mental model in robotic agent's decision making process (Kambhampati and Talamadupula, 2015). If we assume $\mathcal{M}^R$ and $\mathcal{M}^H$ are the AI agent's and human's model respectively; then agent's (henceforth *agent* used for an AI agent only) understanding of the human's model $\mathcal{M}^H$ can be represented as $\mathcal{M}_r^H$. It is assumed to be a close approximation of the teammate's (henceforth *teammate* is always the human member of the team) model $\mathcal{M}^H$. Human-aware planning proposes that robot should use $\mathcal{M}_h^R$ instead of it's own model to act in the environment with humans (Kambhampati and Talamadupula, 2015). For example,

if we apply the framework to a robot providing explanations, then the agent should use $\mathcal{M}_h^R$ instead of $\mathcal{M}^R$ to provide explanations (Chakraborti *et al.*, 2017b). This framework has been applied to showcase different kind of behaviors with humans, such as being explicable (Kulkarni *et al.*, 2016), predictable (Dragan *et al.*, 2013), and also provide explanations for it's decisions (Chakraborti *et al.*, 2019b). Explicability means the robot performs actions that match human's understanding of the robot, i.e. ideally it uses $\mathcal{M}_h^R$ instead of $\mathcal{M}^R$. Obviously, $\mathcal{M}_h^R$ is the human's model, hence robot can only approximate the model, however, ideally we assume the robot's approximation is equivalent to $\mathcal{M}_h^R$. A robot's plan is predictable when the robot is explicable and executes easiest to predict sub-plan (Chakraborti *et al.*, 2019a).

With active teammates, a new model $\mathcal{M}_r^H$ gets added to this tale of three models (Chakraborti *et al.*, 2018b). The agent needs to consider $\mathcal{M}_r^H$ with it's own model $\mathcal{M}^R$ in it's decision making process. Extending the original framework to incorporate $\mathcal{M}_r^H$ can be useful to support, teach and collaborate with human-teammates. For example, if we apply this extended framework to ITS, then $\mathcal{M}^R$ represents the model of the tutoring system, i.e. all the actions a tutoring system can perform, such as presenting a problem to the student, guiding them through the problem solving process, provide feedback, etc. $\mathcal{M}^H$ is the model of the student, i.e. all the actions it can perform on the interface of the ITS, and the concepts they have learned. $\mathcal{M}_r^H$ represents the system's understanding of the human, such as the concepts they have learned or practiced and the problems they can solve and their model for interacting with the interface. Generally, $\mathcal{M}_r^H$ is an approximation and is learned from the behavior of the human and we assume the approximation is close to the original mental model. We also assume that the approximated model $\mathcal{M}_r^H$ is granular enough to be useful for planning in the environment. In this work, we are looking at collaboration between the human and automated agents. Thus, the planning task for the agent is based on

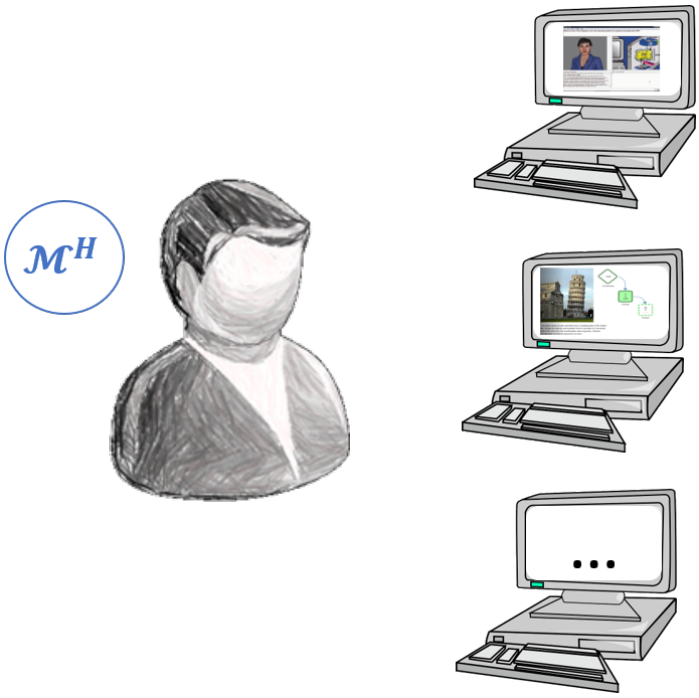


Figure 1.2: In ITS setting, same human works with different tutoring systems, where none of them tend to incorporate the student's understanding of the system. Instead, they ensure the learning and understanding of the specific topic through interaction.

the teammate's current requirements. For example, for a tutoring system, goal is to support the student solve the problem presented on the interface and to help them complete their practice material.

The updated generalized framework for human-aware AI with active human-teammates, can be applied to many different human-robot settings. However, due to lack of stakeholders and also due to domain specific challenges, evaluating them through user studies can be challenging. We believe Intelligent Tutoring System can be used to evaluate these techniques and these techniques can provide personalization techniques for the supporting students in classroom scenario as well as the more recent systems for independent learning through MOOCs. Before we present the details for how these techniques can be used for ITS in future sections, we will discuss a review of the theory from the tutoring system community.

1.2 Intelligent Tutoring System

A system that presents a problem to a student, provides feedback to them and ask the instructor for intervention at the relevant moment is called an Intelligent Tutoring System (Anderson *et al.*, 1985). The feedback provided while solving a problem, is an essential part of a tutoring system and it separates ITS from computer-aided instruction. Infact, it has been shown that if the system can provide the right feedback at the right moment, then it can be as effective (if not more) as a personal human tutor (VanLehn, 2011). Ideally, the system needs to understand the student's response and provide feedback to them. However, this process is simplified by comparing the student's response against an ideal response for the action they perform. For example, Dragoon tutoring system provides feedback of correct/incorrect after comparing every response to the correct solution for the problem (VanLehn *et al.*, 2017), on the other hand Autotutor parses the natural language based response provided by the student and looks for the keywords that are partly missing from the response to provide feedback for those important keywords (Graesser *et al.*, 2005).

The construction of an ITS can be divided into four modules – user interface, knowledge base, student model and pedagogical module (Dede, 1986). The user interface helps the student to interact with the system. Knowledge base includes the material for teaching and questions for helping the student practice the knowledge concepts. Student model holds the current model of the concepts that the student has already learned. Pedagogical module encapsulates the strategy for providing feedback or questions based on the student model. This structure has become a template for constructing different tutoring systems. A normal construction cycle for tutoring system involves understanding the student model to design the pedagogical strategy and finally develop the interface. In case if the pedagogical strategy is general enough,

then knowledge base can be developed for different topics (domains), thus showcasing some degree of domain independence. For example, Autotutor (Graesser *et al.*, 2005) was designed as dialogue based tutoring system and it has been used for teaching electronics, computers, physics etc. Similarly, Dragoon (VanLehn *et al.*, 2016) which teaches first order differential equations, has been used for teaching dynamic systems, electronics, etc. Although these systems have been able to achieve some degree of domain independence, it still needs to go a long way for the tutoring system to change the pedagogical strategy based on the student's responses. For example, changing from dialogue based pedagogy to simple correct/incorrect pedagogy for students who are expert in the material.

The behavior of an intelligent tutoring system can be divided into *outer* loop and *inner* loop (Vanlehn, 2006). Outer loop is responsible for choosing the relevant practice problem, ideally based on the concept student has learnt. Several strategies have been evaluated for asking different sequence of questions such as multi-armed bandits to maximize concepts learnt by the student (Clement *et al.*, 2013). Inner loop represents step-by-step evaluation of the student's work and providing feedback to support student while solving the problem. A lot of different techniques have been used to provide feedback to students such as correct/incorrect or provide hints to elicit the correct answer (Chi *et al.*, 2008). Decisions for both the outer and the inner loop of the behavior are internally decided by the pedagogical module of the tutoring system.

Construction of an ITS describes different places HAAI techniques can be applied to support both outer and inner loop decisions in predictable and explicable manner. These techniques depend on the model of the interface and the concepts that the student has to learn, which can be described by the tutor (Grover *et al.*, 2018a) or they can be learned through data of the behavior in the environment (Tian *et al.*,

2016). The model can be dynamically updated and based on goal-directed learning, specific strategies to improve student's learning can be applied.

1.3 HAAI and ITS for the Future – Contributions

As we head into the future, where we would want several robots and automated agents to support us in our daily activities and work with us like teammates by helping us in the activities which can be require complex decision making, thus, showcasing the need to create human-aware system to enter our world. They should incorporate our understanding in their decision making process. This thesis lays the foundation for human-aware framework for active teaming, where it connects the intelligent systems (i.e. the ITS) to this framework, and also describes how they should be extended to trickier scenarios in other domains. We overcome the challenges of "inmates running the asylum" (Miller, 2017), and evaluate the framework proposed through user studies. The outline of the thesis is –

Chapter 2 presents a literature survey of Intelligent Tutoring System and AI techniques that have been applied for teaching the students.

Chapter 3 discusses application of state-of-the-art human-aware planning techniques to tutoring scenarios such as providing feedback to students and support more recent phenomenon of online learning through dynamic curriculum.

Chapter 4 discuses the first user study to evaluate the framework described in chapter 3. The results show that HAAI techniques are effective in providing support to the human decision makers such as students while working on a sequential decision making problem.

Chapter 5 discusses a novel method to refine $\mathcal{M}_r^H$, by asking directed question(s). This problem is at the intersection of both human-aware AI and ITS, as asking questions for learning student's current model and support learning is part of the outer-loop and for HAAI it can support collaboration in human-robot teams.

Chapter 6 discusses the user study and results for evaluating the framework presented in the previous chapter.

Chapter 7 presents the concluding thoughts for how these communities can work together to create automated systems for collaboration and support the humans in their tasks.

Chapter 2

AI TECHNIQUES FOR ITS – A LITERATURE SURVEY

To understand the application of AI techniques, we review some of the fifty year literature of Intelligent Tutoring System. First we understand some of the basic theories that helped develop various methodologies for teaching. We also discuss some of the basic cognition models that have been applied to understand different aspects of learning. Then we present how state of the art ITS are being constructed and present some recent examples that have been used in several classrooms, followed by several student modeling techniques, used to model the learning of the student while interacting with a tutoring system. Then we summarize some of the AI techniques that have been used to provide feedback to students, or present questions to them. Finally we present discuss the challenges faced and some AI techniques developed to support recent techniques such as massive open online courses (MOOCs) and recent ideas like classroom orchestration.

2.1 Fundamental Theory

In this section we discuss some of seminal works that have shaped the current state of ITS. It includes one of the earliest classification for evaluating effectiveness of instruction, followed by more recent cognition model to better understand the connection between cognition and knowledge which was later generalized to the idea of knowledge concepts across different subjects. Finally, we present a model that has been used to understand the effectiveness of different modes of teaching.

Bloom's Taxonomy

– Bloom *et al.* (1956) proposed the first classification of educational objectives to define the expectations from students and help measure the result of instruction. The taxonomy divided the learning process into six major categories – *Knowledge*, *Comprehension*, *Application*, *Analysis*, *Synthesis* and *Evaluation*. Knowledge in a specific field provided to students was divided into three categories, i.e., the specifics, how to deal with the specifics, and different principles and theories in the field. Each category was further sub-divided to broadly explain the different types of knowledge that belonged to each category. The other five categories were related to various students' steps for acquisition of knowledge where comprehension to synthesis are considered the most important educational goals. From the student's perspective, the first step is comprehending the knowledge through translation, interpretation, and extrapolation. Then they apply the knowledge followed by the analysis of different elements, relationships, and organization principles. Synthesis represents producing or deriving new relations and operations. Finally, the evaluation process included both internal evaluation through evidence and evaluating through external criteria (Krathwohl, 2002).

Revised Bloom's Taxonomy

– If one looks closely at Bloom's taxonomy Bloom *et al.* (1956), one can realize it consists of two different dimensions – knowledge to be acquired by the student and the cognitive process for obtaining it. Thus, the original Bloom's taxonomy was revised to create these two dimensions (Anderson and Krathwohl, 2001). The knowledge dimension consists of four different kinds of knowledge that a student needs – factual, conceptual, procedural, and meta-cognitive. The cognitive process dimension was

divided into six categories (based on the five dimensions of original taxonomy) – remember, understand, apply, analyze, evaluate and create. Comprehension from the original taxonomy was divided into remember and understand, and the rest of the steps were similar in definition to the original taxonomy. More interestingly, these two dimensions could be arranged on a table with different topics should be placed. For example, a simple history lesson would be factual knowledge/remember, and solving a math problem could be placed in procedural knowledge/analyze (Krathwohl, 2002).

As discussed, Bloom's taxonomy (both original and revised) describes different knowledge and cognitive processes that a student should learn. However, there is a need to understand various activities in a classroom. Thus, we look at relatively new theories that go further to define both knowledge and cognitive process that student partakes in a classroom.

ACT and ACT-R

– Applied Cognition Theory (ACT) Anderson (1983) and ACT-Revised Anderson (1993, 1996) was designed to describe the connection between knowledge and cognition. The procedural knowledge consists of production rules; defined using declarative knowledge. The smallest unit of declarative knowledge was called *chunks*. Complex cognition is constructed using different production rules together. Authors divide the process of complex cognition in knowledge acquisition (acquiring the relevant chunks of knowledge) and using the knowledge-based on memory (Anderson, 1996). ACT-R theory of complex cognition was based on earlier views such as Human Associative Memory (HAM) (Anderson and Bower, 1973), to model several memory-based behaviors through experiments or the theory about the human method of problem-solving (Newell *et al.*, 1972). The authors directly applied these theories for teaching LISP (Anderson *et al.*, 1995). Recently, the schema structures based division of declarative

knowledge has been widely used in teaching algebra (VanLehn *et al.*, 2020), as well as constructing complex population dynamic models using tutoring systems such as Dragoon (VanLehn, 2013; Grover *et al.*, 2018b).

However, these theories have been able to describe only one way to represent knowledge, i.e., through template-like structures applied to topics such as LISP or algebra, where the combination of these rules forms complex applications. On the other hand, there have been theories that can be applied to other topics or even other activities in a classroom such as, hierarchical division of knowledge to constant and variable concepts and the kind of activity that supports learning in the classroom. We will now discuss these frameworks.

KLI Framework

– Knowledge Learning and Instruction (KLI) framework (Koedinger *et al.*, 2010) tries to generalize earlier theories across science, math, and language learning domains. They structured it around defining taxonomies across three essential parts of the classroom KLI by first defining different kinds of events that occur for a student – learning event, instructional event, and assessment events that occur while a student works on a topic. They distinguish between these events as observable (instructional and assessment) and unobservable (learning events). They explain that instructional events cause learning events and the knowledge components affect the assessment events.

The knowledge is divided into the smallest acquired unit called *knowledge components* (KCs). Koedinger *et al.* (2010) further divides the KCs into two different types – constant and variable. If the KC is a fact, such as the area of a circle is $\pi * r^2$, it is a constant KC, and variable KC would be asking a student to find the area of a circle with radius $5cms$. In this framework, they take student interaction into the

picture, i.e., recognizing the kind of KC and the response from the student. Thus, we get three different kinds of interaction – constant-constant, variable-constant, and variable-variable. Constant-constant KC would be in the case – what is the acceleration due to gravity (where students' response would be $9.8m/sec^2$). Variable-constant would be – what is the unit of volume. Variable-variable is to apply a hypothesis for different quantities. Constant-variable is not possible from a student interaction. However, I believe a tutor uses this kind of interaction to create a general understanding using examples.

Taxonomy for learning events is also divided into three types – memory and fluency-building, induction and refinement, and understanding and sense-making. As the name suggests, memory and fluency building is the first process where a student compiles the knowledge non-verbally. Induction and refinement is also a non-verbal learning process where students find connections between different knowledge components and refine these bigger structures. Understanding and sense-making is a verbal process where they apply KCs and understand their rationale. They further use these two taxonomies to define different instructional principles for every learning process. This framework connects theoretical ideas to real-world classroom teaching and provides taxonomies that can help improve the learning process.

ICAP Framework

– There are four different modes of engagement – Interactive, Constructive, Active, and Passive (ICAP). The framework evaluates these modes for their effect on learning in a classroom. Chi and Wylie (2014) describe these four modes of engagements to show theoretically and empirically that Passive < Active < Constructive < Interactive is the order of impact on learning. It only looked at evaluating a different mode of

engagements and assumed underlying knowledge and cognitive processes were similar to earlier frameworks Chi and Wylie (2014).

These frameworks divide the tutoring process into two critical dimensions (from Bloom's taxonomy), knowledge and cognitive task. These two dimensions have been separated into various categories, sometimes based on a domain-dependent or domain-independent fashion. This research dates back to the 1950s, whereas the automated tutoring system were developed for the first time in the 1970s. These theories form the baseline to many modern tutoring systems that will be discussed in the next section.

2.2 ITS – Construction and Types

Since the advent of computer systems, many computer-based tutoring systems have been constructed to provide individual tutors to students, such as computer-aided instructions, computer-based training, or web-based homework. Scholar is sometimes referred as the first tutoring system which provided CAI based on the task material provided by the tutor called frames (Carbonell, 1970). Vanlehn (2006), mentions these systems do not follow any pedagogical techniques to provide intelligent feedback to the students. On the other hand, Intelligent Tutoring System (ITS) (Anderson *et al.*, 1985) has a separate pedagogical module for providing feedback to students. It can be classified based on many different paradigms. I will present some of these classifications and then present some example ITS.

2.2.1 Modules of an ITS

An ITS can be divided into four modules – user interface, knowledge base, student model, and pedagogical module, based on its construction (Dede, 1986). The user interface helps the student to interact with the system. The knowledge base includes

the material for teaching and questions for helping the student practice the knowledge concepts. The student model holds the current model of the concepts that the student has already learned. The pedagogical module encapsulates the strategy for providing feedback or questions based on the student model. This structure has become a template for constructing different tutoring systems. A typical construction cycle for the tutoring system involves understanding the student model to design the pedagogical strategy and develop the interface. If the pedagogical strategy is general enough, then the knowledge base can be developed for different topics (domains), thus showcasing some degree of domain independence. For example, Autotutor (Graesser *et al.*, 2005) was designed as a dialogue-based tutoring system, and it has been used for teaching electronics, computers, physics, etc. Similarly, dragoon (VanLehn *et al.*, 2016) which teaches first-order differential equations, has been used for teaching dynamic systems, electronics, etc.

2.2.2 Knowledge Representation

As discussed, ITS provide intelligent feedback to students, and the tasks represent the internal knowledge of the system. ITS has been divided into two broad categories – model tracing and constraint-based tutoring systems, based on how the student can work with the interface. *Model tracing* compares the students' solution of the assignment to the correct solution provided by the tutor. For example, Dragoon (VanLehn *et al.*, 2016) and TopoMath (VanLehn *et al.*, 2020) are model tracing tutoring systems, and the student's responses are compared to the correct model of the task. *Constraint based systems* evaluate the student response to ensure every constraint is correctly satisfied in the response. For example, SQL tutor (Mitrovic, 2003) is a constraint-based system, where every response from the student is evaluated to test if the student's response satisfies the constraints for specific queries.

2.2.3 Behavior of an ITS

The behavior of a tutoring system represents its reasoning capability and the way a student can interact with it. Various decisions taken by a tutoring system can be divided into two stages – *outer* loop and *inner* loop (Vanlehn, 2006). The outer loop involves deciding the sequence of tasks that a student performs. Several strategies have been evaluated for asking a different sequence of questions, such as multi-armed bandits to maximize concepts learned by the student (Clement *et al.*, 2013). Inner loop represents the step-by-step evaluation of the student's work and providing feedback to support the student while solving the problem. Now I will discuss some of the critical decisions taken in each loop individually –

Outer Loop

As described earlier, the outer loop is responsible for providing practice problems to students.

Task Selection Usually, the task sequence is manually defined by the teacher. Recently, many different strategies have been tried to present problems to students. For example, providing them a list of problems to complete (in no specific order) and letting them choose the next problem, or, ensure mastery of the concept for students called mastery learning and then go to problems for next concept (Bloom, 1984), or maintain a detailed model of KCs mastered by the student to and choose the task based on the intersection of the mastered concepts and the concepts that the student is supposed to practice (macroadaption) (Anderson *et al.*, 1995; Shute, 1993). Macroadaption has been applied in many different ways, which we will be discussing in some detail in the later section. However, various factors has to be kept in mind during task selection (Emre, 2020) –

- Zone of Proximal Development – the student is neither bored nor struggling in solving the new problem, i.e., they can progress while solving the problem (Lee, 2005).
- Spacing and Sequencing effect – space between two similar tasks (with similar kind of concept application) is relatively far apart.
- Relational graph of concepts – the relative dependency between concepts is maintained.

Here we have highlighted the critical factors for automatic task selection. The exact techniques are discussed after discussing the student model.

Inner Loop

As the name suggests it runs on every step that a student performs, where the student's step is assessed and feedback or hints are provided to the student.

Feedback is provided at every step while solving a problem. Two different kinds of feedback can be provided – (1) minimal feedback and (2) error-specific feedback. Minimal feedback refers to providing binary or simple feedback whether the step or the problem has been solved correctly. For example, Dragoon (Wetzel *et al.*, 2017) provides green/red (correct/incorrect) feedback on each step. Error-specific feedback usually points to the exact step or part of the response that is incorrect. For example, Andes (Gertner and VanLehn, 2000) suggests checking the specific part of the response (such as trigonometric equation) to the student. A central challenge for the tutoring system is to find when to provide feedback to the student. Based on when the feedback is provided, ITSs can be divided into three major categories –

- Immediate feedback – The feedback is provided as soon as the student completes a step of the problem.

- Delayed feedback – Usually, it refers to providing the feedback at the end of the problem. For example, in Sherlock (Katz *et al.*, 1998) student is provided the feedback in review mode after they have completed the problem and submitted their attempt.

- Demand feedback – The feedback is provided when the student explicitly asks for it.

There can be many complex strategies to provide feedback, such as delaying the feedback for a student when it is nearing mastery of a concept, whereas someone who is practicing a new concept is provided feedback immediately. In many instances, these feedback strategies are modeled based on the tutor's behavior (Collins *et al.*, 1988). On the other hand, error-specific feedback is presented in many natural language-based tutoring systems.

Hints are provided when the student has provided a wrong answer to the step, or they are stuck at what step to execute. The central challenge to provide hints to students is when the system should provide a hint, what step it should provide a hint on and how it should provide hint (Vanlehn, 2006). Some of the factors for when to hint depends on whether the student is stuck with the step. It is suggested that the hint should be provided for the next step that the student is trying to complete.

There has been much work to define various strategies for how to provide hints to students. Generally, the idea is to provide a general hint and then move to more specific hints, with the final hint providing the student's answer. The final hint is many times called the bottom-out hint. It ensures that the student continues

progressing through the problems. In other cases, different strategies are applied based on the student's response to the problem. For example, in AutoTutor (Graesser *et al.*, 2005) the system provides a hint when the student provides an incomplete answer to the problem in the form of asking questions from them. There has been some work to find an optimal strategy for tutoring action or the kind of hint that should be provided to the student (elicit/tell) (Murray *et al.*, 2004; Chi *et al.*, 2010).

Knowledge Assessment is useful to the instructor, student, and the tutoring system. The instructors or a tutoring system can use the assessment to define the next task or as a heuristic to improve students' learning in some capacity, whereas a student can use it for self-evaluation. However, the fundamental challenge for assessment is its grain size. For example, a final grade at the end of the semester does not provide enough information about the student's topics, and a fine-grained assessment of every step being reported might not help the teacher a lot either. Thus, a general strategy for assessment is to count the learning events for the concepts or even count the failures (Vanlehn, 2006). More involved methods are part of student modeling, which will be discussed in the next section.

Gaming the System refers to shallow techniques that a student may choose to solve the problem and finish the unit. Such techniques involve help abuse, where a student might choose to provide some incorrect very quickly to reach the bottom-out hint and use that to progress in the problem. This kind of student behavior is studied in the literature (Muldner *et al.*, 2010), and many strategies are applied for understanding the causes of it (Baker *et al.*, 2009). Baker *et al.* (2009) found that unhelpful hints and un-intuitive software features led to students gaming the system. Muldner *et al.* (2010) analyzed the log data for Andes tutoring system while

teaching physics found that up to 22.5% tutor-student turn pairs were gamed out of 18.4% actions were gaming of hints. They further evaluated that the time spent on high-level hints was 9.2 seconds vs. 5.7 seconds for the students gaming the system. Similarly, students with a higher tendency for gaming the system spent less time on the bottom out hint (10.9 seconds vs. 7.5 seconds).

2.2.4 Examples of ITS

In this section, I will look at some of the examples of Intelligent Tutoring Systems.

Algebra Cognitive Tutor is one of the most widely used tutoring systems used to teach algebra (Anderson *et al.*, 1995). The inner loop consists of a student solving a problem using multiple representational tools, such as an equation solver. The outer loop follows the complex strategy based on the student model, and exercises are selected to ensure enough practice for each KC. The tutor analysis each step of the student by comparing it to anticipated steps using a rule-based problem solver. Hints are provided when a student asks for it, and the tutor chooses what step to hint at.

AutoTutor is a natural language dialogue-based tutor where a human head talks to a student like a teacher. The tutor presents a task (a single question) to the student, and they respond with an essay-like answer to the question Graesser *et al.* (2005, 2012). At every step for the inner loop, the student types the answer or asks a question (which usually means that the student is asking for help. Step analysis happens by matching the student's response to its list of possible correct-incorrect responses. The tutor also has a set of knowledge components that have to be correctly

applied for task completion. Once a student's answer applies all of them, the task is completed, and the next task is chosen from a list of tasks.

SQL tutoring system teaches writing a query for relational databases (Mitrovic, 2003). It is a constraint-based tutoring system, where a constraint consists of a relevance condition and satisfaction condition. The tutor ensures for all true relevance conditions corresponding satisfaction condition is also true in the student's response, else the constraint is violated. The student usually has to submit the query and ask for feedback from the tutor. If multiple constraints are violated, then the tutor chooses the important constraint to provide the hint.

Dragoon is a model-tracing tutoring system is used to teach dynamic model construction (based on first-order differential equation) (Wetzel *et al.*, 2017; VanLehn *et al.*, 2017). Each task consists of constructing a dynamic model where the step requires defining different quantities and relationships (interaction) between them. The simplest interaction between basic quantities is called schemas, and they combine to solve the problem. The system provides immediate feedback with three levels of hint, where the third hint adds the correct value in the interface. The outer-loop consists of a pre-defined sequence of models that constructs real-world scenarios such as modeling the rabbit and wolf population, with the relationship between quantities become more complex with every problem.

Andes is a physics tutoring system, where a student solves a physics problem using tools provided in the system and enters the final answer while showing all the steps to solve the problem (VanLehn *et al.*, 2005). In the outer-loop, the student selects the problem, and the inner-loop consists of the student performing various kinds of steps such as entering the equation or defining a vector, etc. The tutor provides minimal

immediate feedback (red/green incorrect/correct). Students can initiate two different kinds of help – ask for a hint for the next step or ask for step analysis (the system will provide error-specific feedback if the step is incorrect). Step evaluation compares the student response to valid responses in the tutor's model.

Several tutoring systems do not tend to follow these kinds of structures. **Game based** tutoring systems are based gamine environment where a student plays a character in the form of a game (sometimes as a student himself that can be with or without other students) tries to solve the game and earn rewards (Jackson and McNamara, 2013, 2011). It helps develop skills that require long-term interaction. The student gets hints from the environment it has to use while solving problems, earns rewards/trophies, unlocks new features, etc., to keep them motivated. Physical robots have also been used to keep younger students involved, and there has been some research on their effectiveness in tutoring (Walker and Burleson, 2012). More recently, virtual reality-based interfaces have been used to support students in a classroom (https://www.prismsvr.com/, start-up Brett is working with, they have launched courses for teaching Maths this Fall for schools in New York). There are some open-ended tutoring systems as well in which the students learn through experimentation, and there are multiple ways to complete the same problem, such as virtuallabs (Scheckler, 2003).

Vanlehn (2006) suggests that the student should be given feedback or hint based on the student's approach. Although most of these tutoring systems have a single solution to a problem, however, there are a few that provide freedom to students to solve the problem in a free approach, such as, Andes (Gertner *et al.*, 1998; VanLehn *et al.*, 2010) or Virtuallabs (Scheckler, 2003). There has been some work in detecting what the student is trying to do in lab based open systems, where the authors used plan libraries to find the plan and the goal that the student is trying to

achieve (Mirsky *et al.*, 2017). Our work (Grover *et al.*, 2018a) on integrating human-aware planning techniques with Dragoon, make use more recent out-of-the box plan recognition methods (Ramírez and Geffner, 2010) for the same purpose.

2.3 Student Modelling

The inner loop of an ITS involves assessing the student's knowledge. These systems tend to model and maintain the current state of the student's knowledge, which can be further used for assessment and pedagogical strategies. The models are constructed using the student's activity and responses. Pavlik Jr *et al.* (2013) divide these models into – programmed models, overlay models, knowledge space models, dialogue student models, state and trait identification models. Most of these models are for different types of assessment and do not model the student's skill. For example, programmed models were constructed to create a sequence of tasks for the student to practice (Pavlik Jr *et al.*, 2013). Similarly, dialogue student models were tutoring systems based on natural language dialogue with students, such as, Why-Atlas (VanLehn *et al.*, 2002) or AutoTutor (Graesser *et al.*, 2012, 2005). On the other hand, overlay models model the student's understanding of the KC using some Q-matrix, which maps the problem to KC required to solve the problem. It divides the dependency structure of the domain and learns the parameterized model through the model fitting. We will be talking about some of these techniques that model students skill –

2.3.1 Item Response Theory

IRT is one of the oldest models which falls under the traditional psychometry methods for latent variable testing (Harvey and Hammer, 1999; Lord, 1952). IRT is computationally expensive compared to Classical Testing Theory methods (CTT).

However, it incorporates many extra features compared to old CTT methods. The chance of correctly answering the problem is calculated using learned parameters. It assumes that the student's response to a question depends on their ability and difficulty of the question. Since it is used for testing, when no feedback is given to the student, IRT is assumed to be a static model and does not model the change in the student's knowledge. The probability that a student would respond to a query is calculated as logistic regression with parameters for student ability and the item difficulty (Wilson *et al.*, 2008; Rasch, 1993). Performance Factor Analysis (PFA) (Pavlik Jr *et al.*, 2009) uses extra learnable parameters to incorporate the number of times student has practiced a KC. PFA has shown comparable results to BKT (Pavlik Jr *et al.*, 2009).

2.3.2 Bayesian Knowledge Tracing

On the other hand, the learning community assumes that student knowledge for a specific knowledge concept exists in two states – mastered (1) or not-mastered (0). A posterior probability is calculated when a student solves a problem involving the KC, using the prior probability of mastered and the correct and incorrect response. If the student keeps practicing, the student can transition from one state to another with transition probability and guess the answer or make a mistake based on emission probabilities. If we assume these transition and emission probabilities stay constant over time, then the knowledge tracing method can be represented using a hidden Markov model (Corbett and Anderson, 1994). BKT also assumes that the application of every KC is independent, and the transition and emission probabilities vary due to different KCs.

An important point to note is that IRT and BKT are based on orthogonal assumptions. However, they model the chance of students solving the problem cor-

rectly. Thus, there have been many attempts to merge these two models with some success (Khajah *et al.*, 2014a; González-Brenes *et al.*, 2014; Xu and Mostow, 2012). González-Brenes *et al.* (2014), discusses extending to more general features such as sub-skills, and uses the values as logistical regression to represent emission probabilities or transition probabilities. It found that emission probabilities were assumed to be constant. These models face two major challenges, learn the parameter values and different parameters of the models giving rise to the same predictions about student performance (called identifiability) (Beck and Chang, 2007; van De Sande, 2013). There have been many Expectation-Maximization (EM) based approaches to learning the parameter values, such as Feature-Aware Student Tracing (FAST) (González-Brenes *et al.*, 2014) and Latent Factor Knowledge Tracing (LFKT) (Khajah *et al.*, 2014a). There is a toolkit to learn the model parameters for vanilla, HMM using Bayesian Network toolkit (Chang *et al.*, 2006; Xu and Mostow, 2011). There are other monte Carlo approaches with different sampling methods as well (Khajah *et al.*, 2014b). Khajah *et al.* (2014b), found that the combined networks had statistically significant improvement for Area Under the Curve (AUC) values for combined models in 3 out of 4 datasets.

Identifiability problem in BKT is a well-known problem and refers to the scenario that multiple BKT parameters can have similar predictions about student performance (Beck and Chang, 2007; van De Sande, 2013). Doroudi and Brunskill (2017), discusses that the problem of identifiability is conflated and misunderstood for BKT. They suggest that BKT models are identifiable under the condition $P(G) \neq 1 - P(S)$ (Doroudi and Brunskill, 2017) and $P(G) + P(S) < 1$ van De Sande (2013). Further, Doroudi and Brunskill (2017), discuss that the more important problem is lack of semantically degenerate values learnt by the model. If you notice the condition

$P(G) + P(S) < 1$ is a stricter condition than $P(G) \neq 1 - P(S)$, and I believe that the methods should learn semantically viable solutions.

2.3.3 Deep Knowledge Tracing

Deep knowledge tracing (Piech *et al.*, 2015) refers to modeling time-series data (student's responses) using Long-Short Term Memory (LSTMs) (Hochreiter and Schmidhuber, 1997) a variant of Recurrent Neural Network (RNN) (Williams and Zipser, 1989). Encoding the input for the neural network-based method is the central challenge. Piech *et al.* (2015) uses one-hot encoding to set up the input from the student, which includes the question answered by the student and whether the response was correct or incorrect. The output represents whether a particular student answered the problem correctly at a given time. Surprisingly, the Deep learning method performs well for the given data and shows higher AUC values as compared to both BKT and PFA (Piech *et al.*, 2015). I believe some of the biggest challenges for deep learning methods are in comparison to BKT, where hidden states represent whether a student has mastered the KC or not. However, hidden states in LSTMs are encoded in a special encoding space and have no physical meaning. Thus, they lack degeneracy claims for students, and we are dependent on the output alone.

Another critical point to note is that Piech *et al.* (2015) have also used large datasets for 47,000 students working with khan academy. This kind of data is not available for learning parameters in many cases. I believe the third challenge that they face is, the input is the one-hot encoding of individual activity and not a skill or a KC. Thus there system is not extendable to specific skills and combinations of the skills. Xiong *et al.* (2016), have tried to reproduce the previous results with smaller datasets, however even they have not extended the ideas to skills.

2.4 Use of AI Techniques in ITS

AI is used to create Adaptive Learning Technology (ALT) (Aleven *et al.*, 2016). ALT represents the techniques for modifying the interface in some form based on the previous data from the students. These techniques support three different levels of adaptivity – (1) design level adaptivity, (2) task level adaptivity, and (3) step level adaptivity. I will be presenting the use of AI technology on two levels, i.e., task level and step level adaptivity. The student model is helpful for all these steps, and the AI techniques used for student modeling have been discussed in the previous section.

2.4.1 Task Level Adaptivity

A tutoring system has to assign the task to students. Ideally, the system should ensure the task chosen is in the zone of proximal development, maximizes learning, is personalized based on the student's current model, etc. In many cases, these tasks are decided by the tutor to fit an average student, and these tasks are not well suited for the least and most skilled students (Lee and Brunskill, 2012). Mousavinasab *et al.* (2021) discusses 53 different studies that use several intelligent techniques, out of which only 1 has task level adaptivity, which is also a misnomer for interactive activity, which can be chosen based on the topic and not suggested by the tutoring system. However, there is some work in automated task selection based on Multi-Arm Bandit techniques (MAB) (Bubeck and Cesa-Bianchi, 2012).

Clement *et al.* (2014) discusses two different methods for dynamically selecting the practice problems. The authors define a set of parameters for each problem and an R-table representing the level of knowledge required to solve the problem (value between $(0, 1)$). They use this value to update the student's capability based on their response (correct or incorrect solution). Then they use the student model values to

dynamically find the next problem based on the utility of change. This algorithm was called RiARiT (Right Activity at the Right Time). Evaluating RiARiT for a class showed promising results, as it either led to modeling the problem KCs for students or the students (with a higher level of understanding) progressed faster through each unit faster (Clement *et al.*, 2013). They also developed where they did not update the student model and directly calculated utility without using the student model based on the zone of proximal development and empirical estimation, called ZPDES (Zone of Proximal Development) (Clement *et al.*, 2014). Evaluation using simulated students showed statistically significant results compared to pre-defined order. Since ZPDES did not use the student model, it performs worse in comparison to RiARiT (Clement *et al.*, 2014). Mu *et al.* (2017) improved the ZPDES algorithm to use the student model (in a similar manner as RiARiT) and analyzed it with simulated students. They found that the method performed better than the expert-defined problem definition for simulated students.

There has been some work to create a deep learning-based task selector (Emre, 2020). An LSTM (improved RNN) was used to find the next question for the students. It was tested using an online course for organic chemistry, with 3000 problems. The mapping of KCs and the problem was provided by tutors (or subject matter experts). The baseline for the evaluation was the questions selected by the students, and their comparison showed that the LSTM based question suggestions were more accurate in predicting the next helpful question and whether the student would be able to answer the question correctly.

There have also been some POMDP based approaches to decide the activity or task for the student (Rafferty *et al.*, 2011, 2016). They model the student as a partially observable state, and the tutoring system is the acting agent. In some cases, the policy is calculated using decision-theoretic methods for the single-step

look ahead method. They create three different kinds of policies memoryless where there is no idea about the student model, and thus there is a fixed policy here which is just based on the utility (which could mean that it maximizes the number of topics learned in each step), with memory but discrete model, and with memory with the continuous model. The control is random policy. The evaluation shows continuous model performs a little better than the discrete model, and every model beats the random model. However, I cannot entirely agree with the random policy, as I believe the control should be the static teacher's policy. Then it would give a comparison of whether the task level optimization affects learning for students who need more support. However, this evaluation was performed by Mu *et al.* (2017) which did not include the teaching material. They were able to show that the personalized practice tasks for students performed better for simulated students as compared to the expert policy.

There has been some work for automated question generation and automated answering. These research topics fall under natural language processing to generate questions from paragraphs (Zhang, 2015) or improve responses using data such as Winograd challenge (Levesque *et al.*, 2012). On the other hand, question generation has had some research in ITS literature to generate questions that can push students to have a more profound thought process. Such questions are generated by using a semantic network to represent the relationship between the concepts. Zhang and VanLehn (2016), evaluates such questions for fluency, relevance, ambiguity, pedagogy, and knowledge depth. Their evaluation shows that students have found these questions to be similar to the textbook. However, tutors found that these questions lacked topic depth compared to expert-generated questions. Zhang and VanLehn (2017) evaluated question generation and presenting them based on the student model, and it showed that adaptive question selection for automated generated questions shown

higher learning gains. More recently, Pan *et al.* (2020) has tried to tackle the problem of generating deeper (neural network-based) questions that require higher analytical skills from students. They use attention-based neural networks to generate questions and evaluate them for fluency, relevance, and complexity to show improved results compared to baseline.

2.4.2 Step Level Adaptivity

A student's step is the single-step interaction between the student and the interface to solve the problem. Step level adaptivity means the changes that the tutoring system makes based on the different student responses. A tutoring system can take many steps after every response, such as, provide feedback or hints for the next step. Mousavinasab *et al.* (2021) presents 53 different studies for tutoring systems, out of which around 50% had some degree of step level capabilities to provide intelligent feedback or hints. The tutoring system ensures that a student does not get stuck with the problem and understands the KC, and retains it to apply in the future.

Step level adaptivity involves two stages – (1) understanding the student input and (2) deciding the tutoring system's response. AI techniques have been used in both the stages, such as AutoTutor uses NLP techniques to understand the student response (Graesser *et al.*, 2012), and decision-theoretic approaches to decide the response of the tutoring system (Murray *et al.*, 2004). I will only discuss the AI techniques for decision-making employed to calculate the tutoring system's policy. Generally, there is a static policy to provide feedback and hints to students, such as, provide correct/incorrect feedback and, if incorrect provide incrementally abstract to more pointed hints, where the final hint provides the answer to the student (called bottom-out hint) (VanLehn *et al.*, 2016, 2010).

Murray *et al.* (2004) uses decision-theoretic (DT) methods to choose the tutoring system's action from – prompt, hint, teach, or do nothing using positive feedback, negative feedback or do (explain how to do the step). The authors created a detailed student network, which provided the concepts that the student had practiced and tracked the state of the problem (by tracking the possible actions of the student). They calculated the utility of each action by single-step lookahead. The transition parameters were set using expert knowledge, and the highest utility action was executed. They applied the approach to teaching calculus and reading using two different interfaces. They evaluated the system to understand the behavior and found it rational and sensitive to the internal weight changes. They were also able to verify that the system can emulate the behaviors of the tutor. Other systems used decision-theoretic approaches, such as the first version of Andes (physics tutoring system) used DT approaches to decide the next step for the students. However, their evaluation in comparison to tutors did not match their predictions for various reasons (Gertner *et al.*, 1998). iTutor tutoring system for teaching Newtonian mechanics also used dynamic bayesian networks deciding the problems to present (Pek and Poh, 2004).

Chi *et al.* (2008) used reinforcement learning techniques to choose the type of feedback (elicit or tell) to provide students or execute a self-explanation step (ask the student to justify the step) while working on a natural language-based physics tutoring system. They used model-based approaches to learn the transition function by executing random actions with students (data was collected from students directly). The state was defined based on 6 (ternary) features, and there were two actions in each state – Elicit/Tell or Justify/Skip-Justify. They tried two different kinds of reward functions (normalized gain – normalized change of post-test to pre-test), dichotomized gain (+100, -100), exploratory (random). They were able to show that out of the policies evaluated for each reward function; normalized gain had the

highest learning gains. There have been many other RL-based approaches in which the data was collected using simulated students, and their application to real students showed favorable results (Beck *et al.*, 2000; Iglesias *et al.*, 2009).

2.5 Recent Developments

MOOCs and social learning – A growing population has brought two most significant challenges to the fore, providing individual attention to students and improving absorbing or retention of knowledge. Massive Open Online Courses (MOOC) (Pappano, 2012) were constructed to provide personalized attention in mind where students can learn through social interaction across a variety of platforms and participate in the learning process as a *community*. This is the Learning 2.0 paradigm Seely Brown and Adler (2008), and requires a rethink of the affordances McLoughlin and Lee (2007) expected from current learning tools. One of the many advantages of social or online platforms for learning is peer feedback and community participation – i.e., social learning (Burke, 2011). This involves two critical aspects – *knowledge advancement as a community* (Scardamalia and Bereiter, 2006) and *information processing* (Webb, 2013) on the part of the individual student as a member of that community.

MOOCs face many challenges in retention. Online courses that were started had a course completion rate of 10% (Hone and El Said, 2016). There has been some research to understand the factors affecting the low completion rate. Hone and El Said (2016) tested different hypothesis for content and interaction among 379 students. They found that the course content can explain 79% of the variance in retention and its perceived effectiveness and the interaction with the instructor. Moreover, they found no significant difference in the completion rate based on gender, level of study, or the MOOC platform.

Classroom Orchestration – ITS has always been imagined to accompany the classroom setting. A tutor explains knowledge concepts for the unit and uses a tutoring system for the practice material. The process of holding such classroom activities with or without the help of technology is called classroom orchestration (Dillenbourg and Jermann, 2010). The tutoring system supports the teachers by learning fine-grained student models. However, another way to support them is to provide a real-time update for the class progress in the classroom, or call the tutor in case a student is stuck (preferably intelligently) (VanLehn *et al.*, 2019).

Jigsaw classroom – The idea of dividing students into smaller groups to work together on tasks is called a jigsaw classroom (Aronson, 2002). Based on increasing the randomness among groups and suggested ten steps to creating better groups (https://www.jigsaw.org/#steps). It has been shown that constructing such groups can help to build compassion among students and improve academic results (Perkins and Tagler, 2011). Even we have tried to construct groups of students with different levels of understanding for the concepts that can support students teaching each other. We show empirically that such groups can help in faster progress among students Grover *et al.* (2018a).

In this chapter we have discussed how several techniques applied to model the student's learning and change in their model of understanding. Automated task planning has a similar definition of the model, where it attributes to the change in current state learning only through the actions a student would perform on the interface. It can track different techniques that a student can apply while solving a problem. We further explore this idea in the next chapter of the thesis.

Chapter 3

WHAT CAN AUTOMATED PLANNING AND HAAI DO FOR ITS?

In the earlier chapters we introduced automated task planning, and human-aware AI framework. In this chapter we present this formal approach applied to ITS. This chapter is divided into four sections. First, we introduce different aspects of computer based learning where automated planning can be useful, then we look at these techniques formally to explain what these techniques are and what are the assumptions. Then we describe application of these techniques to model the behavior of a tutoring system called Dragoon, and also model a classroom to provide support to the teacher. Finally, we showcase specific examples of these techniques from Dragoon, and other parts of the classroom.

3.1 Introduction

While the last decade has seen massive advances in technologies aimed at creation and dissemination of knowledge across a variety of platforms, concerns remain as to how effectively this knowledge is absorbed at the user (student) end. This is especially true for both massive open online courses (MOOCs) and also for (rapidly growing sizes of) physical classrooms where targeted attention towards individual students is often hard to provide. The state-of-the-art in student and instructor support technology has traditionally struggled to catch up with the demands of the rapidly evolving landscape of education in the 21^{st} century. In this paper, we address this by proposing a framework for the design of generic course-independent student and instructor support capabilities using techniques in the field of human-aware planning, and demonstrate those features in `Dragoon`, a celebrated intelligent tutoring system.

3.1.1 Learning 2.0

The world of learning is indeed changing fast - information can now be provided across a variety of platforms to large groups of people who can access *on demand* knowledge and participate in the learning process as a *community*. This is the Learning 2.0 paradigm (Seely Brown and Adler, 2008), and requires a rethink of the affordances (McLoughlin and Lee, 2007) expected from current learning tools.

Learning on Demand

Learning on demand refers to the increasing popularity of individual student-centric and topic-driven learning achieved on the web – i.e. students pick a particular topic they want to learn about and actively consume content just based on that, instead of participating in an entire class or following through an entire curriculum. For example, consider that you want to learn about regression – you could log on to Coursera, complete the relevant tutorials and assignments on regression, and leave the course. This requires a rethink of traditional curriculum generation and course recommendation approaches that would traditionally compute end to end curricula for an entire class. It follows that such new approaches must be able to leverage detailed student models to provide effective support.

Social Learning

One of the many advantages of social platforms for learning is peer feedback and community participation – i.e. social learning (Burke, 2011). This involves two critical aspects – *knowledge advancement as a community* (Scardamalia and Bereiter, 2006) and *information processing* (Webb, 2013) on the part of the individual student as a member of that community. In a sense, this can even be seen as a proxy towards pro-

viding individual classroom attention from the instructor. However, forming study partners remains an arduous task, especially in large classrooms such as in online learning communities where students usually do not know most of their classmates (or their skill sets). It is also fraught with the usual pitfalls associated with group work including individual students hogging all the group activity or slackers not contributing to the group activity at all (Mesch, 1991). Without principled drivers for building in-class communities that can promote learning, effective collaborations are hard to achieve. As such, forming useful teams for collaborative study can become a problem by itself rather than a facilitator for learning to the extent that students can end up spending too much effort in forming and maintaining teams or just prefer to study by themselves, thus leaving the potential benefits of a social learning environment largely untapped. Recent work has shown that peer recommendations can have positive impact (Labarthe *et al.*, 2016) on student engagement but has remained ambiguous (Bouchet *et al.*, 2017) as to the best way to go about it.

3.1.2 A Brief History of ITS and AI

ITSs are aimed to provide personalized support to students and bring in expert (human) tutors in the loop wherever necessary, thus reducing the burden on the instructor as well as improving the learning experience of the student. In fact, it has been shown that when designed correctly, an ITS can be as effective as a human teacher (VanLehn, 2011). A thorough description of the different components of ITSs can be found in (Vanlehn, 2006). Existing applications of such systems range from solving numerical problems like Andes (Gertner and VanLehn, 2000) which can help in teaching basic laws of physics (Schulze *et al.*, 2000), Dragoon (VanLehn *et al.*, 2017), Q&A type problems as in Autotutor (Graesser *et al.*, 2005) or for an SQL tutor (Mitrovic, 2003). ITSs, of course, go beyond individual information processing

stage and find uses in knowledge building as a community (Magnisalis *et al.*, 2011) as well, thereby embracing the principles of the Learning 2.0 paradigm.

Student Assessment Models

One of the most important capabilities an ITS needs to have is to be able to estimate the (mental) model or capabilities of the student. This has been explored in the context of the (1) *item response theory (IRT)* (Hambleton *et al.*, 1991) which treats learning and testing as separate processes and the (2) *Bayesian knowledge tracing (BKT) theory* (Corbett and Anderson, 1994) which considers a more dynamic model of the student state. The latter becomes more relevant in the context of ITSs that can provide more dynamic feedback and hints as discussed next. Indeed this is an issue where AI techniques have been deployed before for dynamic modeling of the evolution of the student model in terms of knowledge components, concentration / focus levels, etc. (Murray *et al.*, 2004). This includes different techniques such as *decision theoretic approaches* (i.e. Markov Decision Processes or MDPs) (Murray *et al.*, 2004; Murray and VanLehn, 2006), and *reinforcement learning* (Chi *et al.*, 2010; Mandel *et al.*, 2014; Mandel, 2017). This paper assumes for the most part [1] that these techniques are available and builds on top of that assumption, i.e. being able to estimate the student model is necessary for ITS techniques and we want to demonstrate, from the perspective of automated planning how this can be exploited to provide a better learning experience to a student.

[1] In fact, the "model reconciliation" technique discussed later can handle uncertain models (Sreedharan *et al.*, 2018) and can even be modified to function as an estimator for the student model but this is outside the scope of the paper.

Feedbacks and Hints

Once the ITS has estimated a model of the student, it can provide targeted feedback to improve the learning process. Existing work in this area (Barnes and Stamper, 2010; Stamper *et al.*, 2013; Rivers and Koedinger, 2013, 2017) has largely focused on ITSs operating as *recommender systems*. This paper is largely situated in this space but aimed at providing much more sophisticated feedback in both the inner and outer loops (Vanlehn, 2006) of an ITS which requires longer-term sequential reasoning.

3.1.3 What can Planning Bring to the Table?

Automated planning, as a field, has been around ever since the inception of AI, and is considered a necessary ability of any autonomous system – the ability to reason about and decide on a course of action (CoA) or *plan* given the current state of the world. Many of the challenges faced in the design of an ITS bears parallels to the planning agenda – making a curriculum, solving a given problem, or in general dealing with the combinatorics of orchestrating a class can be potentially seen through the lens of planning, i.e. computing a sequence of steps given a set of constraints. This was the starting point of our investigation in this direction.

However, when operating with humans in the loop, traditional planning techniques are not sufficient (Kambhampati and Talamadupula, 2015). A "human-aware" planner must be able to take into account the (mental) model (Chakraborti *et al.*, 2017a) of the user. Recent work (Sengupta *et al.*, 2017) has looked at how planning techniques can evolve in the context of *decision support* to guide the planning process of a *human* decision-maker. This includes support for plan validation, critiquing, recommendation, explanations, and so on. Much of the discussion here derives inspiration from recent advances in the planning community along these directions.

Contributions

Thus, to answer the question what automated planning can do for the ITS scene, we build on the following two features of planning techniques –

- *Domain independence* – Planning techniques have been particularly geared towards domain-independent solutions – i.e. algorithms that can work across a variety of domains provided in higher-level specification. This is especially useful in the contexts of ITSs which have traditionally been restricted to class or course specific solutions that do not generalize; and
- *Model-based reasoning* – Personalized support for students require higher level and sequential reasoning about the course and *student models*, planning techniques remain ideally suited for this.

In this paper, we expound on the above two themes to –

- Provide targeted feedback when students are stuck on problems by leveraging the student model; (Section 3.3.2)
- Compute *on demand* curriculum based on class materials requested by the student; (Section 3.3.3)
    - We will show how this technique can be used to teach concepts to a student to attain different levels of expertise as desired by the student; and
    - We will show how student models may be composed to form joint plans of study.
- Generate class curriculum in the spirit of social learning by including fellow classmates in a student's curriculum while also guaranteeing desired properties

of the curriculum – e.g. that students not only learn but also apply all the concepts at least once. (Section 3.3.4)

We do not, of course, set out to model the full scope of challenges [2] in building and end-to-end ITS. However, we recognize that much of the existing work on deploying ITS systems, if not in conceptualizing them, has focused on specific learning platforms or courses without any coherent approach or general principles of design and implementation of the roles usually attributed to ITSs. The aim of this paper is thus to introduce techniques from the planning community that can formalize some of these concepts and provide a generalized framework for building such systems from the ground up. This has useful implications for both the planning as well as the educational technologies communities – i.e. the former can provide solutions to existing problems in ITSs (as we demonstrate in this paper) while feedback form the learning community can provide useful feedback towards the refinement of said techniques, including defining new areas of research of mutual interest. The biggest advantage of such an approach, as mentioned above, is that the techniques are domain-independent, i.e. they are defined at the procedural level and can be grounded with the description of a particular course as specified by the instructor. Of course, the problem of knowledge representation is (for a specific course and assignments in it) remain a challenge, but the ITS features themselves generalize given the proposed planning framework.

3.2 Background

In the following, we will introduce concepts from the planning literature that will be used in the rest of the paper.

[2] For example, the current discussion only focuses on the learning and interaction phase and does not include post-hoc reflection / evaluations as explored in (Katz *et al.*, 2000, 2003; Connelly and Katz, 2009)

A Classical Planning Problem (CPP)

(Russell and Norvig, 2003) is the tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, and A is a set of actions - and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$. Action $a \in A$ is a tuple $\langle c_a, \mathit{pre}(a), \mathit{eff}^{\pm}(a) \rangle$ where c_a is the cost, and $\mathit{pre}(a), \mathit{eff}^{\pm}(a) \subseteq F$ are the preconditions and add/delete effects, i.e. $\delta_{\mathcal{M}}(s, a) \models \bot$ *if* $s \not\models \mathit{pre}(a)$; *else* $\delta_{\mathcal{M}}(s, a) \models s \cup \mathit{eff}^{+}(a) \setminus \mathit{eff}^{-}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. The cumulative transition function is $\delta_{\mathcal{M}}(s, \langle a_1, a_2, \ldots, a_n \rangle) = \delta_{\mathcal{M}}(\delta_{\mathcal{M}}(s, a_1), \langle a_2, \ldots, a_n \rangle)$.

A CPP is represented using the Planning Domain Definition Language or PDDL (McDermott *et al.*, 1998).

A Plan Generator Module (PGM)

(Helmert, 2006) computes a solution to a CPP $\mathcal{M}$ as sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. The cost of π is $C(\pi, \mathcal{M}) = \sum_{a \in \pi} c_a$ if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; ∞ otherwise. The cheapest plan $\pi^* = \arg\min_{\pi} C(\pi, \mathcal{M})$ is the optimal plan with cost $C^*_{\mathcal{M}}$.

A Plan Validation Module (PVM)

(Howey *et al.*, 2004) outputs, given plan π and planning problem $\mathcal{M}$, `True` iff $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; `False` otherwise.

A Plan Recognition Module (PRM)

(Ramírez and Geffner, 2010) outputs, given a *partial plan* $\hat{\pi}$ and a planning problem $\mathcal{M}$, a plan π that maximizes the probability that $\hat{\pi}$ is a sub-plan of π –

$$\pi \leftarrow \arg\min_{\pi} \mathcal{P}([\hat{\pi}]_{k=0}^{k \leq |\pi|})$$

Note that the above approach does not directly compute this. Instead, we use the compilation approach from (Ramírez and Geffner, 2009) to compute *the optimal plan that satisfies a set of observations given a goal* as the output of the PRM.

A Landmark Generation Module (LGM)

(Hoffmann *et al.*, 2004) outputs, given a planning problem $\mathcal{M}$, a set of state (or action) landmarks $\mathcal{L}$ containing states (or actions) that must be passed through (or executed) in any satisficing solution of $\mathcal{M}$. Thus –

- An action landmark $a \in A$ requires that $a \in \pi$
  $\forall \pi : \delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$.
- A state landmark $s \subseteq F$ is such that $\forall \pi : \delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$, $\exists [\hat{\pi}]_{k=0}^{k \leq |\pi|} : \delta_{\mathcal{M}}(\mathcal{I}, \hat{\pi}) \models s$. (Zhu and Givan, 2003)

A Human-Aware Planning Problem (HAP)

is given by the tuple $\Psi = \langle \mathcal{M}, \mathcal{M}^H \rangle$ where $\mathcal{M}^H = \langle D^H, \mathcal{I}^H, \mathcal{G}^H \rangle$ is the human's understanding of the planning problem $\mathcal{M}$ (Chakraborti *et al.*, 2017a, 2019b).

An Explicable Planning Module (EPM)

computes a plan π such that it is a satisficing solution to $\mathcal{M}$ and is as close as possible to the expected plan in the human's model (Zhang *et al.*, 2017; Kulkarni *et al.*, 2016) –

$$C(\pi, \mathcal{M}) \approx C^*_{\mathcal{M}^H}$$

A Plan Explanation Module (PEM)

outputs, given a HAP $\Psi = \langle \mathcal{M}, \mathcal{M}^H \rangle$ and the optimal solution π^* to $\mathcal{M}$, the *shortest* explanation (Chakraborti *et al.*, 2017b) in the form of a model update to the human mental model $\mathcal{M}^H$ so that the same plan is now also optimal in the human's updated mental model $\widehat{\mathcal{M}}^H$ of the problem –

$$C(\pi^*, \widehat{\mathcal{M}}^H) = C^*_{\widehat{\mathcal{M}}^H}$$

The PEM can, in fact, trade off (Chakraborti *et al.*, 2018a) the relative cost of explicability (i.e. deviation from optimality in the planner's model) to the cost (i.e. length) of explanations during the plan generation process itself by computing a plan π and an explanation or model update $\mathcal{E}$ such that π is a solution to $\mathcal{M}$ and is the optimal solution to $\widehat{\mathcal{M}}$ modulated by a hyperparameter α –

$$\pi \leftarrow \arg\min_{\pi} \ |\mathcal{E}| \ + \ \alpha \times \mid C(\pi, \mathcal{M}) - C^*_{\mathcal{M}} \mid$$

With higher α, PEM computes plans that require more explanation, while with lower α, it generates more explicable plans. We refer to this variant as PEM(α).

Internally, PEM performs what is referred to as a *model space search* to come up with these explanations. This is done using *unit edit functions* λ that progressively try out one or more updates to the model $\mathcal{M}^H$ from the set of possible updates in $\mathcal{M}\Delta\mathcal{M}^H$ until the optimality conditions as described above are satisfied. This is known as the process of *model reconciliation* (Chakraborti *et al.*, 2017b, 2018a).

3.3 ITS as Planning

We will now cast the design of a generic ITS in terms of the planning modules discussed in the previous section.

3.3.1 Class Configuration

A class configuration is defined as the tuple –

$$\mathcal{C} = \langle \{KC_i\}, \{T_i\}, \{A_i\}, \{S_i\} \rangle$$

- *Knowledge Components or Concepts:* $\{KC\}$ is a set of knowledge components or concepts KC_i. In ITS literature, the process of knowledge acquisition by a student has been decomposed into smaller components referred to as KCs (Koedinger *et al.*, 2010). KCs can be anything from a production rule (Mayer, 1981), to a facet, misconception, fact or even a skill (Bloom *et al.*, 1964). The aim of the social learning process is to make a student acquire different KCs based on their and their classmates already existing ones.

- *Tutorial:* The class also constitutes of a set $\{T_i\}$ of tutorials $T_i \subseteq \{KC_i\}$ that consist of a set of KCs on which they provide information on. These directly modify the student's knowledge state by providing information on specific topics or on how certain problems or (parts of) assignments may be solved. These form an integral part of a curriculum for the class.

- *Activities / Assignments:* The class also has a set $\{A_i\}$ of activities or assignments $A_i = \langle \mathcal{M}, \kappa \rangle$ where $\mathcal{M}$ is the model of the assignment and $\kappa \subseteq \{KC_i\}$ consists of a set of KCs that are required to solve it. These engage the student in actions that derive from knowledge introduced in the class (learning by doing). These form the core content of the class. Technically, these can also be used as sensing actions for the ITS in determining the knowledge state of the student. Thus, an assignment may be used both as a way of estimating the student model as well as a technique for imparting knowledge to the student.

- Finally, the class has a set $\{S_i\}$ of students S_i. The student knowledge state or *model* is defined as $S_i = \langle \{A_i^S\}, \kappa_1, \kappa_2 \rangle$ where A_i^S is the student's understanding (similar to the definition of a HAP) of the assignment model A_i and $\kappa_1, \kappa_2 \subseteq \{KC_i\}$ consists of a set of KCs that they have *learned* and *applied* respectively.

Given a class configuration $\mathcal{C}$, a curriculum is given by a sequence $c(\mathcal{C}) = \langle c_1, c_2, \ldots, c_n \rangle; c_i \in \{T_i\} \cup \{A_i\} \cup \{S_i\}$ of tutorials, assignments and partnerships with other students.

3.3.2 Tips and Hints

A solution to an assignment in a general sense can be seen as a sequence of steps, a.k.a. a *plan*. Thus, we posit that a large variety of assignments can in fact be modeled in terms of the planning problem. The model $A_i(\mathcal{M})$ of an assignment A_i (as mentioned before) is thus the model of a planning problem CPP. As explored in (Sengupta *et al.*, 2017) in the context of decision support using automated planners, this opens up the slew of planning techniques (described in Section 3.2) that can be readily adopted to provide targeted (problem specific but domain independent) feedback to the students.

Solution Validation

For a partial attempt (represented as a partial plan $\hat{\pi}$) on an assignment A_i, the Plan Validation Module (PVM) indicates conditions that were unsatisfied, which can be used to provide targeted feedback. For example, the PVM can be used by the instructor to auto-grade solutions proposed by a student, since this is a domain independent way of checking if the plan is a valid solution of the given assignment (represented as a CPP $A_i(\mathcal{M})$). This is also useful for the student as well who can receive immediate feedback on whether they are successful (and why, if not) without having to wait for the instructor. This is one of the features that most ITSs

already possess. However, they are usually system level implementations that do not generalize across assignments.

Solution Completion

For a partial attempt (represented as a partial plan $\hat{\pi}$) on an assignment A_i, the Plan Recognition Module (PRM) produces a completion that can be *sampled* from to provide hints that guide the student towards the full solution. The PRM thus allows the ITS to anticipate what actions the student needs to take given what they have already done in order to achieve their goal. Notice that the partial plan is generated by the student (from the model A_i^S) even though the completion is done using A_i. This can thus help the student in cases of cognitive overload, but not if they lack the knowledge to solve the problem, i.e. $A_i^S \neq A_i$. We will discuss ways to deal with the latter case in Sections 3.3.3.

The PRM module can be also used to provide *proactive support* by recognizing that the students is going astray and providing pop-ups to guide them towards the right solution. Proactive support and has been shown (Zhang *et al.*, 2015b; Sengupta *et al.*, 2017; Chakraborti *et al.*, 2017c) to be desirable of an artificial agent in collaborative settings. Interestingly, one could also imagine using the PRM to detect gaming of the tutoring system (Muldner *et al.*, 2010) by defining it as a possible goal that a student might be trying to achieve, and based on the observations identify whether a student is working diligently or trying to game the system.

Problem Summarization

Finally, the Landmark Generation Module (LGM) takes in the Classical Planning Problem (CPP) representation $A_i(\mathcal{M})$ for a specific assignment A_i and produces a set of steps (action landmarks) or situations (state landmarks) that the student *must*

go through in order to solve the assignment. This can be very useful in providing a concise summary of "TODOs" required of the student to arrive at the solution, or by considering the domain variables that the student has already set to true, measure the progress of a student and thereby help the instructor in classroom orchestration (Dillenbourg *et al.*, 2011).

We shall illustrate each of these use cases in Section 3.5.1.

3.3.3 On-demand Curriculum Generation

A typical feature of online learning, as we discussed in Section 3.1, is that students increasingly select a subset of class materials to follow and leave once they are done (e.g. MOOCs are known to have notoriously low completion rates (Amy Ahearn, 2017)). As a result, students end up following individual and different curricula asynchronously. From the students' perspective an obvious problem with this is that they might not have the required knowledge to complete the materials they want. In the following, we thus address the problem of *on-demand curriculum generation*. In this paradigm, the student selects a particular assignment A_i to complete and the ITS performs argumentation with the assignment model $A_i(\mathcal{M})$ and the students model of the assignment $A_i^S(\mathcal{M})$ to identify deficiencies in the student model that need to be addressed using relevant tutorials.

In order to achieve this, the ITS spawns an instance of the Plan Explanation Module (PEM) with the HAP $\Psi = \langle A_i(\mathcal{M}), A_i^S(\mathcal{M})\rangle$ – here the instructor model is the ground truth and the student model needs to be reconciled. The model edit functions λ are the tutorials in the class. The output of the PEM is thus the *optimal set of tutorials* (this forms the recommended curriculum) that guarantees that the same solution (plan) is optimal in both the student model as well as the instructor model (even though they are not equal). This is especially useful since the instructor model

is going to contain information pertaining to the entire class, while the student does not need to know all these details in order to solve a specific assignment. The PEM is thus able to leverage the student and instructor models of an assignment to provide the exact set of tutorials that the student requires. We will provide illustrations of this process in Section 3.5.2. Notice that, the ITS can either use its estimate of A_i^S or engage in active information gathering by asking the student questions to determine parts of the student model it is uncertain about (Sreedharan *et al.*, 2018), in order to meet the specific needs of the student.

Teaching as an α trade-off

Notice that the formulation of the assignments as planning problems allow us to spawn CCPs with the student models (indicating how the student can solve the problem) or the instructor model (indicating how the instructor will solve the same problem) or anywhere in between (as computed by PEM(α)). The student solution (equivalent to an explicable plan) is likely to be suboptimal, or in most cases, not feasible in the ground truth or instructor model. An instantiation of PEM(α) with the HAP $\Psi = \langle A_i(\mathcal{M}), A_i^S(\mathcal{M})\rangle$ thus allows us to modulate the level of expertise with which a student wants to solve an assignment. For low values of α, the ITS will recommend the smallest possible curriculum that will just enable the student to solve the assignment (albeit suboptimally) while for progressively higher values of α it will start recommending more and more advanced curriculum to the point it matches the output of PEM, i.e. the optimal complete curriculum. From the perspective of the instructor as well, the α hyperparameter can be gradually increase from a low value to generate study materials for individual students as the course progresses. Thus the teaching process itself can be viewed through the lens of the model reconciliation

process as one of modulation of the value of α in the PEM(α). We shall demonstrate this in Section 3.5.2.

Remark

To the best of our knowledge, algorithms for the on-demand curriculum generation process driven by a specific class activity, and the argumentation process over the curriculum with the desired expertise level of student, have not been explored before in the ITS literature. This technique can be useful from the perspective of both the instructor and the student – e.g. the former can stagger the course content to meet the student's expertise level, while the latter can chose to learn at different levels of expertise (thus possibly reducing the high dropout rates that plague the on-demand learning communities).

Composition of Student Models

Finally, we note that we can extend the model edit functions in the PEM from just the tutorials in the class to the other student models as well. Thus the model updates during the model reconciliation process can be affected by either the KCs provided by tutorial or a composition of one or more student models. The output of PEM will now provide an optimal recommendation of tutorials and potential study partners based on the skill sets (i.e. models) of the individual students.

3.3.4 The Jigsaw Problem

The Jigsaw Problem is the process of creating smaller groups in a class for co-operative learning (Aronson, 1997). It has shown to have positive effect on students learning the course material together, and then engaging in discussions. This leads to a more active and deeper learning in class (Aronson, 2011). Aronson, points out

ten fixed steps to achieve this where the groups are created based on the ethnicity, race, gender and ability. However, it is intractable for a teacher to reason about all the student models and create study groups. Casting the class-level curriculum generation problem as a planning problem allows us to generate curricula for the entire class while enabling the instructor to specify desired properties of the curricula that needs to be maintained. These properties may be –

- Maximum size of study groups;
- Specific assignments of students;
- No repetition (or conversely, continuation) of study partners; and so on ...
- In this paper, we specifically focus on the following property – *every student not only learns but applies all concepts in the class at least once.* This is especially important in the social learning paradigm, to ensure that students have mastered all concepts and not depended on other students to finish a shared curriculum.

In order to achieve this, we define a planning problem with the start state compiled from the class configuration $\mathcal{C}$ and a goal state that model a class configuration where $\forall S_i : S_i(\kappa_2) = \{KC_i\}$ – i.e. every student has applied all the concepts in class. The operators are generated from the set of tutorials and assignments – the tutorial operator has its associated KCs as effects of being learned; while assignment operators has KCs as preconditions (that need to be learned) and effects (of those KCs having been applied).

This formulation [3] thus not only ensures that all the students have mastered all the concepts in the class materials but also that the length of the curriculum is reduced (from $|\{S_i\}|$ times the length of the curriculum for individual students) due to the collaborations across students who can bring in complementary skill sets and transfer knowledge. We provide an illustration of this in Section 3.5.3.

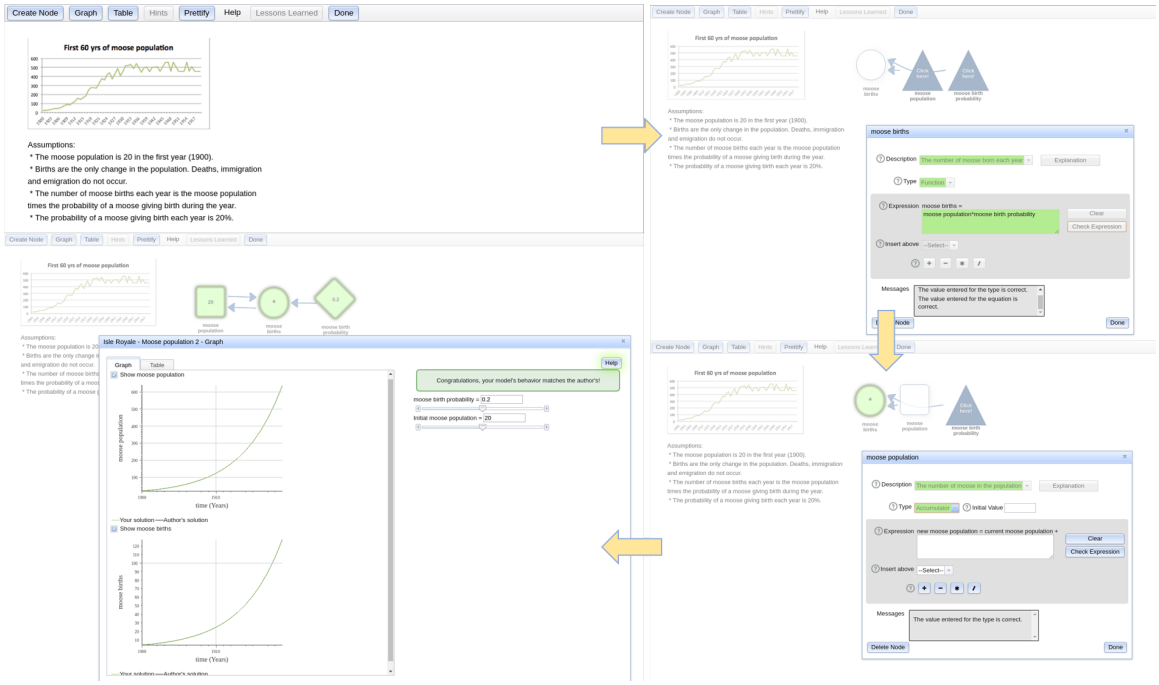


Figure 3.1: Illustration of the different stages of a "plan" being executed by a student in Dragoon – (1) the empty interface at the start of the problem (initial state); (2) the first node being completed; (2) the second node being created; and finally (3) the problem being completed with the feedback on the graph.

3.4 Introducing Dragoon

We will illustrate the above capabilities in Dragoon an ITS developed at Arizona State University to teach *dynamic system modeling* (VanLehn, 2013) in the physical classroom setting – over the course of almost half a decade of deployment, the system

[3]Note that this problem may be solved by horizon-limited planning, which is known to be NP-complete, the horizon being equal to $|\{S_i\}|$ times the length of the curriculum for individual students, which is the worst case curriculum length when no groups could be found. Thus, the jigsaw problem does not need the full expressiveness of CPP which is known to be PSPACE-complete.

has served 13 courses with approximate class sizes of 30, with more than a 1000 sessions per class. It is an ideal testbed for studying the nuances of tutoring systems currently deployed in classes in the space of mathematics, algebra and any other generic step-based tutoring systems. Figure 3.1 provides a snapshot of the interface.

In dynamic system modeling, a *system* is a part of the environment and *dynamic system* is the part of the environment that changes with time. Usually, first (or higher) order differential equations (differentiated with respect to time) represent dynamic systems mathematically. For simplicity of solving differential equations, time is discretized to calculate the values of different quantities. A *Model* refers to a representation of the system in a formal language.

Dragoon's formal language is based on Stella's stock and flow network (Doerr, 1996). It consists of three different types of quantities – (1) **accumulator** (quantity that changes); (2) **function** (quantity that may or may not change); and (3) **parameter** (quantity that remains constant). These quantities are called *nodes*. To create a node a student needs to define its *properties* – i.e. description, type, value, units and equation. They are connected to each other by equations called *relations*. Students are taught template structure for interaction between nodes, which show particular rate of change in values called *schemas* – e.g. linear schema represents linear change in values while exponential schemas represent exponential changes. Students practice on Dragoon through tutorial and assignment workbooks. A detailed description of Dragoon is available at (Wetzel *et al.*, 2017; VanLehn *et al.*, 2016, 2017).

3.4.1 The Isle Royale Workbook

We use the Isle Royale Workbook (https://goo.gl/ECrNnt) to illustrate the proposed techniques. It teaches students population dynamics of moose and wolf

population and learn interactions in a predator prey environment. There are six problems in the workbook (time step is a year) –

- **Isle-1** – Linear growth model of moose population, that is constant growth of two moose.
- **Isle-2** – Exponential growth model of moose population. The problem defines a constant growth rate which is multiplied by the population in the previous time-step to calculate the net growth.
- **Isle-3** – Exponential growth and death model of moose population. This problem adds the a constant death rate and the change in moose population is defined as the difference number of moose born and died.
- **Isle-4** – Exponential growth and death model of moose population with a fixed carrying capacity of the environment which effects the moose death rate.
- **Isle-5** – Exponential growth and death model of Wolf Population. This model is similar to Isle 3 problem.
- **Isle-6** – Exponential growth and death model of moose and wolf population with constant effect of wolf (predator) population on death rate of moose (prey) and constant effect of moose population on birth rate of wolf.

Epidemic schema is sometimes confused with exponential schema. Thus, we use one extra problem modeling flu epidemic in college which spreads through meetings between students. The number of students in the meeting and the chance that a student is affected is assumed to be constant.

The Zener Diode Problem

Most problems in Dragoon are solved with a single or unique *set* of steps. The only thing that changes is the sequence in which nodes are created. However, there are a few problems which can be solved in multiple ways, where a student can change the equations in the nodes to solve the problem in lesser number of nodes. One such problem is to model a Zener diode using Dragoon – if a student has a more advanced understanding of circuit theory, then they can easily solve the problem in fewer steps (i.e. using fewer nodes). We will thus use this problem to demonstrate the usefulness of PEM(α).

3.5 ITS as Planning in Action

We will now illustrate how the techniques introduced in Section 3.3 manifests themselves on Dragoon. The first step is to construct the instructor model $\mathcal{M}^I$ – examples can be accessed at – https://goo.gl/cyVthK.

We used nested object types to represent different objects in Dragoon, i.e. node, schema (KCs) and properties. Accumulator, parameter and function were of type node. Linear, exponential, extended_exponential, carrying_capacity and epidemic were types of schema. Description, value, type, equation and units are type of properties. These object types were used to define the state variables which characterize the properties that were part of a node, nodes that were part of schema, and schemas that were part of the problem. The operators in the domain represent the actions that are available student in the Dragoon environment. For example, a student fills each property to complete a node and it can be done in a fixed order. So the operator definitions were also related to initializing a node, filling every property of the node, completing a node and completing a schema. Students need an understanding of the

```
(create_node moose_birth_probability)
(create_node moose_births)
(create_node moose_population)
(fill_description da moose_population)
(fill_type_single_schema ta da moose_population s1 exponential_growth)
(fill_equation_single_schema ea ta moose_population s1 exponential_growth)
(fill_units ua ta moose_population)
(fill_value va ta moose_population)
(complete_accumulator da ta va ua ea moose_population)
(fill_description df moose_births)
(fill_type_single_schema tf df moose_births s1 exponential_growth)
(fill_equation_single_schema ef tf moose_births s1 exponential_growth)
(fill_units uf tf moose_births)
(complete_function df tf uf ef moose_births)
(fill_description dp moose_birth_probability)
(fill_type_single_schema tp dp moose_birth_probability s1 exponential_growth)
(fill_units up tp moose_birth_probability)
(fill_value vp tp moose_birth_probability)
(complete_parameter dp tp vp up moose_birth_probability)
(complete_exponential_schema moose_birth_probability moose_births moose_population exponential_growth s1)
; cost = 20 (unit cost)
```

```
(create_node moose_birth_probability)
(create_node moose_births)
(create_node moose_population)
(fill_description da moose_population)
(fill_type_single_schema ta da moose_population s1 exponential_growth)
(fill_units ua ta moose_population)
(fill_value va ta moose_population)
(complete_accumulator da ta va ua ea moose_population)
(fill_description df moose_births)
(fill_type_single_schema tf df moose_births s1 exponential_growth)
(fill_equation_single_schema ef tf moose_births s1 exponential_growth)
(fill_units uf tf moose_births)
(complete_function df tf uf ef moose_births)
(fill_description dp moose_birth_probability)
(fill_type_single_schema tp dp moose_birth_probability s1 exponential_growth)
(fill_units up tp moose_birth_probability)
(complete_parameter dp tp vp up moose_birth_probability)
(complete_exponential_schema moose_birth_probability moose_births moose_population exponential_growth s1)
; cost = 18 (unit cost)
Plan failed to execute

Plan Repair Advice:

(complete_accumulator da ta va ua ea moose_population) has an unsatisfied precondition at time 8
(Set (is_filled ea moose_population) to true)

Successful plans:
Value: 20
 ../fast-downward/isle2_complete_plan 20


Failed plans:
 ../fast-downward/isle2_incomplete_plan
```

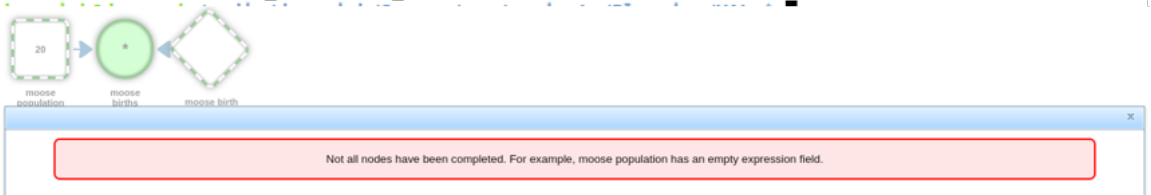


Figure 3.2: Response of PVM to the correct and incorrect or incomplete attempts in the **Isle-3** problem.

schema to fill the type and equation of the node. Thus actions for those steps have a precondition of has_schema to create the node. Finally, the initial state consists of all the nodes and schemas that are part of the assignment as well as the knowledge state of the student, that is whether they understand the schemas required to solve the problem. The goal state required that the student complete all the schemas that are present in a given problem.

3.5.1 Tips and Hints (c.f. Section 3.3.2)

Plan Validation

Figure 3.2, shows the 20-step solution for **Isle-2**, and Figure 3.1 shows some of these actions in the `Dragoon` environment. Figure 3.2 presents the incomplete attempt of the student being flagged as unsuccessful by the PVM, and shows the error generated after executing the incomplete plan in the `Dragoon` interface.

Plan Recognition

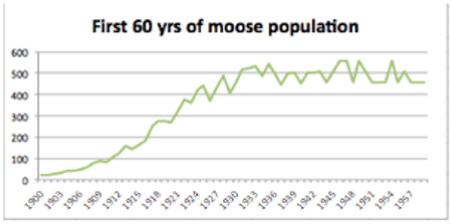
Figure 3.3 shows the correct identification by the PRM among two possible solutions of the **Isle-3** assignment using the "exponential_growth" schema or the "exponential_decay" schema from partial observations of the actions of the student in `Dragoon`.

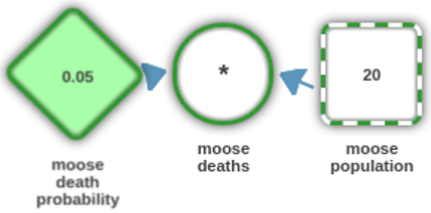
Landmarks

Figure 3.4 shows the 35 state landmarks produced by the LGM for the **Isle-3** assignment.

```
1  Experiment=../test-pr2/test-dragoon/test-dragoon.tar.bz2
2  Num_Hyp=2
3  Hyp_Atoms=(applied_exponential_schema moose_birth_probability moose_births moose_population)
4  Hyp_Test_Failed=True
5  Hyp_Cost_O=10000000.000000
6  Hyp_Cost_Not_O=10000000.000000
7  Hyp_Prob_O=0.500000
8  Hyp_Prob_Not_O=0.500000
9  Hyp_Plan_Time_O=0.030000
10 Hyp_Plan_Time_Not_O=0.040000
11 Hyp_Trans_Time=0.050000
12 Hyp_Plan_Time=0.070000
13 Hyp_Test_Time=0.120000
14 Hyp_Is_True=False
15 Hyp_Atoms=(applied_exponential_schema moose_death_probability moose_deaths moose_population)
16 Hyp_Test_Failed=True
17 Hyp_Cost_O=10000000.000000
18 Hyp_Cost_Not_O=10000000.000000
19 Hyp_Prob_O=0.500000
20 Hyp_Prob_Not_O=0.500000
21 Hyp_Plan_Time_O=0.020000
22 Hyp_Plan_Time_Not_O=0.040000
23 Hyp_Trans_Time=0.030000
24 Hyp_Plan_Time=0.060000
25 Hyp_Test_Time=0.090000
26 Hyp_Is_True=True
```

Create Node | Graph | Table | Hints | Prettify | Help | Lessons Learned | Done





Assumptions:
* The moose population is 20 in the first year (1900).
* Births and deaths are the only change in the population.
Immigration and emigration do not occur.
* The probability of a moose giving birth each year is 20%.
* The probability of a moose dying each year is 5%.

Figure 3.3: The output of the PRM in the **Isle-3** problem which can be solved in two separate ways. Here the student seemed to have decided to work on the exponential_decay schema.

3.5.2 On-demand Curriculum Generation (c.f. Section 3.3.3)

We use the same domain that we used in tips and hints. We are testing the case where a student wants to solve the **Isle-4** problem. Figure 3.5 shows the output of PEM when a student expresses a desire to complete the **Isle-4** assignment and requests a curriculum for it. The explanation presents the model differences in the

```
34 Atom applied_exponential_schema(moose_birth_probability, moose_births, moose_population)
33 Atom applied_exponential_schema(moose_death_probability, moose_deaths, moose_population)
8 Atom applied_schema(exponential_decay)
17 Atom applied_schema(exponential_growth)
1 Atom is_complete(moose_birth_probability)
18 Atom is_filled(up1, moose_birth_probability)
4 Atom is_filled(vp1, moose_birth_probability)
19 Atom is_filled(tp1, moose_birth_probability)
21 Atom is_filled(dp1, moose_birth_probability)
20 Atom init(moose_birth_probability)
6 Atom is_complete(moose_births)
5 Atom is_filled(ef1, moose_births)
3 Atom is_filled(uf1, moose_births)
7 Atom is_filled(tf1, moose_births)
24 Atom is_filled(df1, moose_births)
0 Atom init(moose_births)
31 Atom is_complete(moose_death_probability)
29 Atom is_filled(up2, moose_death_probability)
28 Atom is_filled(vp2, moose_death_probability)
13 Atom is_filled(tp2, moose_death_probability)
10 Atom is_filled(dp2, moose_death_probability)
9 Atom init(moose_death_probability)
30 Atom is_complete(moose_deaths)
26 Atom is_filled(ef2, moose_deaths)
11 Atom is_filled(uf2, moose_deaths)
32 Atom is_filled(tf2, moose_deaths)
27 Atom is_filled(df2, moose_deaths)
12 Atom init(moose_deaths)
14 Atom is_complete(moose_population)
2 Atom is_filled(ea, moose_population)
23 Atom is_filled(ua, moose_population)
22 Atom is_filled(va, moose_population)
25 Atom is_filled(ta, moose_population)
16 Atom is_filled(da, moose_population)
15 Atom init(moose_population)
```

Figure 3.4: The 35 state landmarks generated by the LGM for the **Isle-3** problem.

```
Explanation >> has-initial-state-has_schema s1 carrying_capacity
Explanation >> has-initial-state-has_schema s1 exponential_decay
Explanation >> has-initial-state-has_schema s1 exponential_growth
Explantion Size: 3
Total Time  24.517663002
```

Figure 3.5: On-demand curriculum generated by the PEM. This is the smallest change to the student model required to solve the **Isle-4** problem.

```
current explanation 2
['create_node current_thru_r'
 'create_node v_across_load'
 'create_node v_across_r1'
 'fill_description df4 v_across_r1'
 'fill_type_double_schema tf4 df4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_equation_double_schema ef4 tf4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_units uf4 tf4 v_across_r1'
 'complete_function df4 tf4 uf4 ef4 v_across_r1'
 'complete_kvl_schema_temp v_across_r1 kvl s1'
 'fill_description df5 v_across_load'
 'fill_type_single_schema tf5 df5 v_across_load s1 zener_voltage_regulator'
 'fill_equation_single_schema ef5 tf5 v_across_load s1 zener_voltage_regulator'
 'fill_units uf5 tf5 v_across_load'
 'complete_function df5 tf5 uf5 ef5 v_across_load'
 'fill_description df6 current_thru_r'
 'fill_type_single_schema tf6 df6 current_thru_r s1 zener_voltage_regulator'
 'fill_equation_single_schema ef6 tf6 current_thru_r s1 zener_voltage_regulator'
 'fill_units uf6 tf6 current_thru_r'
 'complete_function df6 tf6 uf6 ef6 current_thru_r'
 'complete_zener_voltage_regulator_schema v_across_r1 v_across_load current_thru_r zener_voltage_regulator s1']

current explanation 3
['create_node current_thru_r'
 'create_node v_across_load'
 'create_node v_across_r1'
 'fill_description df4 v_across_r1'
 'fill_type_double_schema tf4 df4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_equation_double_schema ef4 tf4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_units uf4 tf4 v_across_r1'
 'complete_function df4 tf4 uf4 ef4 v_across_r1'
 'complete_kvl_schema_temp v_across_r1 kvl s1'
 'fill_description df5 v_across_load'
 'fill_type_single_schema tf5 df5 v_across_load s1 zener_voltage_regulator'
 'fill_description df6 current_thru_r'
 'fill_type_single_schema tf6 df6 current_thru_r s1 zener_voltage_regulator'
 'fill_equation_single_schema_expert ef6 tf6 s1 zener_voltage_regulator'
 'fill_units uf6 tf6 current_thru_r'
 'complete_function df6 tf6 uf6 ef6 current_thru_r'
 'complete_zener_voltage_regulator_schema v_across_r1 v_across_load current_thru_r zener_voltage_regulator s1']
```

Figure 3.6: Different plans and associated model updates generated by the PEM(α) based on the α-hyperparameter. For a high value of α the curriculum is of size 3 after which the problem can be solved in 17 steps. With a lower value of α, the problem can be solved with a longer 20 step plan.

initial state that prevents the student from completing the assignment at this time and suggests tutorials to introduce these concepts. The explanation is of size 3, and references the missing knowledge concepts that are needed for solving the problem in the 40 steps.

Figure 3.6 shows how PEM(α) can be used to modulate the expertise levels of the recommended curriculum. The complete curriculum is of size 3 after which the problem can be solved in 17 steps. But, with lower value of α, the problem can be solved with a longer 20 step plan. As explained earlier, even though the student needs two knowledge concepts to solve the problem (zener_voltage_regulator and kvl schema), but to solve the optimal plan a student needs to be an expert and improve the equations in one of the nodes and create a better model for Zener Diode problem.

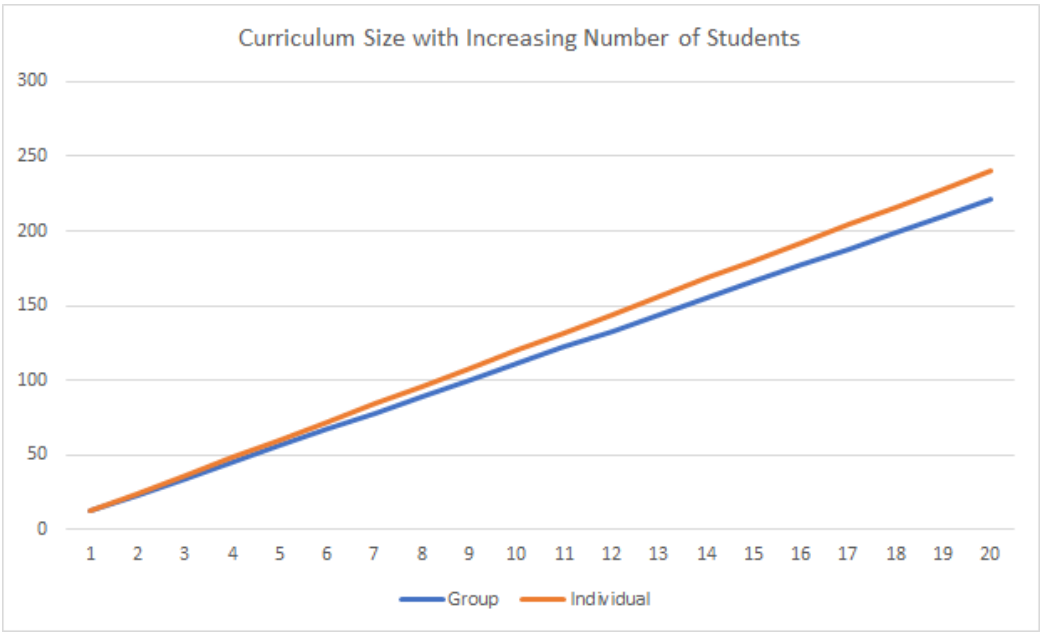


Figure 3.7: Group versus individual curriculum lengths with increasing class size.

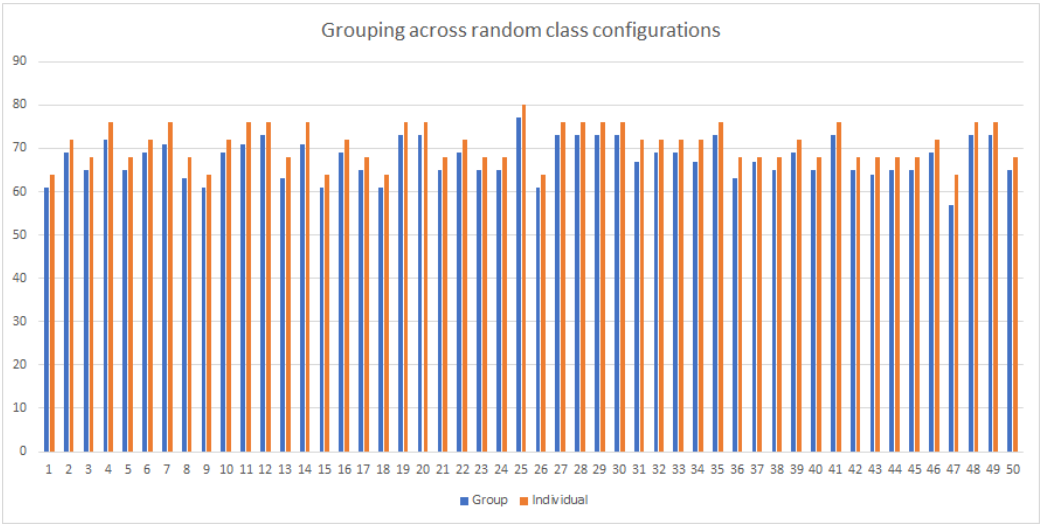


Figure 3.8: Group versus individual curriculum lengths in different class configurations.

3.5.3 Jigsaw Problem (c.f. Section 3.3.4)

Here, we took an instance of a Dragoon class with 7 concepts and 9 assignments. A single student curriculum comes out as 12 steps long, with 7 tutorials and 5 assignments. However, with the introduction of groups of two students, this reduces to a combined curriculum of 23 steps where every student applies every concept at least once. For every new student, plan size increases by 11 steps, showing that one of the assignment can be done in the group. This is shown in Figure 3.7, which plots the curriculum length with increasing class sizes. In this particular class configuration, only one of the assignments could be done in a group.

Now we study the effect of varying class configurations by the making assignments that randomly teach up to 4 concepts. The number of concepts were fixed to 10 and there were 20 assignments that would teach these concepts. Figure 3.8 shows the curriculum length for 50 different randomly generated four student class configurations. We observe a decrease of 3 to 7 steps in every class.

3.5.4 Conclusion

In this chapter, we demonstrated how an ITS framework can be built using the state-of-the-art in human-aware planning techniques for the design of course independent support features. The last section illustrated these properties in a real tutoring system Dragoon. We used Dragoonãs has been used in several classes at ASU (in Sustainability department) to teach students modeling dynamic systems as represented by the Isle Royale Workbook. Dragoonh̃as been shown to be effective for teaching students, thus, in the next chapter we describe our efforts towards evaluating the effectiveness of automated planning and HAAI techniques.

Chapter 4

IPASS – ACTIVE DECISION SUPPORT FOR STUDENTS

In the previous chapter we discussed how planning and HAAI techniques can be user in a classroom. but to be certain about the usefulness we need to perform a user study. The dragoon system has already been tested with several user studies (VanLehn *et al.*, 2016, 2017; Grover, 2015; Grover *et al.*, 2018a) and many courses at ASU. However, it is essential to directly evaluate the effectiveness of these techniques, thus we created another system called `iPass` to help university students make their *plan of study* (iPOS), as their are plenty of domain experts in the university. We begin with a brief description of the challenges faced while applying these techniques to provide support to exprets, followed by describing the domain for iPOS and the `iPass` interface and its decision support components.

4.1 Introduction

Human-in-the-loop planning (HILP) (Kambhampati and Talamadupula, 2015) is a requirement in many present-day complex decision making and planning environments. In this paper, we consider a case of HILP where the human responsible for making the decisions in a complex scenario is supported by an automated planning system. High-level information fusion that characterizes complex long-term situations and supports the planning of effective responses is considered the greatest need in crisis-response situations (Laskey *et al.*, 2016). Indeed, automated planning based proactive support was preferred by humans involved in teaming with robots (Zhang *et al.*, 2015a) where the cognitive load of the involved subjects involved was observed to have been reduced (Narayanan *et al.*, 2015).

Traditional planning techniques have focused on end-to-end plan generation rather than proactive support. Although there has been some work recently to make these techniques *human-aware* that try to account for human activities and intents while constructing the plans (Zhang *et al.*, 2017), such as generating explicable (Zhang *et al.*, 2017) or legible plans (Dragan *et al.*, 2013), these works still focus on complete plan generation. Thus, none of these techniques can be directly adapted to providing decision support. In this work, we investigate the extent to which an automated planner can support the human's decision-making process, despite not having access to the complete domain and preference models, while the humans remain in charge of the process. This is appropriate in many cases, where the human-in-the-loop is ultimately held responsible for the plan execution and it's results. This is in contrast to earlier work on systems such as `TRAINS` (Allen, 1994), `MAPGEN` (Ai-Chang *et al.*, 2004) and (Kim *et al.*, 2017) where the planner is in the drivers seat, with the humans "advising" the planner. Thus, our work is distinct from them on *mixed-initiative* planning where humans enter the land of automated planners, manipulating their internal search process – here, the planners enter the land of humans.

An important complication arises because the planner and the human can have different (even complementary) models of the same domain or knowledge of the problem at hand (as shown in Figure 4.1). In particular, humans might have additional knowledge about the domain or the plan preferences that the automated planner is not privy to. This means that plan suggestions made by the automated planner may not always *make sense* to the human in the loop, i.e. appear as sub-optimal in their model. This is an ideal opportunity for the system to provide model updates as explanations (Chakraborti *et al.*, 2017b) and reconcile the models through iterative feedback from the human during the plan generation phase.

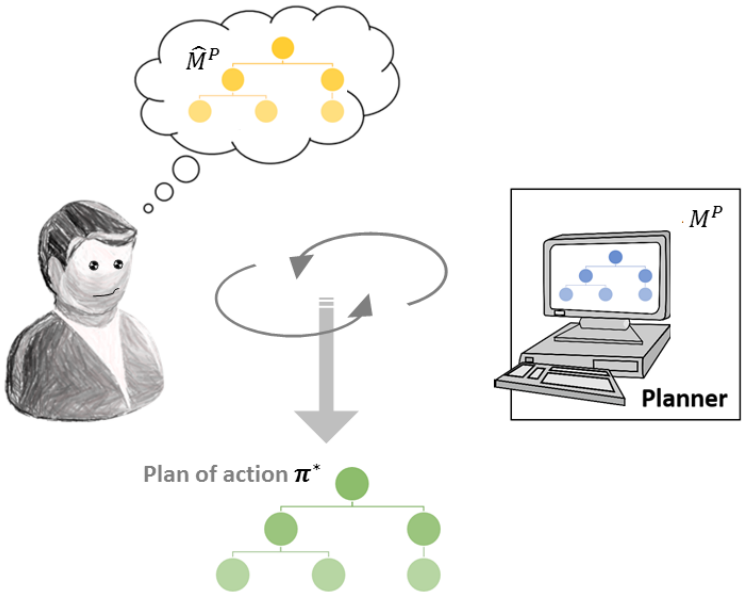


Figure 4.1: Planning for decision support must consider difference in models between the planner and the human.

The extent to which a planner can be used for decision support is largely dependent on the nature of the model that is available. For example, if we have an incomplete model that often occurs in many mixed-initiative settings (Smith, 2012), then an automated support component can use the incomplete model to complete or critique existing plans (Manikonda *et al.*, 2017). Keeping this in mind, in the current paper we focus on scenarios which come with more well-defined protocols or domain models, and illustrate how off-the-shelf planning techniques may be leveraged to provide various degrees of decision support (as opposed to complete automation). We believe such technologies will be helpful in naturalistic decision making scenarios such as disaster response where the cognitive overload of the human can negatively affect the quality of decision making.

Human-Computer Interaction (HCI) is thought to have developed as a sub-field in three different areas – management information systems, computer science, and human factors (Grudin, 2011). While human factors have evolved to understand the

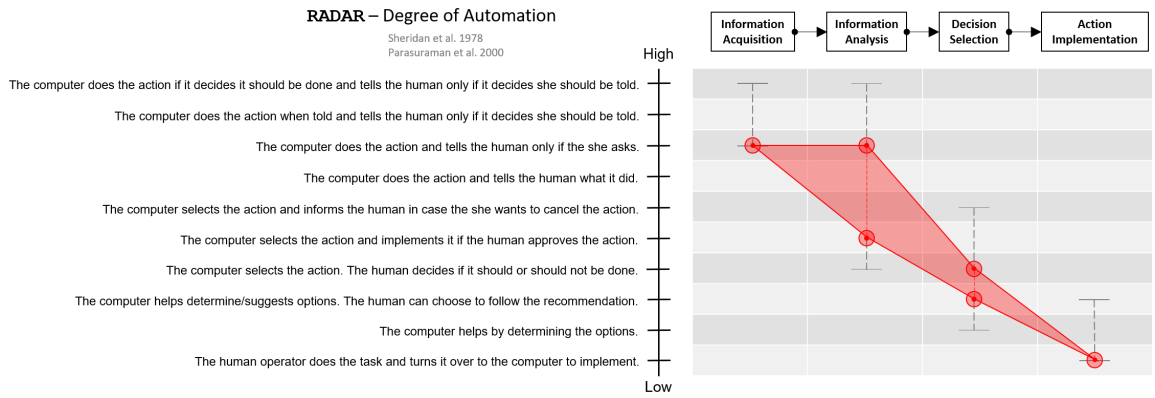


Figure 4.2: Degrees of automation of the various stages of decision support, and the role of `iPass` in it.

behavioral effects of agents on different interfaces, management information systems, and computer science worked on various ways of designing these interactions. In the past, two of the most common methods of interactions were – direct manipulation and interface agents (Shneiderman and Maes, 1997). While direct manipulation occurs when the interface changes only based on the user's instructions, interface agents are assumed to possess more intelligence and adapt by themselves, behaving like a collaborator (Maes *et al.*, 1997) (perceived as intelligent software agents). For example, software for classifying news or email as relevant or not, in the context of a specific user, can be thought of as an intelligent software agent. In this paper, we look at a specific intelligent software agent that provides decision support to a user and reduces their cognitive and information overload for sequential decision-making tasks.

Earlier works have applied the principles of Human-Human Interaction (HHI) for designing a collaborative disclosure interface (Lesh, 2004) rather than motivating the design of decision support software with principles in Human-Computer Interaction (HCI) directly. This work, to our knowledge, is the first to propose proactive decision support (PDS) system `iPass` following some of the design principles laid out in the literature in the (HCI) community. *Proactive decision support* can be described as the

act of providing decision support to the user without waiting for an explicit request and proactively checking for decision failures due to various reasons like resource management. We demonstrate possible roles that existing automated planning technologies can play in the deliberative process of the human decision-maker in terms of the degree of automation of the planning process.

In the past, there have been parallels drawn between the work in HCI and AI where they were described as two fields divided by a common focus (Grudin, 2009). Since the idea of intelligent software agents (Maes, 1995), the HCI community has used AI techniques for many applications, where *adaptive interfaces* connects directly to the notion of *adaptive agents* in the automated planning community. Furthermore, we believe that the notions of predictability of an adaptive interface (Gajos *et al.*, 2008) and explanations in the context of complex strategies for such interfaces (Rader *et al.*, 2018) have connections to the identifiability and predictability of plans (Chakraborti *et al.*, 2019a) and explanations in automated planning literature (Chakraborti *et al.*, 2017b). Although works on the HCI side have shown how such interfaces affect user's behavior (Langley, 1999), their mental workload (Hancock and Chignell, 1988) or user satisfaction (Rader *et al.*, 2018), works in automated planning, in the context of decision support, lack similar human studies. In this work, we seek to address this concern.

Contributions

The purpose of this paper is to showcase how the various planning technologies can be used to support human decision-makers. Thus, in this work, we –

- Show that state-of-the-art planning techniques can be adapted to design a Proactive Decision Support system, `iPass`.

- Describe how the design decisions for `iPass` are driven by the literature in the HCI community.
- Present user studies using `iPass`, a decision support system (similar to `iPass`) designed for university students, to show the effectiveness of automated planning techniques for decision support.

Every domain comes with its nuances and thus, needs a personalized design for the decision support system to better support the use-cases desired by domain experts. `iPass` showcases how a Fire Marshal may need decision support to handle emergencies in time-critical real-world scenarios. Similarly, `iPass` is used by graduate students to create their plan of study. Thus, we designed and implemented two different systems– one to show the applicability of human aware automated planning in real-world decision-making scenarios and the other to highlight the effectiveness of such systems with users. Note that beyond the two domains highlighted in this paper, our methods for decision support can be leveraged, with minor changes to interfaces, across various domains.

4.2 Design Principles

Before explaining how one can use planning technologies for decision support, we highlight the important features of the deliberative process that we have to ensure for seamless interaction with the planner.

Naturalistic Decision Making

The proposed proactive decision support system supports *naturalistic decision making* (NDM), which is a model that aims at formulating how humans make decisions in complex time-critical scenarios (Klein, 2008). It is acknowledged as a necessary

element in PDS systems (Morrison *et al.*, 2013). Systems which do not support NDM have been found to have detrimental impact on work flow causing frustration to decision makers (Feigh *et al.*, 2007). At the heart of this concept lies, as we discussed before, the requirement of letting the human to be in control. This motivates us to build a proactive decision support system, which focuses on aiding and alerting the human in the loop with his/her decisions rather than generate a static plan that may not work in the dynamic worlds that the plan has to execute in. In cases when the human wants the planner to generate complete plans, they still have the authority to ask `iPass` to explain its plan when it finds it to be inexplicable (Chakraborti *et al.*, 2017b). We postulate that such a system must be augmentable, context sensitive, controllable and adaptive to the human's decisions. Various elements of human-automation interaction such as adaptive nature and context sensitivity are necessary for the ease of usability (Sheridan and Parasuraman, 2005). It has been shown, that vigilance requires hard mental work and is stressful via converging evidence from behavioral, neural and subjective measures (Warm *et al.*, 2008). Our system may be considered as a part of such vigilance support thereby reducing the stress for the human.

Degrees of Automation

One of the seminal works by Sheridan and Verplank builds a model that enumerates ten levels of automation in software systems depending on the autonomy of the automated component (Sheridan and Verplank, 1978). Later, in the study of mental workload and situational awareness of humans performing alongside automation software, Parasuraman separates the requirement for automation into four stages (Parasuraman, 2000)– Information Acquisition, Information Analysis, Decision Selection and Action Implementation (see Figure 4.2). We use this system as an objective

basis for deciding which functions for our system should be automated and to what extent so as to reduce human's mental overload while supporting Naturalistic Decision making. Parasuraman and Manzey show that human use of automation may result in automation bias leading to omission and commission errors (Parasuraman and Manzey, 2010), which underlines the importance of reliability on the automation (Parasuraman and Riley, 1997). Indeed, it is well known that climbing the automation ladder (shown in Fig. 4.2) might well improve operative performance but drastically reduce response quality when failures occur (Wickens *et al.*, 2010). Hence, to meet the requirement of naturalistic decision making, we observe a downward trend in automation levels (in Figure 4.2) as we progress from data acquisition and analysis (which machines are traditionally better at) to decision making and execution.

Interpretation & Steering

For the system to collaborate with the commanders effectively, in the context of a mixed-initiative setting, where the planner helps the human, it must have two broad capabilities - *Interpretation* and *Steering* (Manikonda *et al.*, 2017). Interpretation means understanding the actions done by the commanders (eg. sub-goal extraction, plan and goal recognition), while Steering involves helping the commanders to do their actions (eg. action suggestion, plan critiques). The current system mainly addresses the decision making aspect, which requires the ability to both interpret as well as steer effectively, even as it situates itself in the level of automation it can provide in the context of naturalistic decision making.

Human-AI interaction

Recent work by Amershi *et al.* suggested guidelines based on traditional user-interface design techniques to support designers while creating AI agents (or software) (Amershi

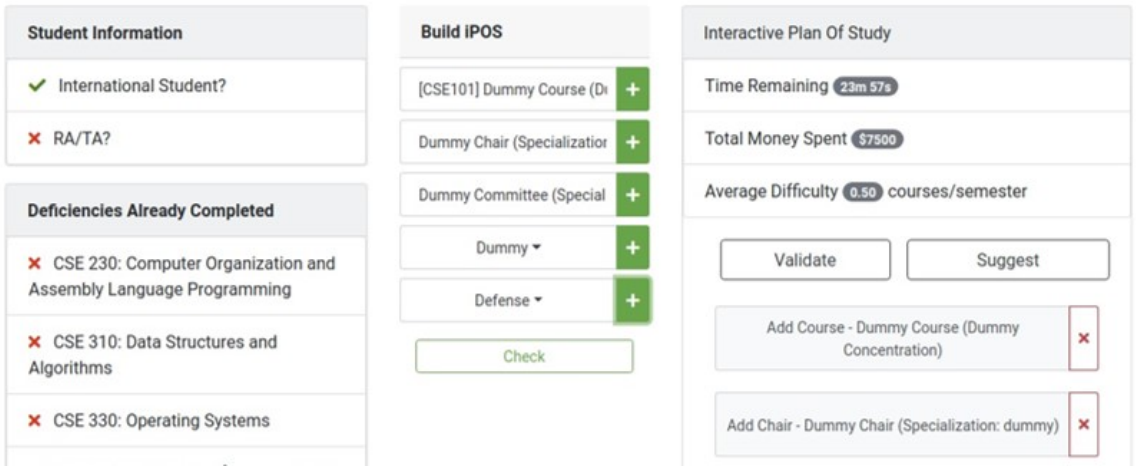


Figure 4.3: Illustration of the `iPass` interface.

et al., 2019) that can be mapped to earlier work on principles of mixed-initiative user-interfaces (Horvitz, 1999). The former work elicitates 18 guidelines under four categories based on the different phases of interaction– (1) initially, (2) during the interaction, (3) when wrong, and (4) over time. We highlight, how these guidelines were are already considered in our interface design.

4.2.1 The iPOS Domain and Interface

One of the major difficulties of designing user studies in the decision support paradigm is access to *domain experts* who can verify the real usefulness of the decision support for sequential decision making. Thus, earlier works that propose software to help the human in their decision making process (Sengupta *et al.*, 2017, 2018) are unable to provide any evidence as to how effective they are in practice. Keeping this in mind, we situate our study in a domain for constructing an *"interactive Plan of Study"* (iPOS) at Arizona State University. This has two implications. On one hand, this task is known to be challenging for any student as per (1) evidence in existing literature (Khan *et al.*, 2012), and (2) its use in the International Planning Competition (Ferland and Sanner, 2018) as a benchmark domain. On the other hand,

this is a domain that graduate students, who are easily accessible in the academic setting for large-scale user studies, are already familiar with because they have to build and maintain an iPOS for themselves as per university requirements. Note that, in `iPass` , we showcase the techniques that can be leveraged for decision support in real-world settings and evaluate the effectiveness of these techniques through `iPass`. This should give the reader an idea that beyond cosmetic changes needed for the user-interface, the decision support methods can be used out-of-the-box for other domains. Important rules that a student needs to remember while constructing an iPOS are:

- ◇ Complete 30 credits and where every course is 3 credits
- ◇ There are three deficiency courses (these courses are to be taken that are pre-requisites from under graduate classes and a student who has not taken them) that are to be finished before any normal course and they do not count towards 30 credits.
- ◇ Define area of specialization.
- ◇ Complete 3 specialization courses.
- ◇ Chose a chair and two other committee members.
- ◇ Chair should be from same area of specialization.
- ◇ Complete 2 research courses – CSE599A & 599B.
- ◇ Defense is to be scheduled in the last semester.

The interface (shown in Figure 4.3) has three panels – (1) The panel on the left shows the relevant information of the student (e.g. what deficiency courses they have, whether they are an international student, if they are research or teaching assistants,

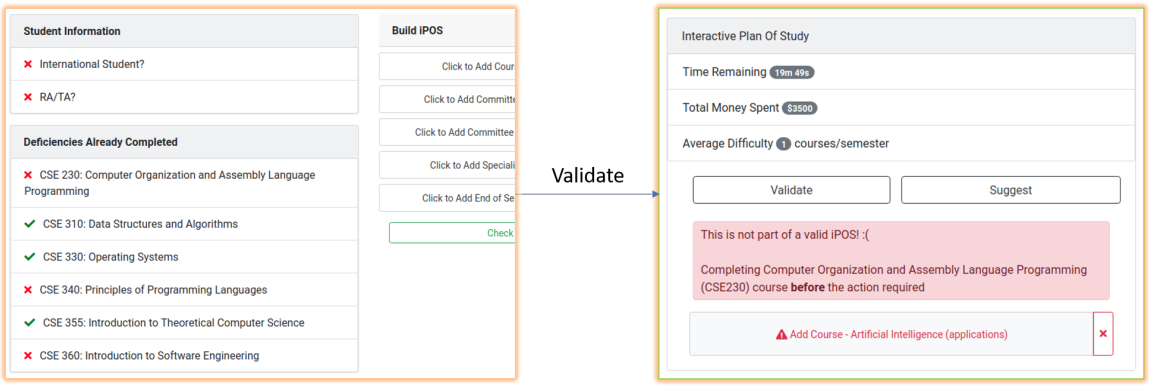


Figure 4.4: Illustration of Plan validation, where a student adds a course and checks whether the course can be taken at the beginning of the first semester. VAL provides feedback to the user, whether taking the action in a particular is possible or not.

etc.); (2) The central panel provides the student with options to build the iPOS for the given student information. Actions in this panel can include adding course, specialization, committee members, etc.; (3) The panel on the right provides an interactive interface to work on the plan (such as rearranging or deletion of action) along with relevant information about the plan (e.g. difficulty or average number of courses a semester, total cost of tuition for the current plan, etc.). This panel also houses the decision support components that, if available, lets the user ask for validation of the current plan or suggestions to complete it. The technical details for `iPass` were presented in the previous section, and now we will discuss specific modules that are provided to students for user study.

4.2.2 `iPass`– Decision Support Components

Although the technical details for all the decision support components have been provided in the previous sections (for `iPass`), there exists certain differences that we describe in this section. In `iPass`, we consider $\pi_e = \pi_h$ because the human-in-the-loop is, at the start of the study, given a randomly allocated initial state and asked to make an iPOS from scratch. We note that due to constraints on the response time,

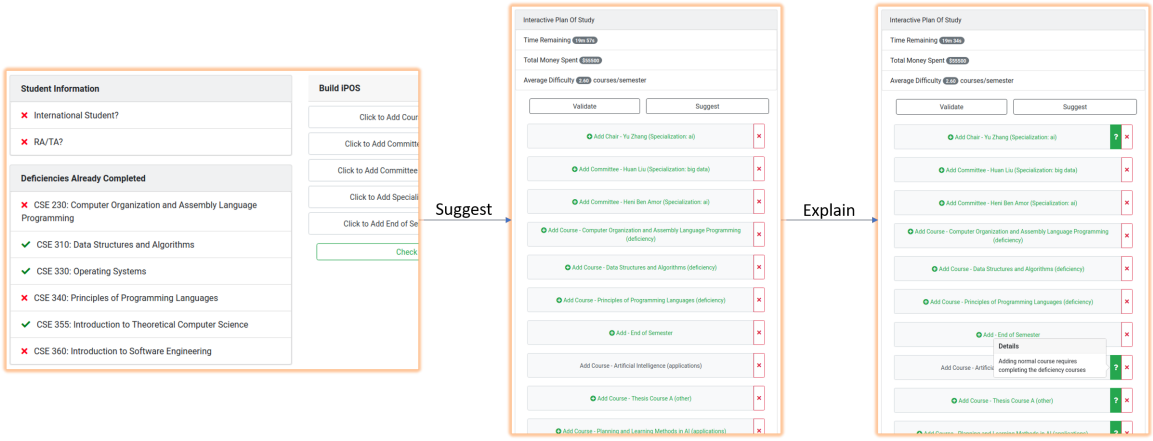


Figure 4.5: Illustration of Plan Suggestion and Explanation. Actions in green have been added by the planner and actions in black were added by the user. It follows from Validation scenario where the user first added the *Artificial Intelligence* course and then asks for suggestion of a complete plan with it. Explanation is shown using the box.

the decision support components of `iPass` differ from `iPass`, in some cases, to make them more scalable. Moreover, due to the availability of additional resources such as the user's manual, providing landmarks on the interface becomes redundant.

Plan Checking is shown as the check button in figure 4.3. Given a plan of study π_h generated by the student, it checks whether $\delta(\mathcal{I}, \pi_h) \models \top$ i.e., the student has a valid iPOSthat fulfils all the requirements necessary for graduation. The feedback generated is binary indicating whether the submitted iPOS is valid or not.

Plan Validation as mentioned before, is used to validate the student's plan π_h. For this purpose, we use `VAL` (Fox *et al.*, 2005), which validates whether all actions $\in \pi_h$ can be executed (i.e. all pre-conditions are satisfied). When an action cannot not be executed, a message is provided to the student explaining the why the iPOSis invalid. For example, figure 4.4 shows that a student, upon validation, is informed that they need to complete the course on Computer Organization (a deficiency) before enrolling in the course on Artificial Intelligence (which is a graduate-level course).

Stage	Support Component	iPass
Information Acquisition	Data Decision Loop	✓
Information Analysis	Plan Summarization	✗
	Model Updates	✓
	Plan Validation	✓
Decision Selection	Plan Correction	✓
	Action Suggestions	✓
	Optimal Plan Suggestions	✗
	Monitoring Plan Generation	✗
Action Implementation		✗

Table 4.1: Different decision support components present in `iPass` system where ✓ means that the support component is part of the system and ✗ means that the support component is not part of the current version of the system.

Action Suggestion The goal of action suggestion is to generate π_s given a partially constructed plan π_h by the student. In order to achieve this, we use the pr2plan compilation (Ramírez and Geffner, 2010) described above. In order to help the user distinguish between the suggested actions $\in \pi_s$ and the existing actions $\in \pi_h$, we highlight them on the interface. In the scenario shown in Figure 4.5, a student chooses their specialization (on Artificial Intelligence shown in black) and asks for suggestions that completes the iPOS. In this case, `iPass`adds the required amount of courses constrained based on the specialization, selects a graduate committee, and ensures that a dissertation is done. These may exist scenarios where the input partial plan π_h cannot be completed in any way to come up with a valid iPOS. In such cases, the used is notified that the added actions cannot lead to a valid iPOS. Note that

there might exist various suggestions for π_s. Some of them might be preferred by a particular student than others. Although we do not consider this setting, a explicable (Kulkarni *et al.*, 2016) plan completion algorithm might be useful.

Plan Explanations In order to provide meaningful explanations based on model reconciliation, we expect to have an idea of the student's understanding of an iPOS. Given that the graduate study domain made by us is significantly different from the university rules, we assumed an empty model of the student (i.e. they are not familiar with any of the constraints while constructing the iPOS). We then provide Plan Patch Explanations (PPE) that can explain the suggested plan. For example, as shown in Figure 4.5, the need for completing a particular course (Artificial Intelligence) is deemed to be necessary for specialization in the selected topic (AI). Note that while explanations provide details of the domain that support a plan, validation points out constraints that invalidate a plan. Thus, these functionalities are complimentary in the context of a sequential decision support system. As stated at the start of the section, due to the lesser complexity of the iPOS design task in comparison to the fire scenario and the need for quick response time, some of the decision support aspects were modified. We highlight the difference in the decision support components present in `iPass` and `iPass` in table 4.1.

4.3 Aim of the Study

In this section, we present the user study using `iPass` to evaluate the effectiveness of the decision support components present in the system. To determine the individual as well as the cumulative impact of the decision support components, we evaluated our interface in four conditions –

$C_{control}$ Both validation and suggestion capabilities are absent. The users do have to pass correctness before they can submit.

C_1 Only validation capability is enabled.

C_2 Only suggestion capability is enabled.

C_3 Both validation and suggestion options are available.

Furthermore, each participant is assigned to one of the study conditions C_i performed the iPOS planning task twice (with different, randomly generated initial states, i.e. student portfolio). We thus, have two sub-conditions (denoted using the super-script) C_i^1 and C_i^2 for each study condition. Given these four conditions, we hypothesize that–

H1. Planning performance P will be in increasing order of –

$$P(C_{control}) < P(C_1), P(C_2) < P(C_3)$$

Note that we do not expect validation or suggestion by themselves to be more useful than the other. "Performance" here can manifest itself in different forms –

H1a. The time to completion $T(C_i)$, $i = \{Control, 1, 2, 3\}$ will follow the same order, e.g. $T(C_{control}) > T(C_1), T(C_2) > T(C_3)$.

H1b. The satisfaction with the final plan of study constructed will follow the same order.

H1c. The satisfaction with the feedback from the interface will follow the same order.

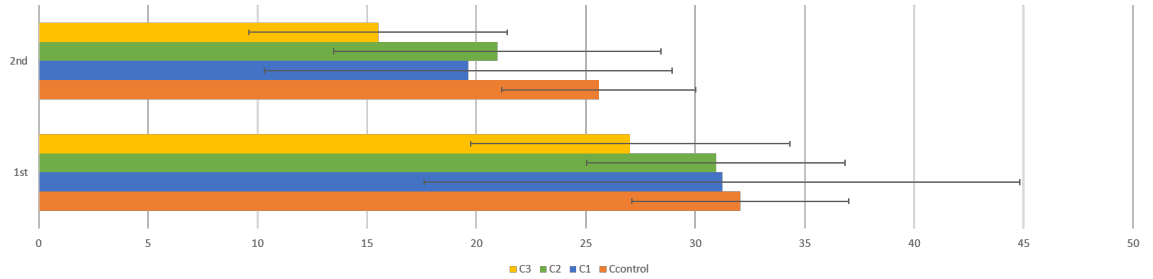


Figure 4.6: Average time taken (along with the standard deviation) by a participant to complete the two parts of the study for each condition C_i^1 and C_i^2.

H2. The time to completion will reduce in all four conditions, however the reduction $\Delta T(C_i) = T(C_i^1) - T(C_i^2)$ will also follow the same order, i.e. –

$$\Delta T(C_{control}) < \Delta T(C_1) < \Delta T(C_2) < \Delta T(C_3)$$

We expect this to happen because, in the later conditions, users are provided relevant details of the domain as they construct a plan, and are thus expected to become more familiar with the domain. We expect this effect to be more pronounced in C_2 and C_3 which provides explanations specifically for purposes of model reconciliation.

H3. The effects of support components on performance will be more pronounced for subjects with less expertise, e.g. students who had not previously completed their own iPOS.

4.4 Experimental Results

The study was conducted on the university premises. Each subject was given $15 for an hour of study where they used the `iPass` software to make two iPOSs. At the start of the study, participants were informed that they would be asked to explain each iPOS with the hope that it will help them be more invested in the task (Mercier and Sperber, 2011). Then they were given a document explaining the

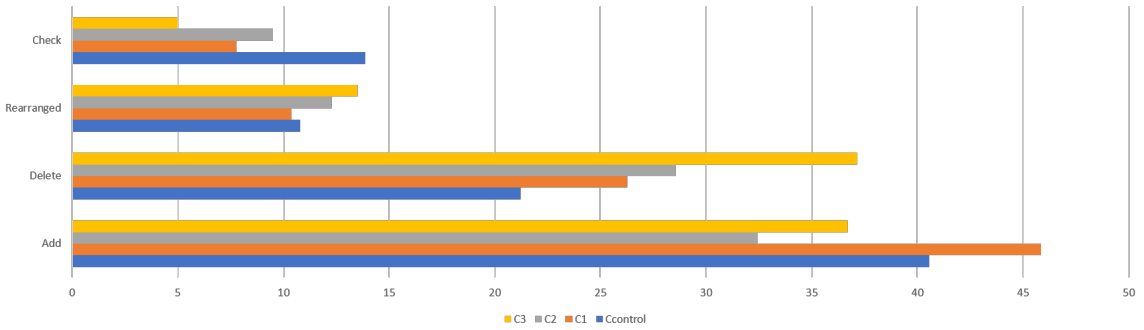


Figure 4.7: Average number of times participants added, deleted, rearranged courses or clicked 'check' while making an iPOS for all the conditions C_i^1.

planning domain and another document explaining the functionality of the elements in the interface. Lastly, they were given 20 minutes to make each iPOS in order to simulate the time-critical nature of the environment. At the end, they were presented with a feedback form.

After performing pilot studies with two participants, we sent out a department-wide advertisement asking interested participants to apply for an hour's slot. Specifically, they were asked to fill a form and choose multiple time slots indicating their availability. We did not have any specific criteria to choose the participants for the study, beyond the notion of first-come-first-serve. The study was conducted over a period of five days in each hour, we had four students be present at the lab to take part in the study. For each participant, the specific system condition was allocated in a round-robin fashion based on their arrival time (i.e. first participant got C1, second got C2 etc.). We obtained data from 59 participants, of whom three were faced with a run-time error. Thus, we ended up with data from 56 participants (13-15 in each condition) of which six were undergraduates and others were graduate students. Out of the 56 participants, a total of 18 students had submitted an iPOS before.

Now we will present, the detailed results from the study. A part of these results were presented in the earlier version (Grover *et al.*, 2019).

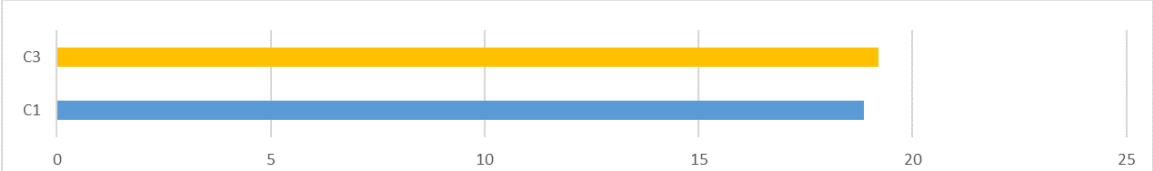


Figure 4.8: Average number of times 'validate' was clicked in condition C_1^1 and C_3^1.

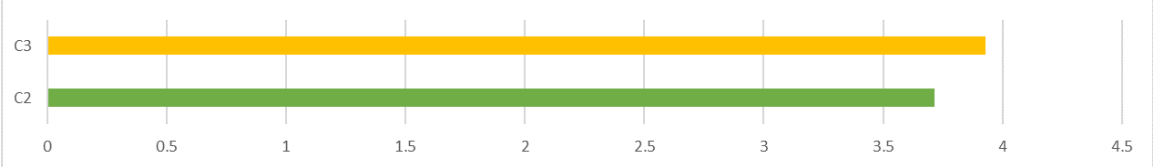


Figure 4.9: Average number of times 'suggest' was clicked in conditions C_2^1 and C_3^1.

4.4.1 Hypothesis H1

H1a. We show the average time a participant took to complete the first and the second iPOS and submit their feedback [1] in Figure 4.6. The data shows a significant improvement in performance with regards to time as one goes from $C_{control}$ to C_3 ($p < 0.05$ for the first and $p < 0.01$ for the second iPOS) showing that the automated planning technologies in conjunction helped in improving the efficiency of the decision making process. Unfortunately, there was no significant improvement seen in performance from (1) $C_{control}$ to C_1 or C_2 and (2) C_1 or C_2 to C_3. Thus, *hypothesis H1a was found to be partially true*, thereby showing that all the planning technologies and not a subset of them were necessary to improve the planning performance for the expert in the loop.

In order to analyze the behavior of the subjects in the different study conditions, we now look at the frequency with which they used different functionalities on the interface – i.e. number of times they checked their solution for submission, and number of times they rearranged, added or deleted actions in the plan, as shown in Figure 4.7. As expected, the average number of checks called in the case $C_{control}$,

[1] Since feedback was part of all the conditions, this is indicative of, even though not the actual, planning time.

which has no plan validation or suggestion support, is the highest, while this value is significantly less for the cases C_3 and C_1 which had validate. Considering that the number of times a user validated their plan in conditions C_1 and C_3 (shown in Fig. 4.8), the use of check did not significantly have an impact on the time taken by the user to finish the iPOS. Also, the average number of times users rearranged actions is almost similar for all the conditions. Interestingly, the average number of times a user clicked delete in the conditions C_2 and C_3, indicates that although they clicked suggest approx. four times in these two conditions (shown in Fig. 4.9), they were not pleased with the plan returned by the automated planning system and landed up deleting (and adding) a lot of actions. This behavior is indicative that the planner failed to capture unspecified user preferences and we believe that the work on building explicable plans (Zhang *et al.*, 2017) will help improve the performance further for the cases C_2 and C_3.

H1b. In Figure 4.13, we show the answers of the users to the subjective statement *Q3: I am happy with the final Plan of Study* on the Likert Scale for all the four conditions. In $C_{control}$, we noticed that the least number of users agreed (either agreed or strongly agreed) with the statement across all the four conditions. This is not surprising because many users were not even able to come up with a valid plan of study without any planning support in $C_{control}$. For C_1, six participants said they were in unison with the statement Q3 and For C_2 and C_3, half of the participants were happier (i.e. either agreed or strongly agreed) with their plan of study, which is the highest across all the four conditions. But, in C_2 there was one participant who strongly disagreed with the statement, which for C_3 there were none. Thus, *the hypothesis H1b holds.*

Note that we mentioned earlier that the users deleted and added more actions for the conditions C_2 and C_3 that can provide action suggestions. In the light of answers

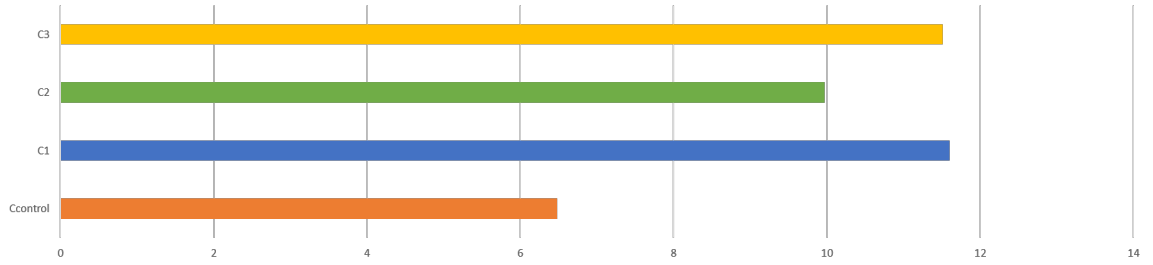


Figure 4.10: Time difference $\Delta T(C_i)$ between two tasks C_i^1 and C_i^2 of iPOS planning for every condition C_i.

to the statement Q3, we find it interesting that although the users had to edit the suggested plan, having a plan available to them to bootstrap on for editing not only made them more efficient, but also increased their satisfaction.

H1c. In Figure 4.14, we show the number of users who agreed with the ratings on the Likert Scale for the statement *Q2: The feedback from the interface helped the iPOS making process.* If we let n_{C_i} denote the number of participants who either agree or strongly agree with the statement, then the following relation holds, $n_{C_{control}} < n_{C_1}, n_{C_2} \leq n_{C_3}$. Although the equality holds n_{C_1} and n_{C_3}, the number of people who strongly agreed to the statement was, by far, the highest for C_3. Thus, we infer that *the hypothesis H1c holds.*

4.4.2 Hypothesis H2

Time to complete the plan will reduce in the second attempt – We plot the average decrease in time in completing the second iPOS after doing the first iPOS with `iPass` for all the four study conditions in Figure 4.10. The lowest reduction in time for $C_{control}$ shows that feedback given to the user by the decision support system helps them learn more about the domain model, thereby improving their performance in making the second iPOS. We also saw that the highest reduction in time occurred for the conditions C_1 ($p < 0.1$) and C_3 ($p < 0.01$). We feel that the presence of plan

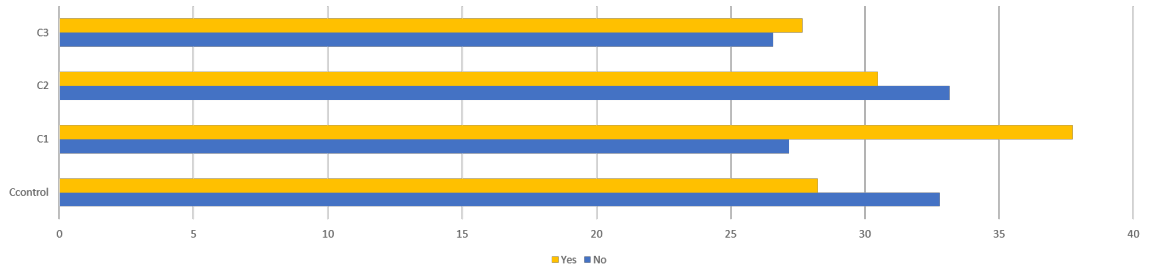


Figure 4.11: Time taken by experienced (in yellow) and non-experienced (in blue) users to make the first iPOS (C_i^1).

validation in both these conditions informed the users about the reason behind each error they made while constructing the first iPOS that was effective in teaching the users about the actual domain. Due to a similar reason, we had also hypothesized that the presence of plan explanations in C_2 and C_3 will reduce the time significantly because these explanations will teach the user about the domain, thus reconciling the models. Unfortunately, this functionality was used very rarely (0.14 and 0.91 average number of times for C_2 and C_3) and thus, improvement in performance was not observed. Hence, *H2 was only found to be partially true*, supporting the cause that use of automated planning C_3 for decision support improved the efficiency of the human thereby reduced the time for making the second iPOS.

4.4.3 Hypothesis H3

Less expert users benefit more from decision support components – We noticed that the performance (time) was *not significantly better* for participants who had filled an iPOS before when compared to participants with no experience (Figure 4.11). Although the experienced participants did perform slightly better in $C_{control}, C_1$ and C_3, to our surprise, we noticed that for C_2, the users who had no prior experience performed better. This might be because the latter group had prior conceptions about the rules of making an iPOS and thus, spent time making plans that appeared valid

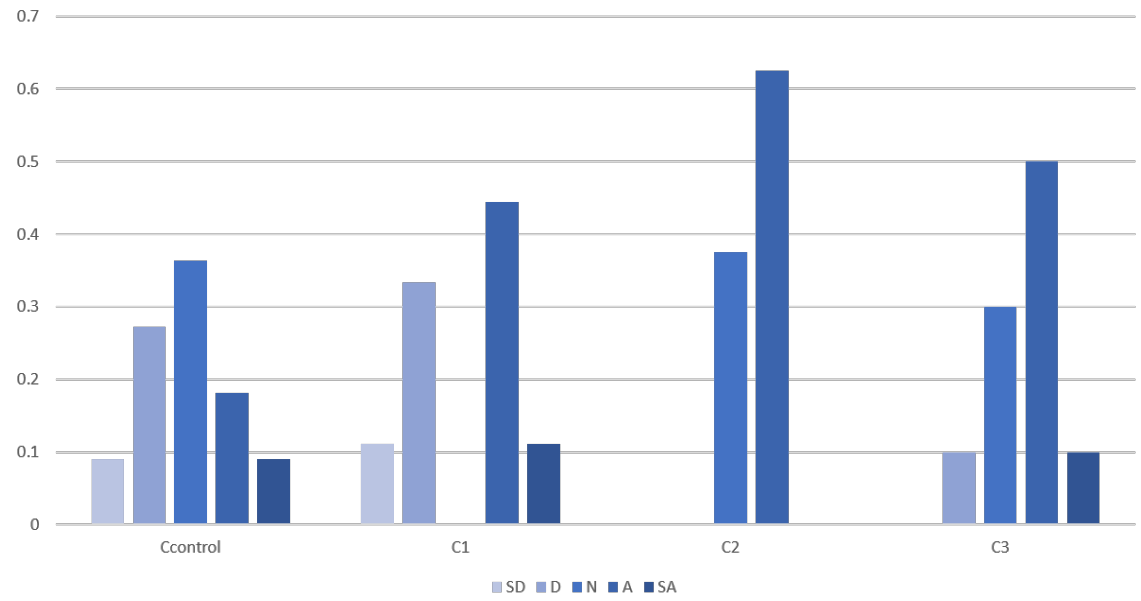


Figure 4.12: Feedback of non-experienced users about the statement 'Q1: The planning task was pretty simple for me' for each condition C_i^1.

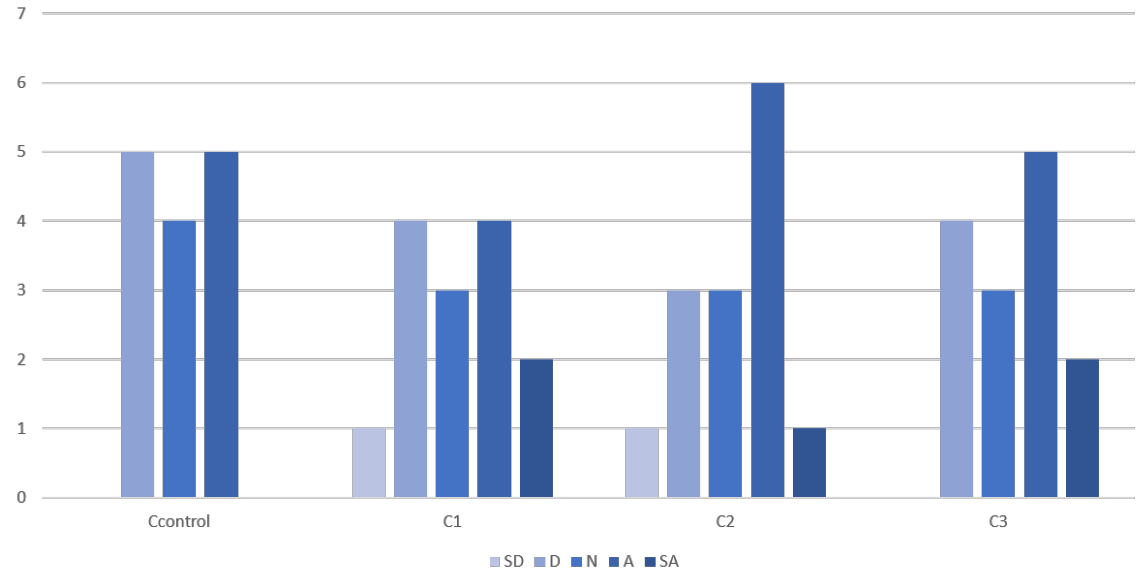


Figure 4.13: Time difference for subjective 'Q3: I am happy with the final iPOS' for conditions C_i^1.

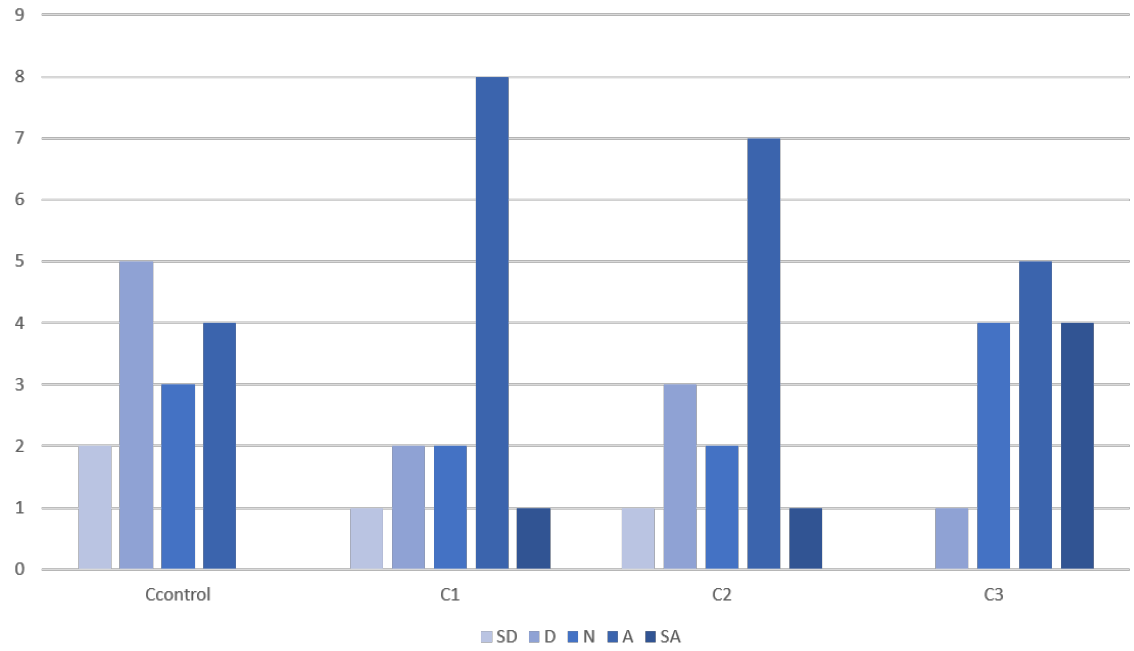


Figure 4.14: User agreement metrics for the statement 'Q2: The feedback from the interface helped the iPOS making process' for each condition C_i^1.

in their model, but were invalid in the `iPass` domain. With the presence of 'validate' in C_1, they might have ended up having to correct their partial plans multiple times, resulting in a longer time and worse performance.

We plot the response of non-experienced users to the subjective question *Q1: The planning task was pretty simple for me* in Figure 4.12. Interestingly, the non-experienced users seemed to agree (or strongly agree) more with the statement in C_3 compared to $C_{control}$, indicating that support features have contributed to decrease in perceived difficulty of the task.

4.4.4 Qualitative Results

We asked three qualitative questions to the users –

Q1. Describe in detail atleast 5 things you liked about the Plan of Study you came up with.

	Measure	Expected	Outcome
H1a.	Time taken	$T(C_{control}) > T(C_1) > T(C_2) > T(C_3)$	$T(C_{control}) > T(C_1) \approx T(C_2) > T(C_3)$
H1b.	Satisfaction iPOS	$n_{C_{control}} < n_{C_1} < n_{C_2} < n_{C_3}$	$n_{C_{control}} < n_{C_1} < n_{C_2} < n_{C_3}$
H1c.	Satisfaction interface	$n_{C_{control}} < n_{C_1} < n_{C_2} < n_{C_3}$	$n_{C_{control}} < n_{C_1} < n_{C_2} < n_{C_3}$
H2.	Time difference	$\Delta T(C_{control}) < \Delta T(C_1) < \Delta T(C_2) < \Delta T(C_3)$	$\Delta T(C_{control}) < \Delta T(C_1) \approx \Delta T(C_2) \approx \Delta T(C_3)$
H3.	Time taken less experienced	$T(C_{control}) > T(C_1) > T(C_2) > T(C_3)$	$T(C_{control}) \approx T(C_1) \approx T(C_3) \approx T(C_2)$

Table 4.2: Summary of Results

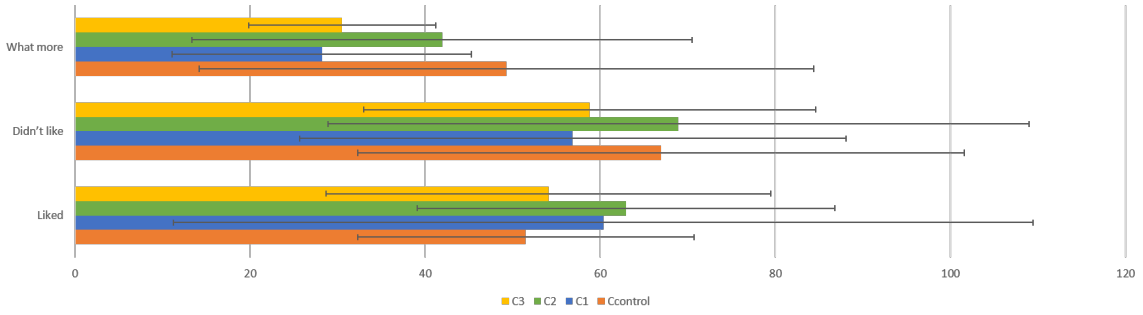


Figure 4.15: Average word count for every feedback question, with error bars showing ±1 standard deviation for the word count. "Liked" is 5 things you liked about your iPOS, "Didn't Like" is 5 things you didn't like about your iPOS and "What more" is what other features of the interface you would like to have.

Q2. Describe in detail atleast 5 things you did NOT like about the Plan of Study you came up with.

Q3. Describe in detail what other features of the interface you would like to have.

Figure 4.15 shows the average word count for the questions with error bar is ± 1 standard deviation. There were three cases where the word count difference was statistically significantly (< 0.05). First, for Q3, the word count was lower for C_3 compared to $C_{control}$. This implied that users in $C_{control}$ requested for many features while, provided with the added functionalities of validate and suggest, their enumeration of *what more* came down significantly. Second, we had a similar conclusion in the case of C_1 and $C_{control}$. This result raises an interesting question– if we were able to significantly reduce the users demand for more features with only the validate functionality in C_1, what added purpose did the suggest and explain functionality in C_3 serve? Third, lower number of participants liked the system (Q1) compared to the ones who disliked it (Q2) for the condition $C_{control}$.

Having read multiple feedback, we found that participants preferred the "Validate" functionality more compared to the "Suggest" functionality because it pointed out specific errors when they were stuck (although this was not statistically significant). For Q3 in $C_{control}$, participants said that they wanted specific errors that would show why a submitted iPOS check fails; in other words, the felt that plan validation functionality would have been helpful. In condition C_3, feedback for Q3 was more in regards to personal preferences, showcasing that users started caring about plan quality when, due to decision support functionalities, coming up with valid iPOS was no longer a challenging task. We further highlight some of the interesting feedback as a part of the future research directions.

4.5 Discussion and Future Work

As mentioned above, the decision support described in this work uses a set of domain-independent techniques in automated planning that provide the back-end functionalities of plan validation, summarization, suggestion, explanation, etc. While these can be used as a plug-and-play, there exists some aspects of the decision support system that needs to be catered to the specific domain for it to be effective. For example, landmarks which provide relevant information to keep a commander aware of their goal in `iPass` ceases to be important for `iPass` where subjects have access to an iPOS handbook. Furthermore, it also helps us to identify some shortcomings of current planning technologies necessary to make the decision support effective in real-time. Before ending the section, we briefly talk about the suggestions provided in the subjective user feedback, highlighting directions for possible future research. We finish the section by highlighting the connections between HCI and AI.

Domain Specific Designs There are various components of the decision support system that need careful attention when it is used in the context of a specific domain. The foremost among this is the user-interface. For example, while a resource panel was useful in the context of a fire-fighting scenario (in `iPass`), it was replaced by a panel that showcases the student information for the `iPass`domain. In domains that are close to scheduling problems, it is often necessary to revamp the entire user-interface design (Mishra *et al.*, 2019). The use of domain-independent technology in the back-end ensures that beyond cosmetic changes to the front-end, all functionalities can be provided with little effort.

Scalability of Back-end Technologies Many of the back-end technologies suffer from scalability issues. When domains become complex, finding optimal plans

within a reasonable amount of time becomes difficult. This leads to longer wait times when the plan or the action suggestion modules are called.

A scalability *vs.* verbosity trade-off exists in the case of generating explanations based on model reconciliations. While minimally complete explanations help the human understand the validity (and optimality) of the suggested plan, time taken to compute makes them unusable in the context of real-world settings like `iPass`. Furthermore, explanations that can be computed faster (for eg. plan patch explanations), are often verbose, adding to the existing cognitive overload of the human-in-the-loop. Furthermore, explanations provided should ideally be the start to conversation that helps users elicit either their preferences or (expert) knowledge about the domain. Thus, research that facilitate the two way communication between the decision support and the human-in-the-loop, thereby learning from one another, could be an interesting future work.

User Feedback Depending on the condition assigned to a particular user, their feedback varied considerably. While users in $C_{control}$ asked for features like validation, users in C_3 expected the system provide suggestions that are more personalized for them. Given that different subjects, with different student information assigned to them, can belong in a spectrum of preferences, solutions that generate explicable plans (Kulkarni *et al.*, 2016) cannot be simply used out-of-the-box. The reason being that they assume all human models come from the same distribution.

HCI and AI

Maes discussed her vision of intelligent software agents that would know the user's interest and act autonomously on their behalf. She divided the task of creating such agents into – (1) knowledge gathering or learning models from the data, and (2) then

utilizing them to support the users (Maes *et al.*, 1997; Maes, 1995). In this paper, we presented an end-to-end software agent which assumes knowledge about user's capabilities and collaborates with them by providing support for sequential decision making. We used ideas from both HCI and AI community to make this software, such as, 'design principles' for the interface (Parasuraman *et al.*, 2000; Amershi *et al.*, 2019), ideas from 'automation' to understand the modules to be automated and the degree of automation (Parasuraman *et al.*, 2000; Sheridan and Verplank, 1978) and 'automated planning techniques' to implement the system (Chakraborti *et al.*, 2019b; Sheridan and Verplank, 1978; Ramírez and Geffner, 2009).

In the past, there have been suggestions that HCI and AI are two different communities with the common focus (Grudin, 2011). The earlier work was related to integrating smaller components to the system, such as integrating email classifier to the email management software (Horvitz, 1999). Combining these components to the software had difficulties in their own right, for example, to ensure that the user may not miss an important e-mail. Through `iPass` we have not just designed an intelligent component for a system, but rather created an intelligent software agent bringing these fields together.

4.6 Conclusion

In this article, we presented a decision support system that uses automated planning techniques to support sequential planning problems for a human-in-the-loop. We first introduced `iPass`and described how different planning technologies such as validation, plan recognition, landmarks, model-reconciliation based explanations can be used to aid a human commander in a time-critical domain. We then situate the various capabilities of `iPass`on the automation hierarchy, carefully describing the design choices we deliberately make. Unfortunately, testing the effectiveness of such

systems became challenging given the lack of experts. To address this challenge, we designed a test-bed system, `iPass` that enables support for a domain in which university students (our subjects for the study) are already experts.

The effectiveness of the system was measured by creating different study groups using different sub-set of capabilities of the system *vs.* a control group. The evaluation was based on their (1) time taken to complete the planning task, (2) time taken to perform similar tasks across multiple trials and (3) the effect of their expertise level. In summary, we found that two key decision support components – validation and suggestion – for human-in-the-loop planning tasks were useful in improving the performance and/or satisfaction of the decision-maker. From subjective feedback, we found that 11 students asked for more feedback from the interface in $C_{control}$ (3 of whom mentioned feedback that can suggest new courses and 5 mentioned validation kind of feedback) thus, highlighting the role of the support components for the normative expectations of the user. We also believed that providing explanations to the users will have a positive impact on decision support but after the study, we realized that we need to learn accurate human models to provide personalized decision support and explanations.

Chapter 5

MODEL ELICITATION THROUGH DIRECT QUESTIONING

Today, robots are working in several domains, such as in a kitchen to cook a meal (Bollini *et al.*, 2013), in restaurants as service robots, [1] etc. Robots need to work individually for these tasks, and in the future, there will be a need to closely collaborate with us and provide active support to complete our tasks and activities. For improved collaboration there will be a need to know and understand our true mental model, that must be either learned or provided to these systems. However, these models can be inaccurate and the system might need to actively update them through interaction with the teammate. In this work, we look at how an automated system/agent can construct directed questions to elicit specific parts of the human mental model. Please note, that we use *robot* or *agent* interchangeably for the automated system and *user* or *teammate* for the human member of the team.

Asking questions to learn more about someone is an age old method. Especially in *education* where human tutors apply several questioning strategies to learn about the student's model. One of those strategies, *diagnostic questioning* (Fairbairn, 1987) asks multiple choice questions where every choice sheds light on student's misconceptions. For example, consider asking students a speed, distance and time problem, such as, calculate the time taken to travel 100 miles at a speed of 50 miles/hr. One of the options could be $50 * 100 = 5000$, and its selection would mean that student fails to understand the relation between speed, distance and time. On the other hand, automated systems such as intelligent tutoring systems have designed techniques to

[1] http://travel.spotcoolstuff.com/unusual-restaurants/bangkok/hajime-robot-restaurant

ask different sequence of questions, chosen from a set of questions for the student to practice (and not for diagnosis) all the topics (Clement *et al.*, 2013, 2014).

Generally, working with human teammates in their working environment should mean, that the robotic agent neither has a defined set of questions to ask from teammate, nor has the ability to understand everything teammate might convey in answer to any generic query. Thus, the agent has to construct it's queries, and also evaluate possible responses for each one of them. These queries have to question different behaviors in the environment. Using diagnostic questioning methodology, ideally the agent would want to construct a query, where every possible response points to a specific model in the set of potential models. However, constructing such a query can be computationally expensive or it could be cognitively overwhelming for the human teammate to solve. Thus, the constructed queries have to consider the teammate's computational capabilities, should be easy to understand for the teammate and the agent should be able to parse the response to update it's underlying models.

Another challenge faced while constructing questions that are not difficult to understand and do not overwhelm the teammate is the lack of understanding about the way humans think or handle information. The education community has spent decades to understand how students conceptualize or interpret knowledge (Ortony and Rumelhart, 1977), such as, knowledge can be in the form of concept maps (Novak and Cañas, 2006) or in the form of causal rules (Shultz, 1982), or the form of *concept images* (Vinner and Hershkowitz, 1980). There has also been some work on how we use these knowledge structures by understanding how students construct formal mathematical proofs (Moore, 1994). There have been studies to test various hypotheses, but the research is inconclusive in general conditions.

Knowledge Elicitation. The field of knowledge engineering was motivated to construct domain-specific expert systems to perform human tasks. Expert systems were

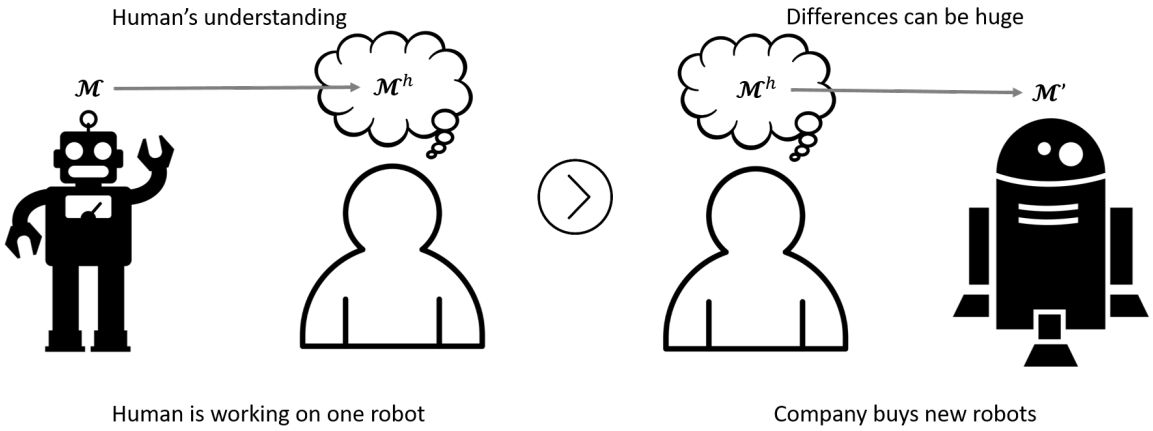


Figure 5.1: A human teammate is working on one of the robots and has an understanding $\mathcal{M}^H$ of robot's model ($\mathcal{M}$). When the company buys a new robot (model $\mathcal{M}'$) but the human teammate's model $\mathcal{M}^H$ does not change and can cause damage to the robot or effect the collaboration of the team.

automated systems that were designed to execute expert-level tasks. Engineering these systems required an understanding of the cognitive processes employed by humans to solve a task (Cooke, 1999). In the 90's knowledge engineering for expert systems was divided into two separate processes of knowledge acquisition and representation (Cooke, 1999). Knowledge acquisition represented the process of acquiring the expert model, and representation meant representing the knowledge for inference. Domain knowledge represented as "facts" and "rules" took center stage constructing these systems (Feigenbaum, 1989).

Knowledge acquisition for expert systems consists of different activities such as knowledge elicitation, which is also considered a sub-field of knowledge engineering (Regoczei and Hirst, 1992). Knowledge elicitation refers to explicating knowledge structures underlying human performance. In the past, expert systems were provided these knowledge structures, and knowledge elicitation was confused with the representation. However, it slowly evolved to understanding the expert solving strategy that could not be attributed to strategy, as much as to domain-specific facts and rules

(Glaser *et al.*, 1988). Thus, the community realized the importance of eliciting expert knowledge from the tasks where simple representation methods were inadequate.

Initial methods for knowledge elicitation were based on measuring performance to compare knowledge between different experts, such as measures of reaction time and error rate (Bailey and Kay, 1986). Some methods involved direct extraction of knowledge through expert responses. However, these methods were riddled with error and bias, and the expert's verbal report and intuitions were often flawed (LaFrance, 1992). It has been shown that different knowledge elicitation methods may tap different types of knowledge (Hoffman, 1989). It has also been observed that there is a knowledge elicitation gap between verbal reports and the performance, sometimes due to implicit and explicit knowledge (Broadbent *et al.*, 1986). The more recent methods involved model construction for the expert's knowledge which could represent reality to a varying degree (Compton and Jansen, 1990).

(Cooke, 1999) divides knowledge elicitation methods into four different categories – observation, interviews, process tracing, and conceptual methods. Observation refers to observing an expert while solving a simulated or a contrived task (Hoffman *et al.*, 1995). Interviews are usually the most frequently employed methods to elicit expert knowledge based on asking unstructured questions (Cullen and Bryman, 1988). Process tracing methods were used for procedural knowledge and refer to using the data of the sequential tasks to elicit expert knowledge. Some of the popular methods are the "think-aloud" technique while solving a problem (Van Someren *et al.*, 1994) or providing verbal reports of tracking an expert or interviewing them (Ericsson and Simon, 1984). Finally, conceptual methods refer to learning domain-dependent concepts or conceptual structures using relatedness information taken from experts interviews (Cooke, 1999). Please note that these tasks are based on either learning from some traces or asking questions.

Our work looks at knowledge elicitation in an active teammate scenario, where the expert is working in the same environment as the automated system. We assume that the system's knowledge about the user's model is in the form of structured rules and facts, and the agent wants to elicit knowledge about the subset. Thus, it asks directed questions to the user to elicit information about the specific rule, and the agent refines the set of hypotheses based on the user's response.

In general, a question consists of two parts – (1) the content of the query, and (2) the set of possible answers. One needs to ask a question where the answer can provide relevant information. However, an automated agent has structured knowledge with the closed-world assumption to calculate and evaluate their decisions and lacks any understanding of the human's mental model. Thus, one of the major challenges faced by an agent is to find a way to question the behavior of the teammate and understand various responses by the teammate to update their structured model.

Human-in-the-loop planning (Kambhampati and Talamadupula, 2015), suggests to incorporate the teammate's model in the robot's planning process. This work supports the idea of hilp, where the agent can refine the robot's understanding of the teammate's model. These questions should be easy to understand and directed to refine the model. Thus, the major contributions for this work are –

(C1) generate questions while accounting for the gap between the structured model and the unstructured (or unknown structured) model of the teammate,

(C2) incorporate the effect of uncertainty for generating questions,

(C3) derive relevant information from simpler answers like – (1) yes/no answers, or (2) the complete plan if possible.

5.1 Background

In the intelligent tutoring system community, there has been a lot of work to represent the knowledge of a student, usually based on the number of problems they can solve. Sometimes, it can be represented using directed graphs, to depict a dependency among different knowledge concepts called *knowledge spaces* (Falmagne *et al.*, 2006; Doignon and Falmagne, 2015). We also use such dependency graphs to represent a robot's knowledge about their teammate's model using STRIPS (Fikes and Nilsson, 1971) as the causal dependency can be represented using cause and effect for an action. Robust planning model (Nguyen *et al.*, 2017) is used to represent the uncertainty, as some conditions that may or may not be a part of the human's model of the environment. These probable conditions for preconditions and effects are called annotations. We will now formally describe the annotated model.

The correct human's model is represented as $\mathcal{M}_H$ is one of the concrete models in the set of models $\mathbb{M} = \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n\}$. $\mathbb{M}$ is represented using a super-set of predicates and operators of each models as $\mathbb{M} = \langle \mathcal{F}, \mathcal{A} \rangle$. $\mathcal{F}$ are the set of propositional state variables or *facts* and any state $s \in \mathcal{F}$. Any action $a \in \mathcal{A}$ is defined as $a = \langle pre(a), eff^{\pm}(a), \diamond pre(a), \diamond eff^{\pm}(a) \rangle$, where $pre(a), eff^{\pm}(a)$ are certain preconditions and effects and $\diamond pre(a), \diamond eff^{\pm}(a) \subseteq \mathcal{F}$ are the possible preconditions and possible add/delete effects, where $\diamond$ being used to differentiate between the certain and possible predicates. It implies that a precondition and effect may or may not be present in the original model and are used to model the uncertainty in the robot's understanding of the human's model. If their are n uncertain preconditions and effects then $|\mathbb{M}| = 2^n$.

A planning problem in the domain is given by $\mathcal{P} = \langle \mathbb{M}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{I}$ is the initial state and $\mathcal{G}$ is the goal. $\delta_{\mathcal{M}}$ is the transition function $\delta_{\mathcal{M}} : S \times \mathcal{A} \rightarrow S$. It can not be executed in a state $s \not\models pre(a)$; else $\delta_{\mathcal{M}}(s, a) \models s \cup eff^{+}(a)/eff^{-}(a)$. There are two

different transition functions for the condition when a specific action in a plan can't be executed (due to $\diamond pre(a)$ being true and not planned for), then the plan either goes to a fail state (pessimistic) or the next action can still be executed (optimistic) in the plan (Nguyen *et al.*, 2017). An agent assumes an optimistic approach as it is concerned about the interaction and the information about the feasibility of the action. A plan π for the model $\mathcal{M}$ is a sequence of actions $\pi = \langle a_0, a_1, ..., a_n \rangle$. A plan is a valid plan if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. A set of plans Π are $\Pi = \{\pi | \forall \pi \delta_{\mathcal{M}}(\mathcal{I}, \pi \models \mathcal{G}\}$

Motivating example

Figure 5.1 shows the situation where a human teammate working with one robot has to migrate to a new one. We now describe the *move* action for the new robot compared to the earlier model. For a robot (such as fetch [2]) it is essential to ensure that the robot's torso is NOT in the stretched state and the arm is tucked close to the body, else it can lead to a fatal accident due to its tendency to topple while turning. The previous model of the robot had no such requirements. The domain for the new robot with uncertainty about the preconditions of a move action, i.e. *hand_tucked* (hand not stretched) and *is_crouch* (torso not stretched). The robot's uncertain model of the human is –

```
(:action tuck
:parameter              ()
:precondition           ()
:possible-precondition  ()
:effect                 (and    (is_crouch)
                                (hand_tucked))
:possible-effect        ()
)
```

[2] https://fetchrobotics.com/robotics-platforms/fetch-mobile-manipulator/

```
(:action crouch
:parameter              ()
:precondition           ()
:possible-precondition  ()
:effect                 (and    (is_crouch)
:possible-effect        ()
)
(:action move
:parameter              (?from ?to - location)
:precondition           (robot-at ?from)
:possible-precondition  (and    (is_crouch)
                                (hand_tucked))
:effect                 (and    (robot-at ?to)
                                (not (robot-at
                                ?from)))
:possible-effect        ())
```

There are two unknowns preconditions hence there are four possible models. If the objects are defined in the model are – *roomA* and *roomB*. Then a question Q, used to elicit human's understanding can be defined as $Q = \langle \mathcal{I}, \mathcal{G} \rangle$, i.e. a planning task,

$\mathcal{I}$ = {robot-at(roomA)},

$\mathcal{G}$ = {robot-at(roomB)}

and the possible answers in each model are –

π_1 = $\langle$move(roomA, roomB)$\rangle$,

π_2 = $\langle$crouch, move(roomA, roomB)$\rangle$,

π_3 = $\langle$tuck, move(roomA, roomB)$\rangle$,

π_4 = ⟨crouch, tuck, move(roomA, roomB)⟩

π_1 shows that there are no preconditions in the human's model, π_2 means the precondition of is_crouch is present, and π_3 shows that human has hand_tucked precondition and π_4 both preconditions are true. Thus, based on the response, the robot can find the specific model of the teammate.

An important point to note is that there might be a correct model for the environment that might be known to the automated agent. However, the agent is only trying to learn the exact human model $\mathcal{M}_H$. The correct model might be different from the human model and may or may not be part of the set of models $\mathbb{M}$.

Similarly, if we assume that the human has a structured model in the form of STRIPS action or some other causal format, which might be possible in the case of experts. Then the robot can ask if a specific precondition or effect is part of the action in their model. The agent needs to ask these questions due to a lack of knowledge about the human's thought process and understanding of the environment.

5.2 Problem Formulation

After describing the robot's understanding of the teammate's model, we are now ready to formally define the questions to elicit the behavior of the teammate.

Definition 1. Question is a tuple, $\langle \mathcal{I}, \mathcal{G} \rangle$, i.e. initial and the goal states. The solution to a question are plans π such that $\delta(\mathcal{I}, \pi) \models \mathcal{G}$. A sequence of questions as $Q = \langle \langle \mathcal{I}_0, \mathcal{G}_0 \rangle, \langle \mathcal{I}_1, \mathcal{G}_1 \rangle, ..., \langle \mathcal{I}_n, \mathcal{G}_n \rangle \rangle$, the tuple of sequential planning tasks where predicates and actions are based on the agent's uncertain model.

As per the definition, a *question* does not take into account the response from the user. In a lot of previous work, responses have been to define the state of failure (Verma *et al.*, 2020), or provide specific model change and the model as the response

(Bryce *et al.*, 2016). However, we believe that giving such responses can be computationally taxing for the teammate. The question was framed using a structured model, but the responses are simple like – (1) whether the query has a valid plan, and (2) the plan for the query in the teammate's model. Before presenting our analysis for each type of query in the next section, we first define the problem of asking questions from the teammate and then discuss how uncertainty in the model affects these queries and the plan constructed by the teammate.

Thus, interaction with the user is through structured questions, and the response is in the form of a plan. For now, we assume that the user is responding with a plan, but it can be computationally taxing for them to construct it. Thus, we look for different responses in the next section, when we define a distinguishing query. Now we formally define the problem of asking questions to the user.

Question Framing Problem

Given the tuple $\langle \mathbb{M}, \mathcal{I}_e \rangle$, find the sequence of questions Q to learn the correct model with $|Q| \leq n$, where n is the number of possible predicates.

$\mathbb{M}$ represents the annotated model of the robot and $\mathcal{I}_e$ represents the fully-specified input state of the environment in which we are interacting with the human-in-the-loop. Defining the initial state of the environment is useful, to support the interaction between a teammate and the robot. Initial state $\mathcal{I}_q$ for any query q can be specified as a set difference from the fully-specified initial state of the environment $\mathcal{I}_e$.

Set of solutions

in a specific model $\mathcal{M} \in \mathbb{M}$ to a particular question Q are the plans $\Pi(\mathcal{M}, Q) = \{\pi | \forall \pi : \delta_{\mathcal{M}}(\mathcal{I}_Q, \pi) \models \mathcal{G}_Q\}$. In this paper we will write it as $\Pi_Q^{\mathcal{M}}$. The cost of a plan is represented as $C(\pi) = \sum c(a)$, where $c(a)$ is the cost of an action in the model $\mathcal{M}$.

π^* is the optimal plan in the set $\Pi_Q^{\mathcal{M}}$ and C^* represents the cost of the optimal plan, i.e. $C^* \leq C(\pi'), \forall \pi^*, \pi' \in \Pi_Q^{\mathcal{M}}$.

Constrained and Relaxed models.

As we described earlier, for any n possible pre-conditions and effects, hence, there is an exponential set of possible models (2^n). It is computationally infeasible to analyze all of them to generate questions. Thus, we look at two extreme models in the robot's model – (1) most constrained model $\mathcal{M}_{con}$ and (2) most relaxed model $\mathcal{M}_{rel}$. $\mathcal{M}_{con}$ is the model where –

- $pre(a') = pre(a) \cup \diamond pre(a)$,
- $eff^+(a') = eff^+(a)$, and
- $eff^-(a') = eff^-(a) \cup \diamond eff^-(a)$,

where a' is the transformed action in $\mathcal{M}_{con}$. On the other hand, $\mathcal{M}_{rel}$ is the model where –

- $pre(a') = pre(a)$,
- $eff^+(a') = eff^+(a) \cup \diamond eff^+(a)$,
- $eff^-(a') = eff^-(a)$

where a' is the transformed action in $\mathcal{M}_{rel}$. Let $\Pi^q_{\mathcal{M}_{con}}$ represent the set of solutions to any question q in model $\mathcal{M}_{con}$ and similarly, $\Pi^q_{\mathcal{M}_{rel}}$ represent the If a plan exists in $\mathcal{M}_{rel}$, then it exists in the rest of the model and the plan length is minimum in min and highest in the $\mathcal{M}_{con}$ (Sreedharan and Kambhampati, 2018). The cost of actions is the same across all the concrete models in the robot's model (as one of them represents the model of the same hilp), $C^*_{\mathcal{M}_{con}}$ is the cost of the optimal plan in

$\mathcal{M}_{con}$, and similarly $C^*_{\mathcal{M}_{rel}}$ for $\mathcal{M}_{rel}$. The corollary follows from the construction of relaxed and constrained models (Sreedharan and Kambhampati, 2018).

Corollary 1. $C^*_{\mathcal{M}_{rel}} \leq C^*_{\mathcal{M}_{con}}$ and $\Pi^q_{\mathcal{M}_{con}} \subseteq \Pi^q_{\mathcal{M}_{rel}}$.

The corollary follows from the method of construction of these models. Equality in the corollary exists when $\mathcal{M}_{rel}$ is the same as $\mathcal{M}_{con}$, and there is no uncertainty in the model. Now we try to extend these properties to any model in the set of models. We use these models to define whether a plan exists in the set of models for the initial and goal states in the question. Now we look at whether an initial and goal state exists for which we can ask humans to construct a plan.

Human Model.

We assume that the human model is one of the models in the set of robot's model $\mathcal{M}^H \in \mathbb{M}$. Thus when presented with a question q, let $\Pi^q_{\mathcal{M}^H}$ represent the possible set of solutions in the human's model and $C^*_{\mathcal{M}^H}$, is the cost of the optimal plan in the human's model. Given this current setup, we can say a few things about the solution provided by the user to any question.

Corollary 2. Assuming user to be an optimal planner then for a specific question, $C^*_{\mathcal{M}_{rel}} \leq C^*_{\mathcal{M}^H} \leq C^*_{\mathcal{M}_{con}}$.

Corollary 3. If we relax the assumption of the optimal planner then we can say that, $\Pi^q_{\mathcal{M}_{con}} \subseteq \Pi^q_{\mathcal{M}^H} \subseteq \Pi^q_{\mathcal{M}_{rel}}$.

The corollary explains that the optimal solution constructed by hilp follows these bounds. The corollary follows from corollary 1, where the human model is either the most constrained or relaxed model for a subset of possible predicates. For comparison between $\mathcal{M}^H$ and $\mathcal{M}_{rel}$, the $\mathcal{M}^H$ is the most constrained version of the $\mathcal{M}_{rel}$ model

for a subset of constraints. Similarly, while comparing $\mathcal{M}^H$ and $\mathcal{M}_{con}$, we can see that $\mathcal{M}^H$ is the most relaxed version of the $\mathcal{M}_{con}$ model on the subset of original constraints. Showing the bounds for the possible human solution to any question doesn't take the human model into account and applies to any model in $\mathbb{M}$. These models help us analyze different cases to formulate the properties of the queries based on the plans that are possible in these models in the next section.

5.3 Distinguishing Query

Due to the combinatorial explosion, the agent can not search through all possible initial and goal states, and the agent needs to generate a query from the bounds on the human model. Thus, to analyze the specific initial and the goal state, we look at the properties of the plans which should be discussed and then find the initial and goal state that will support their execution. We start our analysis with the simplest case, how to distinguish models with a single probable predicate (i.e., a single unknown predicate in an in the model), and then generate a directed question when there are many possible predicates in the model.

The solution provided by the user depends on the question posed by the agent, and the query depends on the annotated constraint. If p is the predicate possibly (for now assuming it's a pre-condition) present in action a (we should be writing $\mathcal{M}^{a\diamond p}$, but for simplicity, we write $\mathcal{M}^{\diamond p}$). Thus there can be four different models which the agent needs to consider –

* $\mathcal{M}^{-p}_{con}$ – Constrained model with p is not part of it.

* $\mathcal{M}^{+p}_{rel}$ – Relaxed model with p is part of the model.

* $\mathcal{M}^{H-p}$ – Human model with p is not part of the model.

* $\mathcal{M}^{H+p}$ – Human model with p is part of the model.

Please note, a predicate p being part of the model means that it is present in the specific action a, for which it was a possible predicate. It does *not* mean an abstracted predicate from the model. Also, note that only one of the models $\mathcal{M}^{H-p}$ and $\mathcal{M}^{H+p}$ is the true model. Given, the solution π_H (provided by hilp of cost C_H), and assuming human is an optimal planner, there can be a few possibilities –

1. $C^*_H < C^*_{\mathcal{M}^{+p}_{rel}}$, would mean that $\pi_H \notin \Pi_{\mathcal{M}^{+p}_{rel}}$ as the solution plan cost provided by hilp is less than the optimal plan cost in $\mathcal{M}^{+p}_{rel}$. Thus, $\mathcal{M}^{H-p}$ is the real human model, i.e. the constraint is not part of the human model.

2. $C^*_H \geq C^*_{\mathcal{M}^{-p}_{con}}$, would mean that $\pi_H \notin \Pi_{\mathcal{M}^{-p}_{con}}$ as it is not a valid plan in $\mathcal{M}^{-p}_{con}$. Thus, $\mathcal{M}^{H+p}$ is the real human model, i.e. the constraint is part of the human model.

In principle, the plan for the question posed by the agent to determine a constraint p will be harder to execute in $\mathcal{M}^{H+p}$ and easier in $\mathcal{M}^{H-p}$ and help us distinguish the models, which can be achieved by tweaking the initial state of the query. For example, if p is not part of the initial state, then the human needs to achieve this pre-condition to execute the action, only if the pre-condition is part of the human model. Similarly, we can extend this idea to any possible effect e, where a plan involving the action would be costlier to generate using $\mathcal{M}^{H-e}$, as compared to $\mathcal{M}^{H+e}$ by changing the goal state (in different scenarios, i.e., with or without the effect e). We call such queries *distinguishing query*.

Definition 2. A question Q_p is a *distinguishing query* if the plan for it can distinguish between models $\mathcal{M}^{H+p}$ and $\mathcal{M}^{H-p}$.

Until now, we analyzed different possible models and the set of responses based on the optimal plan in each model. In the next subsection, we look at the properties of the plan that will be provided by the user π_H for yes/no response.

5.3.1 Properties

There are exponential combinations for initial and goal state (exponential in the number of predicates in the model), which can be queried. But we want to ask questions that help us distinguish between a set of models and searching through all possible initial and goal states is not possible. We have outlined the central idea where the action with an unknown predicate has to be part of the plan for a distinguishing query. Now, we will formally describe these properties and use them to generate a query. For this, we introduce a topic from the planning literature called *landmarks* to explain the properties of the plan.

Landmark.

A *landmark* L is a logical formula, where, $\forall \pi : \delta(\mathcal{I}, \pi) \models \mathcal{G}$ and for some prefix of plan $\pi_{pre} = \langle a_0, a_1, ..., a_i \rangle, i < n$, $\delta(\mathcal{I}, \pi_{pre}) \models L$. An *action* landmark is for any action a,$\forall \pi : \delta(\mathcal{I}, \pi) \models \mathcal{G}, a \in \pi$ (Keyder *et al.*, 2010). For example, the $\mathcal{G}$ is a trivial landmark and if there exists only one action a that reaches a particular fact $f \in \mathcal{G}$, then the action is an action landmark. Finding a landmark is a PSPACE-complete problem (Hoffmann *et al.*, 2004).

Proposition 1. Necessary condition for a question Q_p to distinguish between models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ is $a_p \in L(\mathcal{M}^{-p})$, where a_p is the action with proposition p is a landmark in model $\mathcal{M}^{-p}$ for the query.

Proof. Proof is divided into two parts. First, we show that being a landmark is necessary and then we show that landmark has to be of model $\mathcal{M}^{-p}$. For the first part, let's assume if there is a plan π for a distinguishing problem Q_p and such that $a_p \notin \pi$. Then $\delta_{\mathcal{M}^{+p}}(\mathcal{I}_{Q_p}, \pi) \models \mathcal{G}_{Q_p}$ and $\delta_{\mathcal{M}^{-p}}(\mathcal{I}_{Q_p}, \pi) \models \mathcal{G}_{Q_p}$, i.e. the plan can be executed in both the models. Which refutes the assumption that Q_p is a distinguishing problem.

For the second part, observe that $\Pi_{Q_p}^{\mathcal{M}^{+p}} \subseteq \Pi_{Q_p}^{\mathcal{M}^{-p}}$, i.e. every plan possible in more constrained model ($\mathcal{M}^{+p}$) is also a plan in the less constrained model ($\mathcal{M}^{-p}$). Again, if we assume that the action a_p is a landmark in $\mathcal{M}^{-p}$, then there are some solutions $\pi \in \Pi_{Q_p}^{\mathcal{M}^{-p}} \setminus \Pi_{Q_p}^{\mathcal{M}^{+p}}$, where a_p is not a landmark. Following the previous corollary, we can conclude that all plans have the property $a_p \in L(\mathcal{M}^{-p})$. □

An important point to note here is that we do not distinguish between fact p being a pre-condition or effect for action because this condition is necessary for either of them. The agent has to ensure that the action with a possible predicate is a landmark action. One of the methods is to use the *add effects* as the goal of the question. In theory, we can use this method to ensure that the action a is a landmark action in the human's model, but in practice, we need to assume that the human is an optimal planner, and thus the action is an optimal landmark. Removing this assumption is out of scope for this work and will be part of our future work. Proposition 1 can be extended, that by ensuring that action a is a landmark in $\mathcal{M}^{-p}$, it is harder to achieve all pre-condition for the action a in the constrained model $\mathcal{M}^{+p}$. The proposition directly leads to two corollaries that can establish the distinguishing property.

Corollary 4. The distinguishing problem Q_p should have atleast one solution in $\mathcal{M}^{-p}$.

Proof. For the two models, we know that $\Pi_{Q_p}^{\mathcal{M}^{+p}} \subseteq \Pi_{Q_p}^{\mathcal{M}^{-p}}$. Thus, if $|\Pi_{Q_p}^{\mathcal{M}^{-p}}| = 0$, means there are no solutions in both the models and thus it can not distinguish either of them. □

Corollary 5. The distinguishing problem $\mathcal{Q}_p$ should not be solvable in $\mathcal{M}^{+p}$.

Both the above corollaries follow from the way constrained and relaxed models are constructed, where distinguishing problem Q_p should have atleast one solution in

$\mathcal{M}^{-p}$, and no solutions in $\mathcal{M}^{+p}$. Thus, if the problem is solvable (either by executing the plan in the environment or asking the human for the plan in their model) then the fact is fictitious otherwise, the fact is real. The answer to the distinguishing query is a simple yes/no, facilitating the interaction with the human-in-the-loop.

By using the proposition and the corollaries, we can propose a solution for QFP where we ensure that the action (with possible proposition) is a landmark in the less constrained model ($\mathcal{M}^{-p}$). In other words, finding a distinguishing question is about finding a the initial and goal state, where the action is a landmark. We present the algorithm for this in the next section. But, for now we want to continue this analysis of the distinguishing question, to understand how do they effect the space of probable models.

Difference between Pre-conditions and Effects.

The analysis stands true for any possible predicate, be it pre-condition or an effect, and results in different ways of constructing models $\mathcal{M}^{-p}$ and $\mathcal{M}^{+p}$. The basis for constructing these models is that $\mathcal{M}^{+p}$ is the constrained model, and $\mathcal{M}^{-p}$ is the relaxed one. Thus for any possible pre-condition and a delete effect – $\mathcal{M}^{+p}$ is where the predicate is true, and $\mathcal{M}^{-p}$ is where the fact is not part of the action in the model. Conversely, for add effects $\mathcal{M}^{+p}$ – the predicate is not part of the action in the model, and $\mathcal{M}^{-p}$ – predicate is part of the action in the model.

Another difference lies in the fact that how the query should be constructed for a pre-condition or an effect. Here the idea is that $\mathcal{M}^{+p}$, it is difficult to achieve those predicates. Thus, we need an initial state from which the pre-condition is not achievable, and similarly for the effects the goal states have the extra predicate, thus making it difficult for the user to achieve it. In other words, the agent is ensuring

that the teammate will fail in achieving the pre-conditions, if the predicate is part of their model.

5.3.2 Proposition Isolation Principle (PIP)

Plan Generation Queries

In this section, we discuss how an agent can ask questions to ensure that uncertainty about a predicate is not affected due to interaction with another possible predicate. It is a brute force method to ask questions by isolating the predicate, i.e., there will be n questions for n possible unknowns. The PIP method involves constructing the two models –

- $\mathcal{M}^{+p}$ is the most constrained model $\mathcal{M}_{con}$.
- $\mathcal{M}^{-p}$ is $\mathcal{M}_{con}^{-p}$, i.e. most constrained model where the predicate is not present in case of pre-conditions and delete effects, and considered part of the model in case of add-effects.

It naturally follows that the models differ by a single predicate. Using $\mathcal{M}_{con}$ to pursue a landmark essentially helps isolate the specific predicate p by supporting both the presence and absence of other predicates. For example, an action in the plan may have a possible predicate as a pre-condition, then it is added to the initial state of the query.

Proposition 2. Any question Q_p, which distinguishes the model $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$, is isolated by $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$.

Sketch. To prove the statement, we need to understand that the models differ due to the predicate p, thus all the plans will only differ due to the absence (presence, in case of add effect) of the predicate in $\mathcal{M}_{con}$. Since, the plans are being constructed

for $\mathcal{M}_{con}^{-p}$ the agent ensures that other possible or valid constraints (predicates in the action) are either satisfied through the initial state or are easily achievable by actions in the plan. In other words, the plans are possible in all the other models, and the failure is due to the constraint (predicate) p. This completes the sketch. □

The proposition 2 explains that by using the PIP method, the agent can ask questions about every predicate. It works on the idea of providing all the pre-conditions that affect the action a_p. If the user might fail to execute a plan involving a_p, then it is only due to the constraint p.

The teammate will either respond that a plan is possible in their model, or it is not possible to reach the goal. The robot can understand each possible response, where "yes" means hand_tucked precondition is not part of the human model and "no" means it is a precondition in the humans model. An important point to note is that another possible predicate is_crouch was made possible in the initial state, thus, ensuring that if the user responds "no", it could only be because of the presence of hand_tucked precondition, thus explaining the PI principle. This query depends on the idea that *tuck* action is not part of the teammate's model else this query can not distinguish based on a yes/no response from them. Now we discuss when any question is a distinguishing query.

Sufficiency Condition.

The agent uses a $\mathcal{M}_{con}$ model to find the solution with the landmark. However, it doesn't prove that the same or similar plan involving the landmark action would be optimal in the human model. Due to fewer constraints compared to $\mathcal{M}_{con}$, there could be many other actions that could be used by the agent and thus might have a different plan which might not involve the action. Thus, the agent needs to find this scenario and prevent it by increasing the cost of executing other actions.

To understand these scenarios, the agent can convert action to a landmark by adding it's $eff^{+}(a_p)$ as the goal. But it is always possible to have two or more actions that provide a subset of the goal predicates (whose union is the goal set). In the teammate's model, the plan with these actions could be smaller. Thus, the agent needs to ensure that these actions are harder to execute in any less constrained model. Another scenario could be when an action that provides the precondition p for action a_p also satisfies another pre-condition in it. Thus, when the teammate's plan includes the action which provides the possible pre-condition, the agent can't be sure the action is for pre-condition p or not. It could be impossible to check all the action combinations for every possible condition, but once the plan for the query is known in *con*, the number of actions in the plan are finite, and thus evaluating these conditions is feasible.

As explained earlier, it is computationally infeasible to exhaustively evaluate every combination of initial and goal state. Thus, we look at the properties of the response that can help us *construct* with these states. We assume an empty set of propositions (as initial and goal state) and add the propositions that will be required for the valid responses based on the properties –

(P_1) To distinguish the models, a query (initial and goal state) should be solvable only in $\mathcal{M}^{-p}$ and not in $\mathcal{M}^{+p}$, i.e. in the less constrained model only.

(P_2) The possible predicate has to be a landmark (Hoffmann *et al.*, 2004) for the given query, i.e. all the valid plans should make the predicate true at some stage.

(P_3) Due to other uncertain predicates the agent's query needs to distinguish the models $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$, thus ensuring a plan possible in $\mathcal{M}_{con}^{-p}$ will be a valid plan in any $\mathcal{M}^{H}$ except the constrained model $\mathcal{M}_{con}$.

Any query following property P_1 is called a distinguishing query and automated agent needs to follow while asking a query from the teammate. However, it is not sufficient as the property does not incorporate the effect of uncertainty in other parts of the model. Thus, we further discuss property P_2 and P_3.

An action landmark is an action executed at some point along all valid plans that achieve a goal (Van Beek, 2005). Similarly, fact landmark is a predicate that has to be true at some point along all valid plants that achieve the goal (Hoffmann *et al.*, 2004). The property P_2 describes that the action with the possible predicate is a landmark action. It ensures that the possible plans (or responses) from the teammate is relevant to the possible predicate, and can help the robot distinguish between the correct model based on simple answer such as "yes" or "no". Thus, properties P_1 and P_2 are sufficient for a robot to generate a query. However, it is still only true for a single unknown predicate in the model.

Property P_3, incorporates the uncertainty due to other predicates, and uses $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$ for generating a distinguishing query. As per the construction of $\mathcal{M}_{con}$, all valid plans are valid in all relaxed models (including the human's model $\mathcal{M}^H$, through corollary 2). Thus, the plans in $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$ differ due to uncertain predicate p and the generated incorporates the affect of other uncertain predicates. This property is called the *predicate isolation principle (PIP)* and helps the agent in generating queries offline. These three properties are sufficient to ask a query from the user, as the agent has evaluated – (1) possible responses from the teammate, and (2) the effect of other uncertain predicates on the current query. Now, we define three different types of queries that help the agent elicit the teammate's model based on the responses such as a "yes/no" or a $\pi_H \in \Pi_q$.

5.4 Decreasing Questions

In the previous section, we have shown how the agent can interact with the teammate, and ensure that every interaction can be useful. However, if the agent wants to decrease the number of questions, it has to question more than one predicate, and thus, there could be multiple reasons for the infeasibility of the query. There are conditions when both $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ have another feasible solution. Thus, using PIP and these conditions, every subset of the models is bound to have an optimal plan and the agent uses the differing plans to infer the teammate's model. The analysis assumes the teammate is an optimal planner and the positive action cost for the model. We present the step-by-step construction of such questions that we call *templates*, as they can be merged to construct a query for more than one predicate, where every subset of the model has a valid and optimal solution.

Pre-conditions.

For a proposition p which is a possible pre-condition of the action a_p. For the action to executed $pre(a_p)$ can be provided by – (1) initial state, or (2) executing another action a' where $p \in eff(a')$. When the precondition comes from the initial state, then the distinguishing query can construct using an isolated proposition. Now, we discuss how such a template exists and how a query can ensure different plans in both the models ($\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$).

Proposition 3. Distinguishing question for $\diamond p$ of an action a_p has a distinct valid plan in models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ when for another action a', $p \in eff^{+}(a')$.

Proof. Assume for a given distinguishing problem Q_p actions $a \in L(\mathcal{M}^{-p})$ and a' can be executed i.e. $\mathcal{I}_{Q_p} = \{pre(a) \cup pre(a')\} \setminus \{p\}$ and $\mathcal{G}_{Q_p} = \{eff^{+}(a)\} \setminus \{p\}$. For, model $\mathcal{M}^{+p}$ the plan is $\pi = \langle a', a \rangle$. This plan is also a valid plan in $\mathcal{M}^{-p}$. But due

to optimality and non-zero action costs, the plan π is not an optimal plan in $\mathcal{M}^{-p}$ as pre-condition provided by a' is not required in the model to execute action a. Thus, the optimal plan will be $\pi' = \langle a \rangle$ which is distinct. □

Add Effects.

For possible add effects $p = \diamond eff^c(a)$ an action a' such that $p \in pre(a')$. If there is another action a'' where $p \in eff^+(a'')$. Now we will discuss the template in some more detail.

Proposition 4. Given three actions, a, a', a'' where $p = \diamond eff^c(a)$ $\diamond eff^c(a) \in pre(a')$ and $\diamond eff^c(a) \in eff^c(a'')$ will have distinct plans in models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$.

Proof. We will again use proof by construction. Consider a distinguishing proble Q_p, where $\mathcal{I}_{Q_p} = \{pre(a) \cup pre(a') \cup pre(a'')\} \setminus \{p\}$ and $\mathcal{G}_{Q_p} = \{eff^+(a) \cup eff^+(a')\} \setminus \{p\}$. This will ensure actions a and a' are landmarks and a'' can be executed. Now in the case of $\mathcal{M}^{-p}$ (remember constructive effect are part of less constrained model, follows from proposition 2), $\pi = \langle a, a' \rangle$, and for model $\mathcal{M}^{+p}$, $\pi = \langle a, a'', a' \rangle$, which completes the construction. □

The limitation of asking about binary interaction is that the agent is not using the model structure. However, using the templates checks whether a particular causal dependency exists in the teammate's model. If it is not part of the true model then an optimal plan will not use the specific action in the plan that provides a relationship. In this case, we need to be sure there isn't another causal dependency among those actions, and in that case, the agent can't be certain. The PIP principle holds for asking questions like this and the agent has to isolate the predicates that it wants to use for asking questions. Extra care has to be taken to ensure that the predicates

that are used as a template – (1) do not have more than one causal dependency, and (2) do not have mutually exclusive relationships in the plans.

It is possible to combine the templates because every sub-model (with or without the predicates) has a valid plan and if there aren't any negative interactions between pair-wise actions in the plan. Since its not feasible to check all the cases, we construct a planning problem from the initial state of the environment which will be presented in the next section.

Corollary 6. Given no destructive interactions between the actions, the templates can be combined where each sub-space, such as $\mathcal{M}^{+p_1,-p_2}$ will have a distinct plan.

The proof follows from the assumption of lack of destructive interactions as the plan for each sub-space is a union of distinct plans in the template. We can merge the questions and even ask the teammate which plan from the set of plans is valid in their model. If we check the motivating example, it shows the case of two template queries for questioning about is_crouch and hand_tucked predicate in *move* action.

5.5 Different Types of Queries

In the previous section, we described the sufficient properties for generating a "distinguishing query". Although the analysis assumes the human responds with a plan (π_H), in this section, we analyze how the agent can elicit information from it. Binary responses can elicit information about one predicate at a time, whereas a plan can provide a lot more information. We look at three different types of queries – solve, validation, and template and under what conditions an agent can use them to elicit information about the unknown predicates. The solve and validation queries have binary responses (based on the set of plans), and the template query has a plan as the response.

Plan Generation Query (Plan). In this query, the initial and goal state follow the above three properties and the teammate's expected response is either "yes" or "no". Sometimes, it is difficult to understand the reason for a "no" response from the teammate, but due to properties P_2 and P_3, the agent ensures that the failure is due to the probable predicate only. However, this query is hard to construct due to the ergodic environment, as there could be a plan in every model of $\mathbb{M}$. Let's see an example of the query, for which we will assume that the teammate does not know about action *tuck*. Thus the query for this question would be –

$\mathcal{I}$ = {is_crouch, robot-at(roomA)},

$\mathcal{G}$ = {robot-at(roomB)}

The teammate will either respond that a plan is possible in their model, or it is not possible to reach the goal, where "yes" means hand_tucked precondition is not part of the human model and "no" means it is a precondition in the humans model. An important point to note is that another possible predicate is_crouch was made possible in the initial state, thus, ensuring that if the user responds "no", it could only be because of the presence of the hand_tucked precondition, due to isolation of the predicate hand_tucked.

Validation Query (Val). As explained earlier, it is sometimes infeasible to achieve properties P_2 and P_3, i.e. have a valid plan only in $\mathcal{M}^{-p}_{con}$. However, an agent can still ensure sufficient information can be elicited through "yes/no" response to the query, such as, ask the teammate whether a plan exists in their model without using a specific action *tuck* in the previous example. This can overwhelm the teammate as the onus falls on them to find a valid plan in the constrained model (with fewer actions). Instead, the agent can also provide the plan valid only in $\mathcal{M}^{-p}_{con}$. For example, the query for hand_tucked predicate will be –

$\mathcal{I}$ = {is_crouch, robot-at(roomA)},

$\mathcal{G}$ = {robot-at(roomB)}

π = $\langle$move(roomA, roomB)$\rangle$.

Now if the user responds "yes" hand_tucked is not part of the teammate.s model, and if they respond "no" then hand_tucked is part of their model.

Solving (Plan) v\s Validating (Val) query. We have outlined a method that will be discussed in the solution section, but there are still conditions that might ensure the teammate's plan might not include the a_p. An agent can continue to constrain the set of possible plans by removing actions from the model or some other condition. However, these extra additions can overwhelm the teammate as the communication keeps getting complex and can burden the teammate in finding a plan. The agent can ask the query as the initial, the goal state, and the plan. The teammate can respond to the validity of the plan in the model. In this scenario, the agent needs to ensure that the plan can be executed only in the $\mathcal{M}_{con}^{-p}$. Where solving a problem would have given more information, but currently, our framework handles the human as an optimal agent. Thus, the agent can still ask the plan constructed with the initial and goal state in $\mathcal{M}_{con}^{-p}$ for validation.

Template Query. This query is used when there is a feasible plan in both $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$, however the plans differ due to the possible predicate is present or absent in the true model. The plan change is due to the fact that the probable pre-condition is an add effect of an action, which will be used before the action (or would be absent in $\mathcal{M}_{con}^{-p}$. Similarly, in case of the possible $eff^{+}(a_p)$, if another action $a_p^{'}$ has predicate p as a pre-condition and the robot is certain about it, the plan should include these actions in the $\mathcal{M}_{con}$ model and only a subset in $\mathcal{M}_{con}^{-p}$ (only $a_p^{'}$). For example, using

the motivating example, *tuck* action has the effect hand_tucked which is a possible pre-condition for the *move* action. Thus the query for eliciting hand_tucked will be –

$\mathcal{I}$ = {is_crouch, robot-at(roomA)},

$\mathcal{G}$ = {robot-at(roomB)}

and the possible responses are –

$\pi_{\mathcal{M}_{con}}$ = ⟨tuck, move(roomA, roomB)⟩

$\pi_{\mathcal{M}_{con}^{-p}}$ = ⟨move(roomA, roomB)⟩

Two important points to note here are how PIP was applied for isolating the hand_tucked precondition and the answer to the same question (as plan generation query), requires more information to refine the teammate's model. An important point to note is that these templates can be combined to query about multiple predicates, the construction ensures a different optimal response in every model. For example, the motivating example consists of query for two different predicates hand_tucked and is_crouch predicates. Thus, this query can be used to gather more information in fewer interactions.

5.6 Proposed Solution

As we described in the earlier sections, a query is an initial and goal state in which the action a_p is an optimal landmark in the model $\mathcal{M}_{con}^{-p}$. The easiest way to achieve this is to define the goal state as the add-effects of the action a_p. Finding an initial state is difficult because we need to ensure that given any possible pre-conditions in effect, the interaction should provide information. Thus, in this section, we describe two algorithms – (1) to iterate over each unknown predicate and decide whether templates can be combined, and (2) for generating the query.

The parent routine is to iterate over each unknown in an order decided by $\mathcal{M}_{join}$, which assumes that all the unknown predicates are true. This model can't be used for

Algorithm 1: Query Generation Algorithm (QGA)

Input : $\mathcal{I}_e, \mathcal{M}_{con}^{-p}, a', a_p$

Output: $\langle \mathcal{I}_q, \mathcal{G}_q, \pi_q \rangle$

begin

  $\mathcal{G} \leftarrow pre(a') \cup pre(a_p)$;

  $\pi \leftarrow Solve(\langle \mathcal{M}_{con}^{-p}, \mathcal{I}_e, \mathcal{G} \rangle)$;

  $\pi_q \leftarrow \langle \pi, a_p \rangle$;

  $\mathcal{I}_q, \mathcal{I}_{temp} \leftarrow \text{Project}(\mathcal{I}_e, \pi)$;

  $\mathcal{G}_q, \mathcal{G}_{temp} \leftarrow eff^+(a_p)$;

  for $a_x \in \{a | a \in \pi_q \& eff^+(a) \in \mathcal{G}_q\}$ **do**

    $f \leftarrow \{f | f \in eff^+(a_x) \& f \notin \mathcal{G}_q\}$;

    $\mathcal{G}_{temp} = \mathcal{G}_{temp} \cup \neg f$;

  end

  for $a_x \in \{a | a \in \pi_q, p \in eff(a), pre(a_p) \setminus p \in eff(a)\}$ **do**

    $f \leftarrow \{f | f \in pre(a_p), f \in eff(a_x), f \neq p\}$;

    $I_{temp} = I_{temp} \cup f$;

  end

  $\pi \leftarrow Solve(\langle \mathcal{M}_{con}^{-p}, \mathcal{I}_{temp}, \mathcal{G}_{temp} \rangle)$;

  if $a_p \in \pi$ **then**

    return $\langle \mathcal{I}_{temp}, \mathcal{G}_{temp}, \pi \rangle$;

  else

    return $\langle \mathcal{I}_q, \mathcal{G}_q, \pi_q \rangle$;

  end

end

analysis, as it is neither most constrained nor most relaxed, and the plans generated may or may not be part of another model. Then we construct a relaxed planning graph with pair-wise mutexes called graph-plan planning graph (Kambhampati *et al.*, 1997). It also supports the construction of templates by checking the conditions explained in propositions 3 and 4 are satisfied and mutexes for plans to merge them.

Selecting Sequence of Queries First, the $\mathcal{M}_{con}$, $\mathcal{M}_{rel}$, and $\mathcal{M}_{join}$ models are constructed from $\mathbb{M}$. $\mathcal{M}_{join}$ assumes that all the unknown predicates are true. This model can't be used for analysis, as it is neither most constrained nor most relaxed, and it can't be shown that the plans generated in this model are part of other models too, such as in case of $\mathcal{M}_{con}$. But, it is useful for discerning all the threats to the current plan returned by the query. In other words, this provides an exhaustive set of causal links and possible threats that is possibly part of other models. In line 3, we constructed a relaxed planning graph with pair-wise mutexes called graph-plan planning graph (Kambhampati *et al.*, 1997). It is used to find threats to the plan sequence returned in the query generation algorithm as well as to merge the template queries. Then we iterate over every p, a_p combination to generate a query for each unknown predicate. The returned template queries are merged based on pair-wise mutual exclusion among plans, the initial, and the goal state.

Algorithm 1, generated queries for specific p, a_p pairs. First, it solves a planning problem, where the plan is to reach the preconditions of the action a_p in the model $\mathcal{M}_{con}^{-p}$. If the query is for the template, then preconditions of other actions is used as well (follows from proposition 3 and 4). The solution of the planning problem and then executing action a_p as the goal is $eff^{+}(a_p)$. The projection function finds the subset of the initial state for constructing the plan, to ensure that other plans are not feasible in the teammate's model. Then we satisfy sufficiency conditions for the plan

using $\mathcal{M}_{join}$. From lines 7-10, we handle every effect in a_p that might be provided by other actions. The negation of the pre-condition from these actions is added to the initial and goal state. From lines 11-13, the algorithm checks if an action in the plan threatens p. The threat is handled by removing the action from the plan and adding its constraints to the initial state. Finally, it reevaluates whether the updated initial and goal state constructs the plan in the constrained model or not. If the plan still contains the action a_p then the query is to validate the plan π_q, instead of asking them to generate the plan. This solution follows the complete analysis to ensure a sufficient and minimal response from the teammate. The agent can ask the query in any order due to PIP and sufficiency conditions because other constraints do not affect the current query for any predicate. However, the sequence was derived from the graph-plan planning graph for $\mathcal{M}_{con}$, based on the action closer to the initial state.

5.7 Empirical Evaluation

We have theoretically discussed the process of generating questions. The question generation method uses APDDL parser (Nguyen *et al.*, 2013) based on PDDLPy [3] , and an optimal planner Fast downward (Helmert, 2006) to solve the planning problems. The results reported are from experiments run on a 12 core Intel(R) Xeon(R) CPU with an E5-2643 v3 @3.40GHz processor and a 64G RAM. The experiments were performed on – rover, blocksworld, satellite, and zenotravel [4] . The IPC domains were the correct human model, and randomly chosen predicates were assumed as possible predicates. An equal number of predicates were added to the actions based on the parameters for the action. Special care was taken, to ensure that the

[3]https://pypi.org/project/pddlpy/

[4]https://github.com/potassco/pddl-instances

	$\lvert \diamond p \rvert$	$\lvert \mathcal{Q} \rvert$	Val	Plan	Templ	Time
Blocks	4	3.7	1.9	0.7	1.1	1.79
	6	5.4	3.1	0.8	1.5	5.62
	8	7.4	3.8	1.1	2.5	12.24
Rover	4	3.8	2.0	0.8	1.0	2.21
	6	5.6	3.1	0.8	1.5	7.89
	8	7.3	4.0	1.5	1.8	13.24
	10	9.4	4.8	2.4	2.2	29.53
Satellite	4	3.7	1.8	0.8	1.1	2.11
	6	5.5	2.9	1.2	1.4	6.48
	8	7.4	3.9	1.8	1.7	13.69
	10	9.3	4.8	2.5	2.0	25.55
ZenoTravel	4	3.7	1.8	0.9	1.0	2.05
	6	5.4	3.0	1.0	1.4	5.93
	8	7.4	3.9	1.5	2.0	12.97

Table 5.1: Comparison of different types of Queries.

Objects	$\|\mathcal{Q}\|$	Val	Plan	Templ	Time
8	5.6	3.1	0.8	1.5	7.89
15	5.7	3.1	1.0	1.4	8.02
22	5.6	3.0	1.3	1.3	8.45
29	5.5	3.2	1.1	1.3	9.11

Table 5.2: Effect of $\mathcal{I}_e$ on the Queries for Rover domain.

extra predicate did not make the action impossible by adding a mutex to already available pre-conditions. Please note, that the predicates were randomly removed from the lifted domain, and asking a question about any grounded action will localize the human model in the lifted domain.

Table 5.1

shows the evaluation for the different number of unknowns and the time taken (in seconds) to find the questions for the domains. The number of questions generated and the time are averaged over 10 different runs. The decrease in the questions is because some predicates were merged using the templates. Except for one case in Rover (with 8 unknown predicates), where we were able to find two different merging templates (thus total questions became 6), we usually had roughly 1 question decrease in the problems. The average number of queries are presented in the table for each domain. The algorithm needs to solve multiple planning problems, but due to PIP query generation can be executed offline. All the queries were first constructed and then validated with the human model (correct IPC domain). The table shows that roughly half of the queries were through validation that was higher than our expectations.

Table 5.2

shows varying the initial state condition on question generation. We constructed new initial states by adding objects to the environment. We randomly removed three and added three different predicates in the lifted domain and generated questions using different initial states. The time and number of questions were averaged over ten different random selections. We expected to observe an effect on time due to extra objects for the time taken to solve multiple planning problems. But, since queries were constructed using Graphplan Planning graph, we did not see any change in time, just a very small increase. It shows that the size of the initial state does not affect the queries, whereas the causal structure of the domain does.

5.8 Related Work

Our work of asking directed questions for model localization, and understanding what every response from the teammate could mean has been motivated by the Intelligent Tutoring System community. But the idea of learning models from data points with specific queries or plan traces has been applied in active learning as well as learning models from plan traces (behavior) for the environment.

Intelligent Tutoring System as a community is working towards maximizing the learning of the students for procedural knowledge. Their central goal is to provide a teacher to every student, and the biggest challenge for them is to understand the model of the student from the work they do and provide feedback or new questions to them. The process of generating questions for students has been used in the past (Zhang and VanLehn, 2016), and they have also used structured knowledge bases to generate more meaningful questions for concepts like photosynthesis (Zhang and VanLehn, 2017). The modeling scheme in ITS is shallow where they represent the

knowledge of any concept as a hidden variable using HMM (Corbett and Anderson, 1993). Parameters learning using sequential data of student's interaction for HMM (Grover *et al.*, 2018b) or deep neural networks (Piech *et al.*, 2015). Based on the learned models, they have also applied dynamic policies to present questions to students using multi-armed bandits (Clement *et al.*, 2014). Our work differs from the ITS community as we are learning a detailed human model for collaboration. We can see this as the first step towards having informative interaction with the user to improve collaboration with them.

Learning planning model using traces. There has been some work to learn the planning models using the behavior in the environment using state predicate differences (Gil, 1994; Stern and Juba, 2017), weighted max-sat (Yang *et al.*, 2007) and finite state machines (Cresswell *et al.*, 2009; Cresswell and Gregory, 2011). Author's (Zhuo *et al.*, 2020) used deep neural networks to learn shallow machine learning model and use it to predict behavior on the test set. There have been other structured formulations such as Linear Temporal Logic (LTL) to represent the knowledge and learning the model using behavior trajectories and an oracle to validate the model (Camacho and McIlraith, 2019). In (Bryce *et al.*, 2016), authors use a questioning strategy to decrease the number of particles to find how the model has changed from the original behavior. The response to the query is in the form of – model provided by the user, labeled valid plan, or some specific predicate that is part of the model now. Recently, there has been some work to infer the model by asking specific queries in the form of the initial state and the plan (Verma *et al.*, 2020). Their work differs as they expect detailed responses, such as which step the plan can be executed in the robot's model, which can easily overwhelm the human teammate (due to interrogative nature).

Active Learning has an oracle to question classes of specific data points (Settles, 2009). The community learns the underlying model with the help of an all-knowing oracle. The difference with the field is that the data of plan traces is not readily available to the agent, such as in scenarios of stream-based selective sampling (Cohn, 1994). Stream-based active learning ideas (Dagan and Engelson, 1995) are useful for the continuous space of probability distribution but can't be used directly in discrete space of questions, where we instead have the generative model for the traces.

As we can see, the idea of questioning the user (or an oracle) for relevant information is not new. In this paper, we have looked at how it is useful for human-robot teaming. The essential part is to understand how to formally define interaction in the form of question and answer and generate useful queries to localize the model, and the application to the novel area comes with its different challenges.

Curriculum Generation idea is to learn the model by dividing into sub-tasks. Generating questions or sequence of questions for the user to answer can be seen as curriculum generation which can be useful in performing transfer learning. It has been applied to supervised learning (Bengio *et al.*, 2009) or for reinforcement learning (Florensa *et al.*, 2017). Reinforcement learning is to learn the task model by exploring and executing actions in the environment. Curriculum learning has been specifically applied for learning reward system of an MDP (Kearns *et al.*, 2002) or use these curriculum tasks to evaluate transfer learning among different tasks (Svetlik *et al.*, 2017). Where curriculum generation handles the task in a stochastic environment, the stochasticity in our case is due to interaction with human. Learning their model can be useful for similar collaboration (or can say transfer learning) for human-robot teams.

All these areas have looked at specific parts of the problem, like active learning community looks to chose the data points to ask the oracle, curriculum generation works uses sequential learning and ITS community works towards generating better questions and what are the psychological effects of these questions. We want to bring these ideas together where we use sequential feedback from the teammate over the generated questions to learn their model of the environment.

5.9 Conclusion and Future Work

Through this paper, we have shown how to construct queries with uncertainty in the model. The robot expects simpler answers such as – (1) yes/no response, and (2) if possible, a complete plan to decrease the number of interactions. It also evaluates different conditions under which each query will have a response that can refine the set of models. The construction using PIP ensures that every question can be asked in any order and these queries can be constructed offline (provided it knows the set of possible models). The evaluations show that these questions can be constructed for any unknown predicate in the model.

In the future, we want to analyze a set of plans rather than the optimal plan. It becomes a two-stepped process, a base framework which involves creating questions, and getting information from the response of the user even if that response does not involve the action a_p but still reaches the goal. The agent has to optimize using the value of information to generate queries, instead of assuming an optimal response from the teammate. This analysis would be useful in general decision-making scenarios and can lead to an open-ended discussion with automated agents.

Chapter 6

MAY I ASK A QUESTION – USER STUDY

Chapter 5 describes how robot can ask questions about unknown parts of the model. We modeled the robot's knowledge using robust planning models (Nguyen *et al.*, 2017) to model the robot's uncertainty about specific parts of the domain. We were able to construct queries about the behavior, that can help robot refine the human's model or the differences between robot and the human teammate. The queries ensured that any response from the teammate will help refine the robot's uncertainty, especially a "No" response can refine the set of models to handle assignment of blame (Crant and Bateman, 1993) scenario. Different types of queries such as validation, construction and template, were explored with the conditions when robot can ask them.

We performed an ablation study to evaluate different aspects of query generation process (especially time taken to generate such queries) and understand how effective the queries can be for different predicates (or parts) of the model. We also evaluated whether a bigger model effects the time taken to generate the query. It was shown through evaluation that validation query and construction queries (with no plans possible) can be constructed for every predicate in the model. On the other hand, template queries could not be constructed for all the predicates, however, under specific conditions, most of them could be merged to decrease the number of questions.

The ablation study provided important factors that effected the query generation system; however, there is a need to evaluate whether these queries can be used for a real-world human-robot teaming scenario. There is a need to construct a user study to simulate a real-world interaction, where the robot or a planning system can ask

queries. It needs to check whether the human teammate understands these questions, and whether the response can support elicitation of true human model. Thus, this chapter, discusses the user study to evaluate the query generation process.

Asking questions is an integral part of many user studies, where a user is introduced to a condition, and then questions are asked to understand the effect of the conditions. It can be tricky to evaluate the effectiveness of a query generation process as we can not ask them if they find the generated questions effective. However, as we introduce the environment to the students, we assume that students have a common vocabulary as the robot and ask them the rules applicable in solving the problem. Through their responses, we will be able to match whether the students understand the environment and compare the application of these rules through the queries. Please note, having a similar vocabulary might not be feasible in the real-world scenarios (Kambhampati *et al.*, 2021), and thus, the agent might not be able to ask the rules directly from the user. Moreover, it has been shown in the knowledge elicitation literature that there is an elicitation gap between verbal reports while asking the questions directly and evaluating the actual performance of the task due to implicit and explicit application of the knowledge (Broadbent *et al.*, 1986). In an active environment, it is essential to evaluate the application of the knowledge instead of merely checking memory recall of the environment dynamics.

Query generation process has made several simplifying assumptions to construct queries about the behavior in the environment. For example, the query generation process assumes that human is an active teammate (could be an expert in the environment), and human's understanding belongs to a set of models, where one of them is the true model. Ideally, we would like to evaluate the system with human users who are new to the working environment, such as, a new fire marshal, or a student who has some understanding of their class material. Thus, the system can assume the

user's knowledge can be represented through a set of models. However, evaluating with these stakeholders can be difficult because to ensure all the students understand the course material can be difficult for a 45 minute to a one hour user study. The query generation process also assumes the queries are generated for specific parts of the model that is unknown to the robotic agent. While modeling, it has been assumed to be represented through predicates which may or may not be a part of the action. Thus, we need to model an active environment in which both human and robot can work or atleast robot can work as suggested by the human teammate. This chapter provides several details about handling these challenges to conduct a user study.

Scenario for the user study is motivated from a simplified warehouse scenario. The robot called *Squeaky* works in a warehouse which stores medicines in boxes as ordered by robot expert *Sam*. The robot is supposed to follow some security protocols while working in the warehouse, and Sam has an understanding of these rules. The company is trying to add more robots, and to decrease the workload on Sam, it is also planning to make a plan library which can help automate the planning process for the robots. The planner is asking questions which Sam (the student taking the user study) is supposed to answer to help build the plan library.

This chapter is divided into five sections. The first section, describes the warehouse domain and followed by a section that reviews various types of queries presented in earlier chapter and how they will be modelled for asking questions during the user study. Then we describe the interface for the user study and different conditions that students would be participating in. Then we provide details about the procedure followed by the hypothesis and the results.

6.1 Warehouse Domain – Squeaky & Sam Team-up

The scenario for the user study has been setup on a simplified warehouse domain, where Squeaky and Sam work together and the warehouse is used for storing medicine. Due to cold temperatures for storing medicines, Sam can not work in the warehouse and is responsible for providing commands to Squeaky. The company is trying to add more robots and decrease the workload of Sam by creating a planner to automate the process of giving commands to old and new robots. We use this story to create a mental picture for the user. In this section we describe important parts of the warehouse environment that are used for the study.

The warehouse consists of racks where medicines are stored, and boxes to store the medicines. For simplification it is assumed that the racks have only one level and look like a table. There are two tables Table 1 and Table 2. Boxes are of two types, large box containing 10 bottles of medicines, and small box containing 2 boxes of medicine. Figure 6.1 shows the image of a simplified warehouse with the robot Squeaky.

Squeaky has certain capabilities for working in the warehouse. It can extend it's arm and torso to reach far off places, it can navigate in the warehouse to go to different racks (or tables), and it can pick up boxes and put them down in the racks or can manipulate stacks of boxes one over the other. There are some security protocols that in reality is understood by Sam, and he ensures that Squeaky should perform actions that do not cause accidents in the warehouse. The security rules as explained in the user study are –

- While moving my torso should be in the contracted position, otherwise, I can topple on the small wheels while turning.

Figure 6.1: Simplified warehouse domain for explaining various capabilities of the robot Squeaky.

- While moving my arm should be tucked in, as extended arm is heavy and I can topple.
- While picking up I can pick up both kind of boxes. Although, I can only pick one box at a time.
- A large box can be stacked *only* on a large box, whereas a small box can be stacked on *both* large and small boxes.

- I can pick up a large box *only* when my torso is in extended state, as I can hold the block vertically.

- I can tuck my arm while holding a small box and not while holding a large box. Thus, I can move while holding small box.

For the user study the students are asked question for their understanding of these security rules. The questions are based on the theory as described in chapter 5 create questions or validate plans based on different types of queries constructed using model elicitation algorithms. In the next section we will review different types of queries to explain there differences, and how they will be applied for the user study.

6.2 Query Generation Process – Review

Chapter 5 describes three different types of queries that the agent can use to refine a user model. This section provides a small summary of these queries and how do they differ. The summary of these queries is discussed in table 6.1. The queries are –

- Validation query – initial and the goal state with the plan. The user has to respond to whether the specific plan would be valid in their model.

- Planning query – initial and goal state and user provides a valid plan in their model.

- Template query – initial and goal state where the user's plan provides a response for multiple rules . Through this query, the agent can decrease the number of interactions.

Table 6.1 provides a summary of three different queries. The validation query asks the user whether a specific plan can be solved in their model, and the agent can refine the model based on "yes/no" from the user. The query is limited to refining for a

Query Type	Question Asked	Response assumed	No. of Rules	Discussion
Validation query	Initial State, Goal State, Plan – *Is the plan valid in your model*?	Yes / No	$= 1$	$*$ Binary responses can provide information for a single rule. $*$ To get information from the negative response, the user is asked about the single rule. Thus the failure means the possible rule is part of the user model.
Planning query	Initial State, Goal State – *Construct a valid plan such that Squeaky can go from initial to the goal state*?	Possible plan in user's model Can include no possible plan as a response.	$= 1$	$*$ In this case the user constructs the plan, with the possibility that there might not be a feasible plan. $*$ Since the user can respond with failure scenario, thus, robot can infer information only about a single rule.

Template query	Initial State, Goal State – *Construct a valid plan such that Squeaky can go from initial to the goal state*?	Possible plan in user's model	> 1	$*$ It is feasible to get information about more than one rule, when a plan exists in every subset of models. $*$ In chapter 4, we discuss the theoretical conditions when a template query can be constructed.

Table 6.1: Summary about different types of queries. Validation and planning query provide information about one rule, even though the user needs to solve the problem. Template query is useful to derive information about more than one rule. No. of rules are the rules for which the query is generated by the robot.

single rule for the user. The planning query asks the user to construct a plan, where constructing the plan can be more information. However, users can fail to construct a plan in their model; thus, they can provide information about the single rule in their model. We believe the planning query will ensure active participation compared to the validation query, and the students will choose the rules applicable in the problems with higher accuracy. Finally, the template query also asks the user to construct a plan for the given problem. However, a failure response is not possible as there is a possible plan in every possible model of the set, and thus the agent can refine multiple rules in a single query. We theoretically describe the conditions under which template queries are possible in section 5.4. Please note, based on the responses from the user,

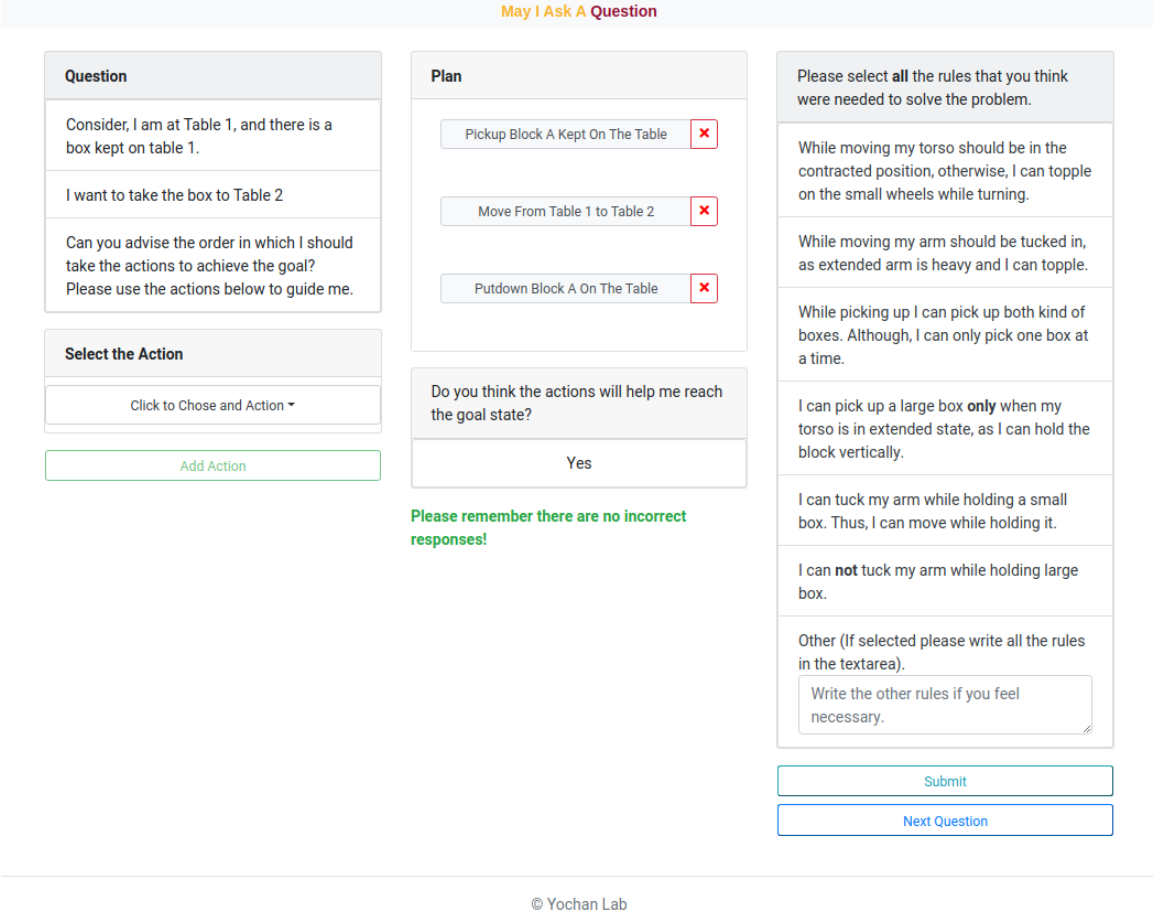


Figure 6.2: Illustration of the `May I Ask a Question` interface for constructing the plan, used for planning and template queries.

the queries can be classified into two broad categories, binary response (yes/no) for the validation query, and a plan for planning and template queries. Thus, we created two different interfaces for both categories which will be discussed in the next section.

6.3 `May I Ask a Question` – User Interface

Based on the responses provided from the student the queries can be classified into two main categories – validating the plan/behavior suggested by the robot (yes/no response), and constructing the plan for the question asked for planning and template queries (a valid plan is the response). Thus we constructed two interfaces, one where

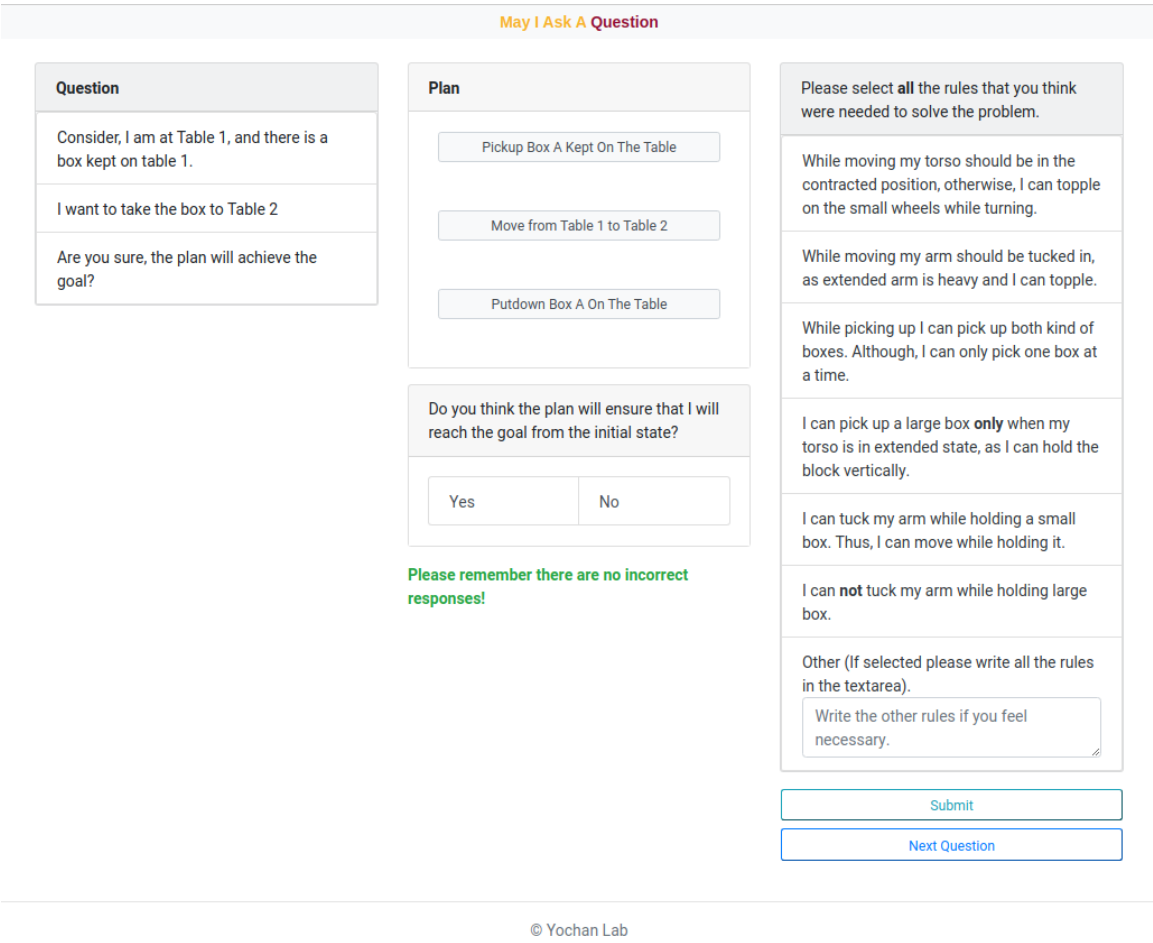


Figure 6.3: Illustration of `May I Ask a Question` interface for validation query

the user constructs a plan for the robot to follow, and an other where user validates the plan presented to them for a specific initial and goal state. Figure 6.2, shows the interface for constructing a plan, and figure 6.3 shows the interface for validating a plan. Now we will describe both the interfaces in some detail, by first explaining the common parts of the interfaces, followed by the differences between them, i.e. extra interface module for constructing a plan for the query.

Validation interface. Figure 6.3 shows the interface used for queries where plan is shown to the user with the initial and the goal state. The interface consists of three panels, left panel shows the question to the user and thus it is called the query

panel. The plan is shown in the middle panel and is called the plan panel. Right panel shows the rules that the student used for validating the plan and is called the rules panel. A student is supposed to read the query (initial and the goal state) from the query panel, understand the plan from the plan panel, select whether they plan is valid or not from the module below the plan and check the rules that they think were applied to solve the problem from the rules panel.

Construction interface. Figure 6.2 shows the interface used for the queries where the plan will constructed by the student. The interface has a similar three panel structure, however the difference lies in the query and the plan panel. In the query panel, below the query module, there is a module to construct actions for adding in a plan. The plan constructed by the user is shown in the plan panel. They need to select whether they have completed the plan by toggling the button below the plan panel. The same flow is followed as the validation interface, where the student reads the query from the query panel, constructs the plan using plan construction module below query module, toggle the button to alert the interface they have completed the plan construction step in the plan panel, select the rules that were used to create the plan and submit their response.

The study requires the person to be an expert in the warehouse domain and act like Sam to provide responses. Thus, we describe the procedure in some detail in the next section. The complete interface details can be checked in the appendix section. It also provides all the forms that were provided to the students.

6.4 User Study – Sam Comes to Help

The user study has been constructed to evaluate the usefulness and effectiveness of directed questions toward model elicitation of the user. To this end, `May I Ask`

`a Question` interface is constructed to first explain the scenario to students and describe the security rules to them, conduct a pre-test to help them commit these rules to memory, then ask them different types of directed queries, and at the end take their feedback. In this section we provide details about different conditions for conducting the user study, our hypothesis over those conditions and the procedure that we evaluated while conducting the user study.

6.4.1 Conditions

As mentioned earlier there are three different queries that Squeaky can ask from Sam, i.e. validation, planning and template queries. Based on the responses from the user these queries can be classified into two main categories where the user either provides a binary response to Squeaky, or constructs a plan to give a response. However, our conditions are based on different types of queries –

(C_v) student validates a plan provided based on the validation query.

(C_p) student constructs a plan for planning queries.

(C_t) student constructs a plan for template queries, thus providing a response for more than one rules at a time.

Please note, that there is no control condition, due to three reasons. First, we are *not* trying to evaluate the effectiveness of these queries as compared to a control case. Secondly, these queries do not evaluate the memorization of the security rules, instead they evaluate the application of the security rules by discussing the behavior. Thus, we ensure that the students have memorized the rules to a certain degree with appropriate feedback at the start of the study. Finally, we are assuming that failure to apply these rules means that the rule is absent from the student's understanding

of the warehouse. Thus, on one hand, these models can be considered brittle and on the other hand, in the real world scenario such (brittle) models exist. For example, several experts forget to check specific conditions which can lead to accidents and one can always argue that the accident was due to lack of knowledge or application or it was a slip (Bayesian knowledge tracing models (Corbett and Anderson, 1993)).

Work in chapter 5 formulated and analyzed whether a structured robot's model can be used to construct questions about the behavior in the common working environment, to elicit specific parts of the human understanding. Moreover, through this study we are evaluating whether these queries are understood by the user and despite the theoretical assumptions can be effective in eliciting the underlying human's model.

6.4.2 Aim of the Study

This study evaluates the utility and effectiveness of the queries to elicit the student's model from a set of models. Effectiveness can be measured using the number of interactions to refine the models, clarity of the questions, and difficulty in response. Theoretically, the number of interactions depends on the number of uncertain predicates and the kind of expected response from the user. However, difficulty and clarity of questions has to be evaluated through the user study. For example, in theory a validation query should be easier to respond as compared to plan construction query, as the student has to provide a binary response for a validation query. On the other hand, for a validation query, giving a plan with the initial state and goal state can confuse them as the plan might be inexplicable to them (Kulkarni *et al.*, 2016), if students are confused about other parts of the model, or if they do not participate actively. Another example of different impacts of these queries could be, the template queries decreases the numbers of interaction (under certain conditions); however, the

queries can be more difficult as the student will have to construct larger plans. Thus, there are several challenges to measuring effectiveness of these queries.

ICAP framework (Chi and Wylie, 2014) discusses the effect of student participation in their learning. It states that there are four modes in which a student can work on some material – interactive, constructive, active and passive. It also states that interactive methods where student discusses the material with tutor or other students is more effective in student learning than, constructive answering questions on the material, than actively reading the material, than passively listening to a lecture. Although our queries do not strictly follow these modes, however, a case can be made that validation query is actively understanding the plan and planning or template query is constructively working on the queries. Thus, the student would be more involved and hence the clarity of planning and template queries could be higher. However, there is a need to evaluate difficulty and clarity of these queries through the user study and these measures will be used for evaluating the effectiveness. We hypothesize that –

H_1 – Difficulty of the query can be measured using –

- H_{1a} – the likert scale from the feedback will follow the order – $D(C_v) < D(C_p) < D(C_t)$, where D represents the difficulty rating.
- H_{1b} – time taken as a representation of the difficulty measure will follow the order – $T(C_v) < T(C_p) < T(C_t)$, where T represents the average time taken to solve the query.
- H_{1c} – Due to the ease of solving queries, preference of the users will follow the order – $\rho(C_v) > \rho(C_p) > \rho(C_t)$, where ρ represents the preference measured through feedback on the likert scale.

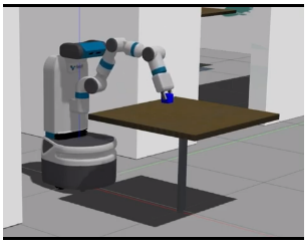


Figure 6.4: Video showing the pick up capability of the robot. We use a combination of text and video to create a mental picture for the student, i.e. to make them expert for handling Squeaky robot around the warehouse.

H_2 – Clarity of the questions will be measured by the choice of correct rules for every question. We expect template and planning query to have improved clarity as it would involve students constructively (ICAP framework (Chi and Wylie, 2014)) working as compared to the validation query. Thus, the order for clarity should be – $\kappa(C_v) < \kappa(C_t) < \kappa(C_p)$, where κ represents the number of questions students choose the correct rules and accurately respond to the queries. Please note, that the clarity measure for planning query is assumed to be better than the template queries.

These hypothesis are directed to the core aim of this user study, i.e. to understand the sweet spot between a user's preference for queries and there effectiveness.

6.4.3 Procedure

The study consists of six different web-based pages, that explains the warehouse domain to the student, takes a pre-test to help students memorize the rules, introduces the interface and then conduct the user study. Finally, the students are provide a form to fill there feedback with a post test to write the security rules in their own words. In this section we will discuss each page in some detail. The material provided to the students while conducting the user study can be checked in Appendix B.

I live at a Warehouse, which consists of serveral objects. To understand the safety rules, I will describe the objects in the warehouse and then describe the rules --

- Warehouse consists of -- racks and boxes.
- Racks where medicines are kept.
- For simplicity, assume that each rack has only one level.
- Boxes of medicine are of two types -- (1) **large** -- containing 10 bottles, (2) **small** -- containing 2 bottles
- I am responsible for making stacks of boxes of medicines and move the boxes around for storage purposes.

Security rules that I have to follow --

- While moving my torso should be in the contracted position, otherwise, I can topple on the small wheels while turning.
- While moving my arm should be tucked in, as extended arm is heavy and I can topple.
- While picking up I can pick up both kind of boxes. Although, I can only pick one box at a time.
- A large box can be stacked **only** on a large box, whereas a small box can be stacked on **both** large and small boxes.

Figure 6.5: Describes the warehouse domain to the student followed by the chosen security rules.

User study introduction consists of discussing about the robot Squeaky working in a warehouse and Sam the robot expert who orders Squeaky to complete the task. It discusses the intent of adding more robots and automating the process of providing commands to Squeaky.

Robot capabilities includes introduction about the robot Squeaky, and its capabilities. The capabilities are shown as small clips of the actions that Squeaky can perform around the warehouse. Squeaky's capabilities are it can pick boxes, move around in the warehouse, stack boxes over each other, extend it's arm and torso to give it more reach. Then the page describes the warehouse, a simplified version of the warehouse with tables and blocks is described. The simplified version consists of one level racks which can be understood as a table, and two different kinds of boxes (large and small). Figure 6.4 shows the screenshot of a video and the text to describe pickup capability of the robot.

Finally, the page introduces the security rules that a student has to understand. The rules can be checked in the warehouse domain section. Six rules are divided into three sets of two rules, where one set is described to every student and one set is randomly chosen from the other two sets of rules. Thus, every student is

Figure 6.6: Shows the pre-test presented to the students. It consists of four correct and four distractor rules. The correct rules are rephrased from the original rules shown to the students.

only shown four out of six rules. For example, the students are shown two rules for motion of the robot, and only one set of the rule either about the robot stacking different boxes (how boxes can be stacked on each other, rules 3 & 4 in the list) or extended capabilities of Squeaky to move with the block in it's hand (rules 5 & 6 in the list). Different sets of rules creates different understanding among the students, however, their understanding can be represented as the set of models. This also tests the validity of our assumption that questions about the behaviors can elicit varying models of the students as well. Figure 6.5 shows the description of warehouse domain and the set of rules shown to the students. Link to videos and the complete text description can be checked in Appendix B.

Pre test. Since, students needs to be an expert on the security rules, we use a pre-test to help them memorize the rules. Pre-test shows a list of eight rules from which the students select the correct rules. The rules are rephrased to test memorization, and distractor rules are added to the list. For example if the rule is Squeaky can pick **only** one box at a time, the distractor rules are Squeaky can pick **many** boxes at a time. Every time a student selects the rules, they are provided feedback whether

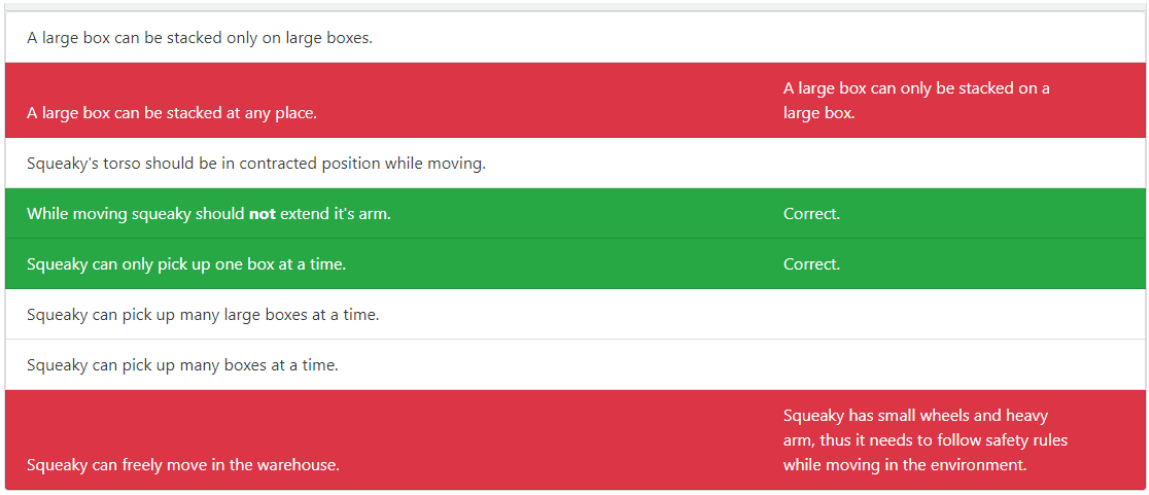


Figure 6.7: Shows the feedback presented to the students while taking the pre-test. Green color represents correctly chosen rule and Red color represents the incorrect rule. With every incorrect selection feedback with the correct rule is presented.

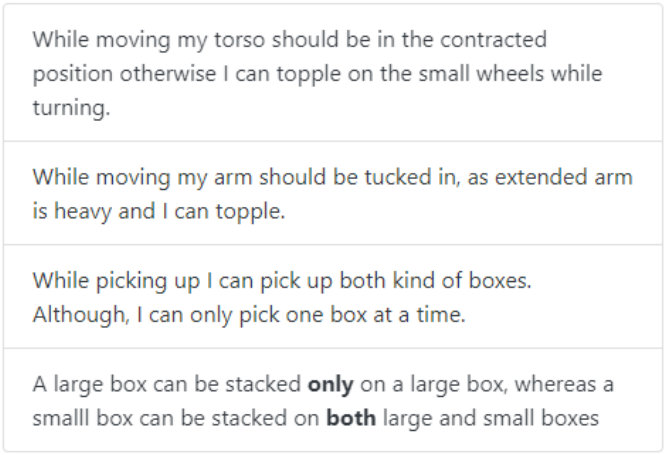


Figure 6.8: Every student takes 3 iterations of pre-test, until they select all four correct rules. After every iteration the set of correct rules originally shown to the student are presented for them to read. Figure shows the set of correct rules shown to them.

there selection is correct of not, and for feedback they are provided correct rule. A student takes the pre-test thrice, and after every attempt a list of correct rules is shown to them. Figure 6.6 shows one of the pre-test provided to the students, figure 6.7 shows feedback provided to the students, and figure 6.8 shows the list of correct rules for feedback after each attempt. Please note, the correct rules are not rephrased, and for each test they have to show memorization and understanding again. After the final (third) attempt correct rephrased rules are shown to the students.

Interface introduction. After the pre-test, the interface shows a video to introduce the study interface. The video consists of explaining all three panels for the respective condition, i.e. validation interface or the construction interface (for planning and template queries). We also show video from the Gazebo robot simulator [1] where Squeaky is executing a plan constructed or shown on the interface. The video helps student visualize an interaction between Sam and Squeaky, where Sam gives it a plan and Squeaky executes in in the warehouse environment. Thus, making it easier for the students to create sequential plans. For the construction interface, the plan construction idea is also described in this video. Finally, video desccribes what a student has to do complete the questions of the user study.

User study. Different conditions have varying questions and response methods. For validation query, six questions are asked from the students where they provide binary (yes/no) response, whether a plan will work in their model. Similary, for planning query, students construct six different plans, and for template query they construct four different plans. The query generation algorithm merged questions for four predicates into two different queries. The other two predicates do not have

[1] https://docs.fetchrobotics.com/gazebo.html

possible plans in the constrained model ($\mathcal{M}_{con}$) and thus there is a possibility of failure response, and the predicates can't be merged. Please note, that even if the students understand only four out of the six rules, they solve the queries for all the security rules, as this will evaluate that despite the user lacking the knowledge about two rules, our methods can elicit their understanding.

Feedback. Finally student answers some feedback through subjective questions about their understanding of the user study, and objective questions (on five points likert scale) to understand how difficult were the questions for them. All the questions are compulsory and submitting the feedback completes the user study.

6.5 Experimental Results

The study was conducted online, where the students were recruited through flyers on various slack channels and through emails to students of Computer Science Department, through different friend. Forty people from several different schools at Arizona State University filled the google form, such as computer science, law, business and education. They were asked for their availability over a span of three days. Thirty-two students participated in the user study with ten–eleven students participating in each condition. These student were compensated $15 for their time. We couldn't use the data from two students as due to network issues all the query responses were not received from one of the student. Another student completed the study in 6 minutes with all the queries response as "No plans possible" and single line feedback. Thus, we present the result from other thirty students (ten in each condition).

Out of these thirty students, twenty-six were from computer science department. fourteen of these thirty students had worked with the robots in the past. Six students were undergraduate students, sixteen were current Masters students at ASU, three

were doctoral students and four had already graduated with Masters degree. One of them chose to not disclose their affiliation or current degree. The students were randomly put into different conditions as the website directly chose the subset of rules described to the students as well as the query to ask them questions. As explained earlier, there were two subsets of rules that could be shown two students (four out of six possible security rules). Out of thirty students the rules were evenly distributed across the user study (fifteen with subset one and fifteen with subset two). However, C_p had six–four distribution over the two subsets, and correspondingly C_t had four–six distribution across the two subsets. Overall students spent 35.27 ± 11.24 minutes doing the user study. Now we will look at some of the detailed results.

Hypothesis H_{1a} In order to measure the difficulty to understand a question we asked a subjective question – `The question above were difficult to understand`. Figure 6.9 shows the response on Likert scale (1–5) where 1 is strongly disagree and 5 is strongly agree. The average score in all the conditions is around 2 (disagree) as shown in the figure. Thus, hypothesis H_{1a} does not hold. Overall score on the likert scale shows that the questions were not difficult to understand for the students in all the conditions.

Hypothesis H_{1b}**.** In this hypothesis we compare the difficulty of each query, as defined by the time taken to solve the problems. Figure 6.10 shows the average time spent by students for the study for all three conditions. Table 6.2 shows the p value statistic for T-test between all three variables. From the figure and the table, it is clear that, on average a student took same time to complete the study (no statistically significant relationship), the per query average is calculated as total time taken / number of queries. As stated in the previous section, that student were

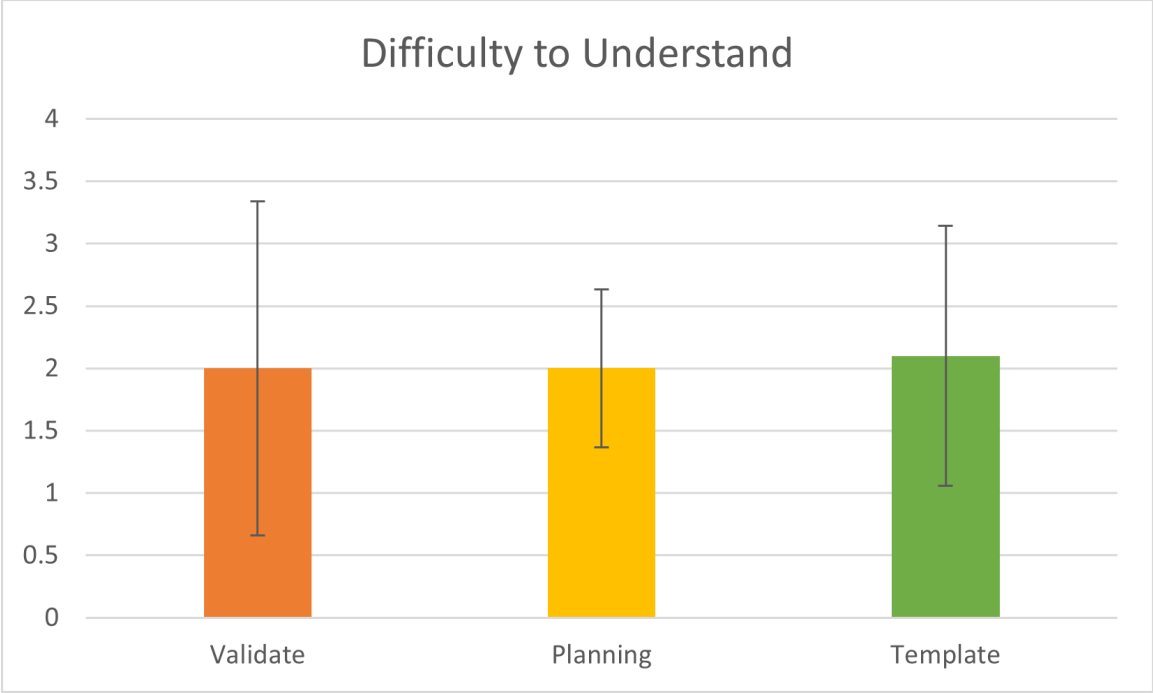


Figure 6.9: Bar graph showing the average Likert Score for the question – `The questions above were difficult to understand`. There was no statistically significant difference for any of the conditions.

asked question about the all the possible predicates, despite lack of knowledge about the specific rules. As we can see, that there is a significant time difference between $T(C_v) < T(C_t)$ and $T(C_p) < T(C_t)$, nothing can be said about the other relationships. Thus, the hypothesis H_{1b} holds partially, and the time taken by template is strictly higher compared to both validation and planning condition. It also means that the template problem as they take more time, can be considered difficult compared to the construction and planning query.

Please note, that since the total time taken to get the information about the queries is statistically insignificant, even if template query gets more information, there is no decrease in time for asking them solve extra information. This can be useful, as template query would be more difficult to solve (provide longer plans),

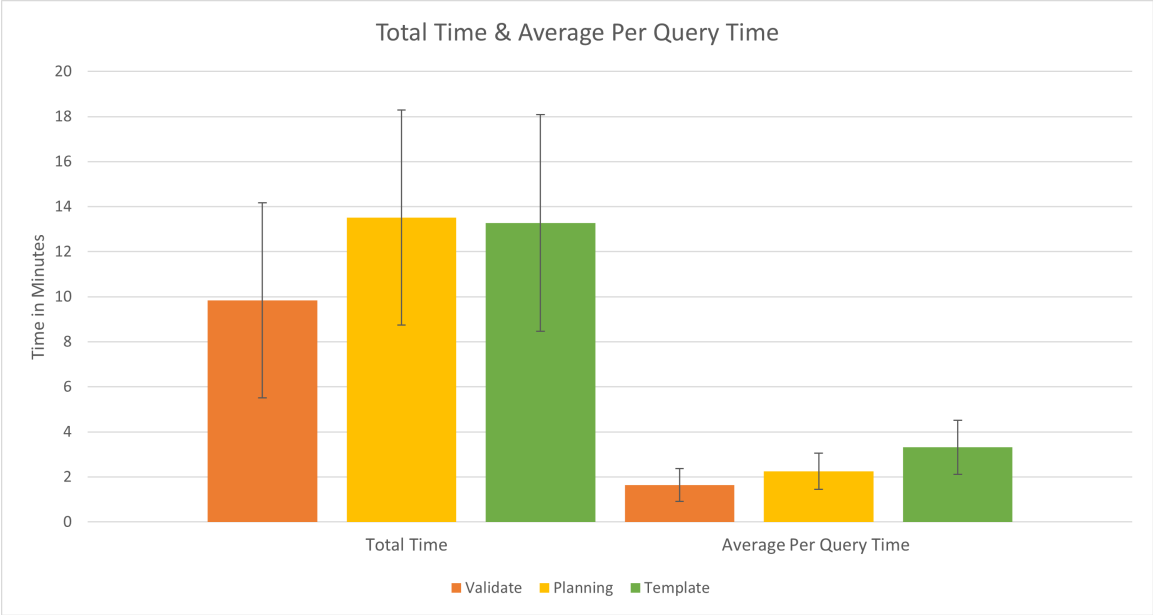


Figure 6.10: Two graphs showing the time taken to solve a query posed by Squeaky in all three conditions. The total time for the user study is statistically significant for $T(C_v) < T(C_p)$ and $T(C_v) < T(C_t)$, however per query time is statistically significant for all three conditions. Average per query time is calculated as Total time / Number of Queries. Validation and Planning query had six queries and Template query has four queries.

	Total time			Average Per Query Time		
	Val	Plan	Temp	Val	Plan	Temp
Val						
Plan	0.052			0.052		
Temp	0.064	0.918		0.0014	0.021	

Table 6.2: T-test value matrix for each query comparison. The α value 0.05, and with Bonferroni correction it is $0.05/3 = 0.0167$.

however, there is no real gain in asking these difficult questions from the human teammate.

Hypothesis H_{1c} tests the ease of solving of these queries. We test it using the Likert score for the question `The questions above were pretty easy to solve for you`. Figure 6.12 shows the average score on the Likert scale (1–5) in each condition. The t-test results were not statistically significant, however, the value for C_t was on average lower than both C_t and C_p.

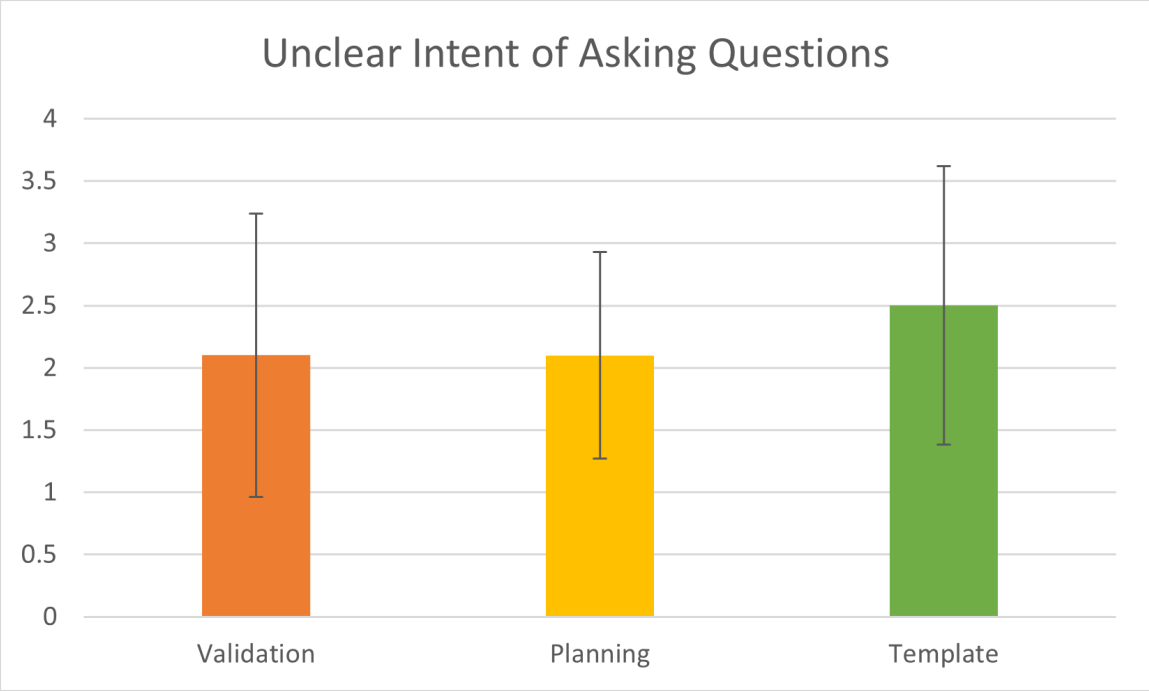


Figure 6.11: Bar graph showing the average Likert Score for the question – `Intent of asking the questions is not clear for the planner.`. There was no statistically significant difference for any of the conditions.

Hypothesis H_2 evaluates the clarity of the questions. The clarity is also evaluated from two different methods. Figure 6.11 shows the average likert scale (1–5) score to the question `Intent of asking the questions is not clear for the planner`

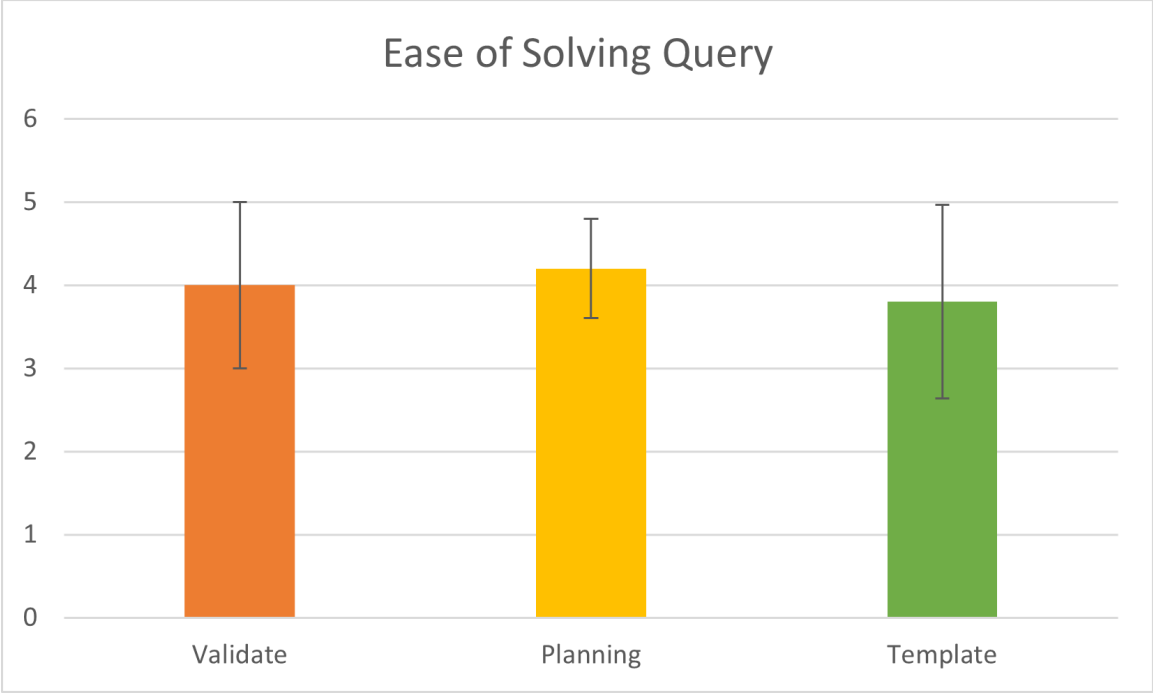


Figure 6.12: Bar graph showing the average Likert Score for the question – `The questions above were pretty easy to solve`. There was no statistically significant difference for any of the conditions.

in all three conditions. While comparing the t-test results aren't statistically significant; however the overall score to the question is 2 (as shown in the figure). Thus, students believe that the questions were clear and the rules to be applied for the questions were easy to find. On the other hand, when comparing the values in each condition, even though the values aren't statistically significant the average value for conditions C_v and C_p is comparable, and the value for C_t is slightly higher.

6.6 Discussion

(a) Applicability for model learning Condition C_v and C_p had 6 questions for every student, and C_t had 4 questions where queries for R_1 and R_2 were merged, and query for R_3 and R_5 were merged. In all the queries students provided the correct response around 80% of the time (48/60 in C_v, 47/60 for C_p, and 30/40 in C_t). Correct

Hypothesis	Outcome	Comments
H_{1a}	≈	Students perceived all the queries to be similarly easy, with overall score of Disagree on the Likert Scale.
H_{1b}	✓	Average time taken per query was statistically significant with template query taking highest time.
H_{1c}	≈	Students felt that the queries were easy to solve, Template Query (C_t) had the lowest rating for ease of solving.
H_2	✓/ ?	The Likert scores were highest for C_t, overall students felt the queries were understandable to them.

Table 6.3: Summary of results for the `May I Ask a Question` user study.

rules were chosen for 72% of the responses (43/60 in C_v, 42/60 for C_p, and 29/40 in C_t). Correct query response is defined as the valid response given the question and set of rules shown to the student. Similarly, correct rules were those when based on the set of rules chosen, they provide the correct reasoning irrespective of the correctness of the response. This, validates that students understood these queries and were able to respond with some accuracy. Overall our results show, average time taken to solve a single query increased for the template condition, however, total time taken to complete the user study did not, i.e. answering queries for all six rules, took similar amount of time. Thus, planning query maintains the right balance between ease and getting the right information from the student.

(b) Post-hoc analysis As discussed earlier, the hypothesis of the user study were constructed to showcase the application of question answering for learning the human mental model. Some of the ideas from human-cognition and from the area of psy-

cholinguistics, such as discussing behavior to understand application of rules instead of memorization and getting information from negative responses (such as "No" in validation query or "no plan possible" for plan generation query) were used to generate these queries. Table 6.3 describes the summary of results for every hypothesis. Although, the hypothesis for ease of solving and understanding only hold partially, however, it supports the central theme of the user study that the all kinds of queries can be applied for working with a teammate in the real world scenario. The difference between these queries is that for a single query usually it's better to ask the possible plan from the user for the initial state, however, for sequence of queries the total time taken is similar. This helps us conclude that asking plan generation queries or plan validation queries is useful for lesser number of rules. However, with higher number of uncertain rules, it is better to decrease the number of interactions using template query, as students felt that the queries were equally difficult or easy to solve and equally intuitive to understand.

(c) Novel user study An important point to note is that asking questions to understand teammate's model is widely used technique. This user study is unique in the ways as it tests core properties of a query, that is whether the teammate can even understand the question that robot is asking. The field of ITS has evaluated many types of questions, and to the best of our knowledge such a user study has never been constructed. In ITS, the core questions are about whether an interface can support specific type of interaction or whether a student finds the interface useful to learn specific material. This user study was constructed to evaluate the effectiveness and intent of asking questions from a teammate.

(d) Plan for Validation query While constructing the plan for validation query, special care was needed, as the plan has to be explicable from the student (Kulkarni *et al.*, 2016). This scenario effected one of the responses as $\mathcal{M}_{con}$ model for a subset of security rules was not correctly represented. For example, one subset of security did not explain that squeaky can only lift a heavy block after extending it's torso. If we assumed the $\mathcal{M}_{con}$ model had this rule, it would have expand torso action in the plan, which can confuse the user. This scenario happened with one of the students, who raised the concern about the extra action in the pilot tests. On the other hand, this condition never occurred in the planning and template queries as the plan was always feasible in the student's model.

6.7 Conclusion

In the previous chapter we were able to theoretically describe how a robot can construct directed question that can elicit a human teammate's model. The important conditions were ensuring agent asks questions such that every kind of response can elicit some part of the model. In this chapter, we were able to show a scenario and evaluate it through a user study. Students were able to provide response and every response from them elicited the applicability of specific parts of the model. Different types of queries and query types were evaluated. The results show that the questions are easy to understand for the students, and also easy to solve.

Average time taken to solve a query increased when for the template condition, however, overall time taken to complete the user study, i.e. answering queries for all six rules, took same amount of time. Thus, planning query maintains the right balance between ease and getting the right information from the student.

While constructing the plan for validation query, special care was needed, as the plan has to be explicable from the student (Kulkarni *et al.*, 2016). This scenario

effected one of the responses as $\mathcal{M}_{con}$ model for a subset of security rules was not correctly represented. For example, one subset of security did not explain that squeaky can only lift a heavy block after extending it's torso. If we assumed the $\mathcal{M}_{con}$ model had this rule, it would have expand torso action in the plan, which can confuse the user. This scenario happened with one of the students, who ended up asking about this extra action which was not needed in one of the pilot tests. On the other hand, this condition never occurred in the planning and template queries as the user was constructed the plan, and the plan was always feasible in their model.

Chapter 7

CONCLUSION

This thesis was motivated towards understanding the current state of the human-aware systems and extend some of the fundamental ideas to different domains. Initially we looked at some of the technology as seen by ITS research community, and applied some of the human-aware AI techniques to support students and teachers in a classroom. Vanlehn (2006) described that, an ITS consisted of two loops – outer and the inner loop. Outer loop runs every problem to choose the next problem for the student and inner loop is applied to every step that a student takes while solving a problem to provide feedback and hints. Through our work we have been able to extended these ideas to other real-world domains, and evaluate the effectiveness of these techniques. We have been able to show that through our work that automated planning and human-aware AI is well suited to be applied for both long term and short term inference, by modeling one of the state-of-the-art tutoring systems. Then we applied the same framework towards supporting an expert and giving them both step-by-step feedback as well as complete plan suggestions to solve the expert's problem. We also evaluated the effectiveness of these techniques through one of the first-of-its-kind user studies for student experts. The results showed both plan validation and plan suggestion could be used for such systems and can provide support to experts in their complex tasks. Applying these techniques to different domains showcased the applicability of these techniques to several real-world domains. In chapter 5 we tried to generalize the outer loop of an ITS, and ask questions from an expert. The underlying model definitions and techniques were motivated from automated task planning literature, with more fundamental human assumptions that an automated

system should keep in mind. The user study showed that the questions constructed using the framework can be applied for asking questions from experts. The empirical evaluation showed the application of questions to several different domains.

From formal techniques such as automated task planning and human-aware AI point of view, this work shows the usefulness of these upcoming frameworks to real-world scenarios. Initially these frameworks were built to solve problems and provide solutions directly to the expert. However, with the improving technology, one of the central challenges is to interact directly with human-in-the-loop and incorporate their thought process. Through this work we have shown that these formal techniques can be applied to interact with the user and can support involved collaboration with active human teammates. Obviously, there are many more challenges before robots or planners can be seen interacting with the humans, however, our work has laid the foundation to take formal methods and apply it to work with humans on real-world applications.

7.1 Future Work

The work presented in this system takes the first steps towards bringing automated systems to work with humans. This work has mainly focused on designing frameworks with domain-independent techniques, and evaluating their usefulness through user studies. This work can be extended to support several different domains such as, Intelligent Tutoring Systems or many other domains to work with experts.

Intelligent Tutoring System community assumes partial observability about student's knowledge about concepts. They also assume that student's understanding of the stochastic and represent it as a chance that a student will correctly apply the concept when provided by a problem. To apply these techniques there is a need to handle stochasticity to accurately represent the student's current knowledge state,

and thus, look for policies and support these actions of the students. While asking questions, the agent can assume some knowledge of the unknown principle, and create a policy robust enough to handle the uncertainty.

Evolving mental models. While collaborating with active teammates, there is a need for the systems to track these evolving models. For example, if a student does not understand some knowledge concept, and correctly applies the feedback provided by the system in the next problem, the tutoring system might need to update $\mathcal{M}_r^H$ model to incorporate users understanding of the current environment. First, there is a need to track these changes and secondly, the agents would also need to understand the impact of these changes. Thus, providing personalized support to the active human teammates.

Incomplete knowledge. AI techniques have close-world assumptions (Talamadupula *et al.*, 2010b). However, these techniques have to be extended to gracefully handle unknown part of the models, such as, the human teammate is able to perform an action that the robot believed was not possible, or had no knowledge about. These properties can be incorporated through learning methods, or the system can ask the active teammate to help them update their model. Please note, incomplete knowledge is different from tracking evolving models, as tracking changes to the teammates model is due to the environment that an agent could understand. Whereas, incomplete knowledge would mean that the system needs to understand something that it has never experienced.

Domain dependent techniques. There have been many domain dependent techniques, that communities use, such as

Agarwal, M., T. Chakraborti, S. Grover and A. Chaudhary, "COVID-19 india dataset: Parsing detailed COVID-19 data in daily health bulletins from states in india", CoRR **abs/2110.02311**, URL https://arxiv.org/abs/2110.02311v2 (2021).

Ai-Chang, M., J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias *et al.*, "Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission", IEEE Intell. Sys. (2004).

Aleven, V., E. A. McLaughlin, R. A. Glenn and K. R. Koedinger, "Instruction based on adaptive learning technologies", Handbook of research on learning and instruction pp. 522–560 (2016).

Allen, J. F., "Mixed initiative planning: Position paper", in "Proceedings of ARPA/Rome Labs Planning Initiative Workshop, Tuscon, AZ", (Morgan Kaufman, Palo Alto, 1994).

Amershi, S., D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen *et al.*, "Guidelines for human-ai interaction", in "Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland", pp. 1–13 (ACM, 2019).

Amy Ahearn, "The Flip Side of Abysmal MOOC Completion Rates? Discovering the Most Tenacious Learners", https://goo.gl/DR7nxa, EdSurge (2017).

Anderson, J., "The architecture of cognition.", Harvard Univ. press, Cambridge, MA (1983).

Anderson, J., "Rules of the mind.", Lawrence Erlbaum Associates Hillsdale, NJ (1993).

Anderson, J. R., "Act: A simple theory of complex cognition.", American psychologist **51**, 4, 355 (1996).

Anderson, J. R. and G. H. Bower, "Human associative memory", Winston and Son, Washington, DC (1973).

Anderson, J. R., C. F. Boyle and B. J. Reiser, "Intelligent tutoring systems", Science **228**, 4698, 456–462 (1985).

Anderson, J. R., A. T. Corbett, K. R. Koedinger and R. Pelletier, "Cognitive tutors: Lessons learned", The journal of the learning sciences **4**, 2, 167–207 (1995).

Anderson, L. W. and D. R. Krathwohl, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives* (Longman,, 2001).

Aronson, E., *The jigsaw classroom: Building cooperation in the classroom* (Scott Foresman & Company, 1997).

Aronson, E., "Building empathy, compassion, and achievement in the jigsaw classroom", in "Improving academic achievement", pp. 209–225 (Elsevier, 2002).

Aronson, E., *Cooperation in the classroom: The jigsaw method* (Printer & Martin Limited, 2011).

Bailey, W. A. and E. J. Kay, "Structural analysis of verbal data", ACM SIGCHI Bulletin **17**, SI, 297–301 (1986).

Baker, R. S., A. De Carvalho, J. Raspat, V. Aleven, A. T. Corbett and K. R. Koedinger, "Educational software features that encourage and discourage "gaming the system"", in "Proceedings of the 14th international conference on artificial intelligence in education", pp. 475–482 (2009).

Barla, M., M. Bieliková, A. B. Ezzeddinne, T. Kramár, M. Šimko and O. Vozár, "On the impact of adaptive test question selection for learning efficiency", Computers & Education **55**, 2, 846–857 (2010).

Barnes, T. and J. Stamper, "Automatic hint generation for logic proof tutoring using historical data", Journal of Educational Technology & Society **13**, 1, 3 (2010).

Beck, J., B. P. Woolf and C. R. Beal, "Advisor: A machine learning architecture for intelligent tutor construction", AAAI/IAAI **2000**, 552-557, 1–2 (2000).

Beck, J. E. and K.-m. Chang, "Identifiability: A fundamental problem of student modeling", in "International Conference on User Modeling", pp. 137–146 (Springer, 2007).

Bengio, Y., J. Louradour, R. Collobert and J. Weston, "Curriculum learning", in "Proceedings of the 26th annual international conference on machine learning", pp. 41–48 (ACM, 2009).

Berezina, K., O. Ciftci and C. Cobanoglu, "Robots, artificial intelligence, and service automation in restaurants", in "Robots, artificial intelligence, and service automation in travel, tourism and hospitality", (Emerald Publishing Limited, 2019).

Bloom, B. S., "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring", Educational researcher **13**, 6, 4–16 (1984).

Bloom, B. S., M. D. Engelhart, E. Furst, W. H. Hill and D. R. Krathwohl, *Taxonomy of educational objectives: The classification of educational goals; Handbook I: Cognitive domain* (New York: David McKay, 1956).

Bloom, B. S., C. of College and U. Examiners, *Taxonomy of educational objectives*, vol. 2 (Longmans, Green New York, 1964).

Bollini, M., S. Tellex, T. Thompson, N. Roy and D. Rus, "Interpreting and executing recipes with a cooking robot", in "Experimental Robotics", pp. 481–495 (Springer, 2013).

Bouchet, F., H. Labarthe, K. Yacef and R. Bachelet, "Comparing peer recommendation strategies in a mooc", in "Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization", pp. 129–134 (ACM, 2017).

Broadbent, D. E., P. FitzGerald and M. H. Broadbent, "Implicit and explicit knowledge in the control of complex systems", British journal of Psychology **77**, 1, 33–50 (1986).

Bryce, D., J. Benton and M. W. Boldt, "Maintaining evolving domain models", in "Proceedings of the twenty-fifth international joint conference on artificial intelligence", pp. 3053–3059 (2016).

Bubeck, S. and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems", arXiv preprint arXiv:1204.5721 (2012).

Burke, A., "Group work: How to use groups effectively.", Journal of Effective Teaching **11**, 2, 87–95 (2011).

Camacho, A. and S. A. McIlraith, "Learning interpretable models expressed in linear temporal logic", in "Proceedings of the International Conference on Automated Planning and Scheduling", vol. 29-1, pp. 621–630 (2019).

Carbonell, J. R., "Ai in cai: An artificial-intelligence approach to computer-assisted instruction", IEEE transactions on man-machine systems **11**, 4, 190–202 (1970).

Chakraborti, T., S. Kambhampati, M. Scheutz and Y. Zhang, "Ai challenges in human-robot cognitive teaming", arXiv preprint arXiv:1707.04775 (2017a).

Chakraborti, T., A. Kulkarni, S. Sreedharan, D. E. Smith and S. Kambhampati, "Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior", in "Proceedings of the International Conference on Automated Planning and Scheduling, Berkley, California, USA", vol. 29, pp. 86–96 (2019a).

Chakraborti, T., S. Sreedharan, S. Grover and S. Kambhampati, "Plan explanations as model reconciliation: an empirical study", in "Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction, Daegu, South Korea", pp. 258–266 (2019b).

Chakraborti, T., S. Sreedharan and S. Kambhampati, "Balancing Explicability and Explanation in Human-Aware Planning", in "AAMAS", (2018a).

Chakraborti, T., S. Sreedharan and S. Kambhampati, "Human-aware planning revisited: A tale of three models", in "IJCAI-ECAI XAI/ICAPS XAIP Workshops", (2018b).

Chakraborti, T., S. Sreedharan, Y. Zhang and S. Kambhampati, "Plan explanations as model reconciliation: moving beyond explanation as soliloquy", in "Proceedings of the 26th International Joint Conference on Artificial Intelligence", pp. 156–163 (AAAI Press, 2017b).

Chakraborti, T., K. Talamadupula, M. Dholakia, B. Srivastava, J. O. Kephart and R. K. Bellamy, "Mr.Jones – Towards a Proactive Smart Room Orchestrator", in "AAAI Fall Symposium on Human-Agent Groups", (2017c).

Chang, K.-m., J. Beck, J. Mostow and A. Corbett, "A bayes net toolkit for student modeling in intelligent tutoring systems", in "International Conference on Intelligent Tutoring Systems", pp. 104–113 (Springer, 2006).

Chi, M., P. Jordan, K. VanLehn and M. Hall, "Reinforcement learningbased feature selection for developing pedagogically effective tutorial dialogue tactics", in "Educational Data Mining 2008", (Citeseer, 2008).

Chi, M., K. VanLehn and D. Litman, "Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics", in "ITS", pp. 224–234 (2010).

Chi, M. T. and R. Wylie, "The icap framework: Linking cognitive engagement to active learning outcomes", Educational psychologist **49**, 4, 219–243 (2014).

Clement, B., D. Roy, P.-Y. Oudeyer and M. Lopes, "Multi-armed bandits for intelligent tutoring systems", arXiv preprint arXiv:1310.3174 (2013).

Clement, B., D. Roy, P.-Y. Oudeyer and M. Lopes, "Online optimization of teaching sequences with multi-armed bandits", in "7th international conference on educational data mining", (2014).

Cohn, D. A., "Neural network exploration using optimal experiment design", in "Advances in neural information processing systems", pp. 679–686 (1994).

Collins, A., J. S. Brown and S. E. Newman, "Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics", Thinking: The journal of philosophy for children **8**, 1, 2–10 (1988).

Compton, P. and R. Jansen, "A philosophical basis for knowledge acquisition", Knowledge acquisition **2**, 3, 241–258 (1990).

Connelly, J. and S. Katz, "Toward more robust learning of physics via reflective dialogue extensions", in "EdMedia: World Conference on Educational Media and Technology", pp. 1946–1951 (Association for the Advancement of Computing in Education (AACE), 2009).

Cooke, N. J., "Knowledge elicitation", Handbook of applied cognition pp. 479–509 (1999).

Corbett, A. T. and J. R. Anderson, "Student modeling in an intelligent programming tutor", in "Cognitive Models and Intelligent Environments for Learning Programming", edited by E. Lemut, B. du Boulay and G. Dettori, pp. 135–144 (Springer Berlin Heidelberg, Berlin, Heidelberg, 1993).

Corbett, A. T. and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge", User modeling and user-adapted interaction **4**, 4, 253–278 (1994).

Crant, J. M. and T. S. Bateman, "Assignment of credit and blame for performance outcomes", Academy of Management Journal **36**, 1, 7–27 (1993).

Cresswell, S. and P. Gregory, "Generalised domain model acquisition from action traces", in "Twenty-First International Conference on Automated Planning and Scheduling", (Citeseer, 2011).

Cresswell, S., T. L. McCluskey and M. M. West, "Acquisition of object-centred domain models from planning examples", in "Nineteenth International Conference on Automated Planning and Scheduling", (AAAI press, 2009).

Cullen, J. and A. Bryman, "The knowledge acquisition bottleneck: time for reassessment?", Expert Systems **5**, 3, 216–225 (1988).

Dagan, I. and S. P. Engelson, "Committee-based sampling for training probabilistic classifiers", in "Machine Learning Proceedings 1995", pp. 150–157 (Elsevier, 1995).

Dede, C., "A review and synthesis of recent research in intelligent computer-assisted instruction", International Journal of Man-Machine Studies **24**, 4, 329–353 (1986).

Dillenbourg, P. and P. Jermann, "Technology for classroom orchestration", in "New science of learning", pp. 525–552 (Springer, 2010).

Dillenbourg, P., G. Zufferey, H. Alavi, P. Jermann, S. Do-Lenh, Q. Bonnard, S. Cuendet and F. Kaplan, "Classroom orchestration: The third circle of usability", CSCL2011 Proceedings **1**, 510–517 (2011).

Doerr, H. M., "Stella ten years later: A review of the literature", International Journal of Computers for Mathematical Learning **1**, 2, 201–224 (1996).

Doignon, J.-P. and J.-C. Falmagne, "Knowledge spaces and learning spaces", arXiv preprint arXiv:1511.06757 (2015).

Doroudi, S. and E. Brunskill, "The misidentified identifiability problem of bayesian knowledge tracing.", International Educational Data Mining Society (2017).

Dragan, A. D., K. C. Lee and S. S. Srinivasa, "Legibility and predictability of robot motion", in "Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction, Tokyo, Japan", pp. 301–308 (IEEE Press, 2013).

Emre, R. K., *Developing a Neural Network Based Adaptive Task Selection System for an Undergraduate Level Organic Chemistry Course.*, Ph.D. thesis, Arizona State University, Tempe, USA (2020).

Ericsson, K. A. and H. A. Simon, *Protocol analysis: Verbal reports as data.* (the MIT Press, 1984).

Fairbairn, D. M., "The art of questioning your students", The Clearing House **61**, 1, 19–22 (1987).

Falmagne, J.-C., E. Cosyn, J.-P. Doignon and N. Thiéry, "The assessment of knowledge, in theory and in practice", in "Formal concept analysis", pp. 61–79 (Springer, 2006).

Feigenbaum, E. A., "What hath simon wrought?", in "Complex Information Processing", pp. 185–202 (Psychology Press, 1989).

Feigh, K. M., A. R. Pritchett, T. W. Denq and J. A. Jacko, "Contextual control modes during an airline rescheduling task", Journal of Cognitive Engineering and Decision Making **1**, 2, 169–185 (2007).

Ferland, L. and S. Sanner, "Academic advising domain, at international planning competition (ipc) probabilistic track", https://ipc2018-probabilistic.bitbucket.io/#domains (2018).

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

Florensa, C., D. Held, M. Wulfmeier, M. Zhang and P. Abbeel, "Reverse curriculum generation for reinforcement learning", arXiv preprint arXiv:1707.05300 (2017).

Fox, M., R. Howey and D. Long, "Validating plans in the context of processes and exogenous events", in "Proceedings of the twentieth aaai conference on artificial intelligence, Pittsburgh, PA", vol. 5, pp. 1151–1156 (AAAI Press, 2005).

Gajos, K. Z., K. Everitt, D. S. Tan, M. Czerwinski and D. S. Weld, "Predictability and accuracy in adaptive user interfaces", in "Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy", pp. 1271–1274 (ACM, 2008).

Gertner, A. S., C. Conati and K. VanLehn, "Procedural help in andes: Generating hints using a bayesian network student model", Aaai/Iaai **1998**, 106–11 (1998).

Gertner, A. S. and K. VanLehn, "Andes: A coached problem solving environment for physics", in "International conference on intelligent tutoring systems", pp. 133–142 (2000).

Ghallab, M., D. Nau and P. Traverso, *Automated Planning: theory and practice* (Elsevier, 2004).

Gil, Y., "Learning by experimentation: Incremental refinement of incomplete planning domains", in "Machine Learning Proceedings 1994", pp. 87–95 (Elsevier, 1994).

Glaser, R., M. Chi and M. Farr, "Overview – the nature of expertise", LEA Hillsdale NJ (1988).

González-Brenes, J., Y. Huang and P. Brusilovsky, "General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge", in "The 7th international conference on educational data mining", pp. 84–91 (University of Pittsburgh, 2014).

Graesser, A. C., P. Chipman, B. C. Haynes and A. Olney, "Autotutor: An intelligent tutoring system with mixed-initiative dialogue", IEEE Transactions on Education (2005).

Graesser, A. C., S. D'Mello, X. Hu, Z. Cai, A. Olney and B. Morgan, "Autotutor", in "Applied natural language processing: Identification, investigation and resolution", pp. 169–187 (IGI Global, 2012).

Grochow, J. M., "A taxonomy of automated assistants", Communications of the ACM **63**, 4, 39–41 (2020).

Grover, S., *Online Embedded Assessment for Dragoon, Intelligent Tutoring System* (Arizona State University, 2015).

Grover, S., K. Arora and S. K. Mitra, "Text extraction from document images using edge information", in "2009 Annual IEEE India Conference", pp. 1–4 (IEEE, 2009).

Grover, S., T. Chakraborti and S. Kambhampati, "What can automated planning do for intelligent tutoring systems", ICAPS SPARK (2018a).

Grover, S., S. Sengupta, T. Chakraborti, A. P. Mishra and S. Kambhampati, "ipass: A case study of the effectiveness of automated planning for decision support", in "Proceedings of 14th International Conference on Naturalistic Decision Making, San Fransisco, CA", (2019).

Grover, S., S. Sengupta, T. Chakraborti, A. P. Mishra and S. Kambhampati, "Radar: automated task planning for proactive decision support", Human–Computer Interaction pp. 1–26 (2020a).

Grover, S., D. Smith and S. Kambhampati, "Model elicitation through direct questioning", ICAPS, XAIP; arXiv preprint arXiv:2011.12262 (2020b).

Grover, S., J. Wetzel and K. VanLehn, "How should knowledge composed of schemas be represented in order to optimize student model accuracy?", in "International Conference on Artificial Intelligence in Education", pp. 127–139 (Springer, 2018b).

Grudin, J., "Ai and hci: Two fields divided by a common focus", Ai Magazine **30**, 4, 48–48 (2009).

Grudin, J., "Human-computer interaction", Annual review of information science and technology **45**, 1, 367–430 (2011).

Hambleton, R. K., H. Swaminathan and H. J. Rogers, *Fundamentals of item response theory*, vol. 2 (Sage, 1991).

Hancock, P. A. and M. H. Chignell, "Mental workload dynamics in adaptive interface design", IEEE transactions on Systems, Man, and Cybernetics **18**, 4, 647–658 (1988).

Harvey, R. J. and A. L. Hammer, "Item response theory", The Counseling Psychologist **27**, 3, 353–383 (1999).

Helmert, M., "The fast downward planning system.", JAIR (2006).

Hochreiter, S. and J. Schmidhuber, "Long short-term memory", Neural computation **9**, 8, 1735–1780 (1997).

Hoffman, R. R., "A survey of methods for eliciting the knowledge of experts", ACM SIGART Bulletin **108**, 19–27 (1989).

Hoffman, R. R., N. R. Shadbolt, A. M. Burton and G. Klein, "Eliciting knowledge from experts: A methodological analysis", Organizational behavior and human decision processes **62**, 2, 129–158 (1995).

Hoffmann, J., J. Porteous and L. Sebastia, "Ordered landmarks in planning", JAIR URL http://dl.acm.org/citation.cfm?id=1622487.1622495 (2004).

Hone, K. S. and G. R. El Said, "Exploring the factors affecting mooc retention: A survey study", Computers & Education **98**, 157–168 (2016).

Horvitz, E., "Principles of mixed-initiative user interfaces", in "Proceedings of the SIGCHI conference on Human Factors in Computing Systems, Pittsburgh, PA", pp. 159–166 (ACM, 1999).

Howey, R., D. Long and M. Fox, "Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl", in "Tools with Artificial Intelligence (ICTAI). 16th IEEE International Conference on", pp. 294–301 (IEEE, 2004).

Iglesias, A., P. Martínez, R. Aler and F. Fernández, "Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning", Applied Intelligence **31**, 1, 89–106 (2009).

Jackson, G. T. and D. McNamara, "Motivational impacts of a game-based intelligent tutoring system", in "Twenty-Fourth International FLAIRS Conference", (2011).

Jackson, G. T. and D. S. McNamara, "Motivation and performance in a game-based intelligent tutoring system.", Journal of Educational Psychology **105**, 4, 1036 (2013).

Kambhampati, S., E. Parker and E. Lambrecht, "Understanding and extending graphplan", in "European Conference on Planning", pp. 260–272 (Springer, 1997).

Kambhampati, S., S. Sreedharan, M. Verma, Y. Zha and L. Guan, "Symbols as a lingua franca for bridging human-ai chasm for explainable and advisable ai systems", arXiv preprint arXiv:2109.09904 (2021).

Kambhampati, S. and K. Talamadupula, "Human-in-the-loop planning and decision support", in "AAAI Tutorial", (2015).

Katz, S., D. Allbritton and J. Connelly, "Going beyond the problem given: How human tutors use post-solution discussions to support transfer", International Journal of Artificial Intelligence in Education **13**, 1, 79–116 (2003).

Katz, S., A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin and L. Greenberg, "Sherlock 2: An intelligent tutoring system built on the lrdc framework (pp. 227-258)", Facilitating the development and use of interactive learning environments. Mahwah, NJ: Erlbaum (1998).

Katz, S., G. O'Donnell and H. Kay, "An approach to analyzing the role and structure of reflective dialogue", International Journal of Artificial Intelligence in Education (2000).

Kearns, M., Y. Mansour and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large markov decision processes", Machine learning **49**, 2-3, 193–208 (2002).

Keyder, E., S. Richter and M. Helmert, "Sound and complete landmarks for and/or graphs.", in "ECAI", vol. 215, pp. 335–340 (2010).

Khajah, M., R. Wing, R. Lindsey and M. Mozer, "Integrating latent-factor and knowledge-tracing models to predict individual differences in learning", in "Educational Data Mining 2014", (Citeseer, 2014a).

Khajah, M. M., Y. Huang, J. P. González-Brenes, M. C. Mozer and P. Brusilovsky, "Integrating knowledge tracing and item response theory: A tale of two frameworks", in "CEUR Workshop proceedings", vol. 1181, pp. 7–15 (University of Pittsburgh, 2014b).

Khan, O. Z., P. Poupart and J. P. Black, "Automatically Generated Explanations for Markov Decision Processes", in "Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions", pp. 144–163 (IGI Global, Hershey, Pennsylvania, USA, 2012).

Kim, J., C. J. Banks and J. A. Shah, "Collaborative planning with encoding of users' high-level strategies", in "Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence", pp. 955–961 (AAAI Press, 2017).

Klein, G., "Naturalistic decision making", The Journal of the Human Factors and Ergonomics Society (2008).

Koedinger, K. R., A. T. Corbett and C. Perfetti, "The knowledge-learning-instruction (kli) framework: Toward bridging the science-practice chasm to enhance robust student learning", Cognitive Science (2010).

Krathwohl, D. R., "A revision of bloom's taxonomy: An overview", Theory into practice **41**, 4, 212–218 (2002).

Kulkarni, A., T. Chakraborti, Y. Zha, S. G. Vadlamudi, Y. Zhang and S. Kambhampati, "Explicable robot planning as minimizing distance from expected behavior", CoRR **abs/1611.05497**, URL http://arxiv.org/abs/1611.05497 (2016).

Labarthe, H., F. Bouchet, R. Bachelet and K. Yacef, "Does a peer recommender foster students' engagement in moocs?", in "9th International Conference on Educational Data Mining", pp. 418–423 (2016).

LaFrance, M., "Excavation, capture, collection, and creation: Computer scientists' metaphors for eliciting human expertise", Metaphor and Symbolic Activity **7**, 3-4, 135–156 (1992).

Langley, P., "User modeling in adaptive interfaces", in "Proceedings of the seventh international conference on User modeling, Banff, Canada", pp. 357–370 (Springer-Verlag New York, Inc., 1999).

Laskey, K. B., H. C. Marques and P. C. da Costa, "High-level fusion for crisis response planning", in "Fusion Methodologies in Crisis Management", pp. 257–285 (Springer, 2016).

Lee, C. D., "Signifying in the zone of proximal development", An Introduction to Vygotsky **2**, 253–284 (2005).

Lee, J. I. and E. Brunskill, "The impact on individualizing student models on necessary practice opportunities.", International Educational Data Mining Society (2012).

Lesh, A. G. N., "Applying collaborative discourse theory to human-computer interaction", (2004).

Levesque, H., E. Davis and L. Morgenstern, "The winograd schema challenge", in "Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning", (2012).

Liang, C., K. Chee, Y. Zou, H. Zhu, A. Causo, S. Vidas, T. Teng, I. Chen, K. Low and C. Cheah, "Automated robot picking system for e-commerce fulfillment warehouse application", in "The 14th IFToMM World Congress", (2015).

Lord, F., "A theory of test scores.", Psychometric monographs (1952).

Maes, P., "Intelligent software", Scientific American **273**, 3, 84–86 (1995).

Maes, P., B. Shneiderman and J. Miller, "Intelligent software agents vs. user-controlled direct manipulation: a debate", in "CHI'97 Extended Abstracts on Human Factors in Computing Systems, Atlanta, GA, US", pp. 105–106 (ACM, 1997).

Magnisalis, I., S. Demetriadis and A. Karakostas, "Adaptive and intelligent systems for collaborative learning support: A review of the field", IEEE transactions on Learning Technologies **4**, 1, 5–20 (2011).

Mandel, T., "Refraction: Teaching Fractions through Gameplay", https://goo.gl/BrPpmV (2017).

Mandel, T., Y.-E. Liu, S. Levine, E. Brunskill and Z. Popovic, "Offline policy evaluation across representations with applications to educational games", in "AAMAS", (2014).

Manikonda, L., T. Chakraborti, K. Talamadupula and S. Kambhampati, "Herding the crowd: Using automated planning for better crowdsourced planning", Journal of Human Computation (2017).

Mayer, R. E., "Frequency norms and structural analysis of algebra story problems into families, categories, and templates", Instructional Science **10**, 2, 135–175 (1981).

McDermott, D., M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld and D. Wilkins, "PDDL-the planning domain definition language", Technical Report, Tech. Rep. (1998).

McLoughlin, C. and M. J. Lee, "Social software and participatory learning: Pedagogical choices with technology affordances in the web 2.0 era", in "Proceedings of ICT: Providing choices for learners and learning", pp. 664–675 (2007).

Mercier, H. and D. Sperber, "Why do humans reason? arguments for an argumentative theory", Behavioral and brain sciences **34**, 2, 57–74 (2011).

Mesch, D. J., "The jigsaw technique: A way to establish individual accountability in group work", Journal of Management Education **15**, 3, 355–358 (1991).

Miller, T., "Explanation in Artificial Intelligence: Insights from the Social Sciences", CoRR **abs/1706.07269**, URL http://arxiv.org/abs/1706.07269 (2017).

Mirsky, R., Y. Gal and S. M. Shieber, "Cradle: an online plan recognition algorithm for exploratory domains", ACM Transactions on Intelligent Systems and Technology (TIST) **8**, 3, 1–22 (2017).

Mishra, A. P., S. Sengupta, S. Sreedharan, T. Chakraborti and S. Kambhampati, "Cap: A decision support system for crew schedulingusing automated planning", Naturalistic Decision Making (2019).

Mitrovic, A., "An intelligent SQL tutor on the web", International Journal of Artificial Intelligence in Education (2003).

Moore, R. C., "Making the transition to formal proof", Educational Studies in mathematics **27**, 3, 249–266 (1994).

Morrison, J. G., K. M. Feigh, H. S. Smallman, C. M. Burns and K. E. Moore, "The quest for anticipatory decision support systems", Human Factors and Ergonomics Society Annual Meeting (2013).

Mousavinasab, E., N. Zarifsanaiey, S. R. Niakan Kalhori, M. Rakhshan, L. Keikha and M. Ghazi Saeedi, "Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods", Interactive Learning Environments **29**, 1, 142–163 (2021).

Mu, T., K. Goel and E. Brunskill, "Program2tutor: Combining automatic curriculum generation with multi-armed bandits for intelligent tutoring systems", in "Conference on Neural Information Processing Systems", (2017).

Muldner, K., W. Burleson, B. Van de Sande and K. VanLehn, "An analysis of gaming behaviors in an intelligent tutoring system", in "International Conference on Intelligent Tutoring Systems", pp. 184–193 (Springer, 2010).

Murray, R. and K. VanLehn, "A comparison of decision-theoretic, fixed-policy and random tutorial action selection", in "Intelligent Tutoring Systems", pp. 114–123 (2006).

Murray, R. C., K. VanLehn and J. Mostow, "Looking ahead to select tutorial actions: A decision-theoretic approach", International Journal of Artificial Intelligence in Education **14**, 3, 4, 235–278 (2004).

Narayanan, V., Y. Zhang, N. Mendoza and S. Kambhampati, "Automated planning for peer-to-peer teaming and its evaluation in remote human-robot interaction", in "Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts, Portland, US", pp. 161–162 (2015).

Newell, A., H. A. Simon *et al.*, *Human problem solving*, vol. 104.9 (Prentice-hall Englewood Cliffs, NJ, 1972).

Nguyen, T., S. Sreedharan and S. Kambhampati, "Robust planning with incomplete domain models", Artificial Intelligence **245**, 134 – 161 (2017).

Nguyen, T. A., S. Kambhampati and M. Do, "Synthesizing robust plans under incomplete domain models", in "Advances in Neural Information Processing Systems", pp. 2472–2480 (2013).

Novak, J. D. and A. J. Cañas, "The theory underlying concept maps and how to construct them", Florida Institute for Human and Machine Cognition **1**, 1, 1–31 (2006).

Ortony, A. and D. E. Rumelhart, "The representation of knowledge in memory", Schooling and the acquisition of knowledge pp. 99–135 (1977).

Pan, L., Y. Xie, Y. Feng, T.-S. Chua and M.-Y. Kan, "Semantic graphs for generating deep questions", arXiv preprint arXiv:2004.12704 (2020).

Pappano, L., "The year of the mooc", The New York Times **2**, 12 (2012).

Parasuraman, R., "Designing automation for human use: empirical studies and quantitative models", Ergonomics (2000).

Parasuraman, R. and D. H. Manzey, "Complacency and bias in human use of automation: An attentional integration", Human Factors: The Journal of the Human Factors & Ergonomics Society (2010).

Parasuraman, R. and V. Riley, "Humans and automation: Use, misuse, disuse, abuse", Human Factors: The Journal of the Human Factors and Ergonomics Society (1997).

Parasuraman, R., T. B. Sheridan and C. D. Wickens, "A model for types and levels of human interaction with automation", Trans. Sys. Man Cyber. Part A URL http://dx.doi.org/10.1109/3468.844354 (2000).

Pavlik Jr, P. I., K. Brawner, A. Olney and A. Mitrovic, "A review of student models used in intelligent tutoring systems", Design Recommendations for Intelligent Tutoring Systems: Volume 1-Learner Modeling **1**, 63–92 (2013).

Pavlik Jr, P. I., H. Cen and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing.", Online Submission (2009).

Pek, P.-K. and K.-L. Poh, "A bayesian tutoring system for newtonian mechanics: Can it adapt to different learners?", Journal of Educational Computing Research **31**, 3, 281–307 (2004).

Perkins, D. V. and M. J. Tagler, "Jigsaw classroom", Promoting student engagement **1**, 195–197 (2011).

Piech, C., J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas and J. Sohl-Dickstein, "Deep knowledge tracing", in "Advances in neural information processing systems", pp. 505–513 (2015).

Rader, E., K. Cotter and J. Cho, "Explanations as mechanisms for supporting algorithmic transparency", in "Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, Quebec, CA", p. 103 (ACM, 2018).

Rafferty, A. N., E. Brunskill, T. L. Griffiths and P. Shafto, "Faster teaching by pomdp planning", in "International Conference on Artificial Intelligence in Education", pp. 280–287 (Springer, 2011).

Rafferty, A. N., E. Brunskill, T. L. Griffiths and P. Shafto, "Faster teaching via pomdp planning", Cognitive science **40**, 6, 1290–1332 (2016).

Rahati, A. and F. Kabanza, "Automated planning of tutorial dialogues", in "2010 International Conference on Autonomous and Intelligent Systems, AIS 2010", pp. 1–6 (2010).

Ramírez, M. and H. Geffner, "Plan recognition as planning", in "Proceedings of the 21st international joint conference on Artifical intelligence, Pasadena, California, US", pp. 1778–1783 (2009).

Ramírez, M. and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners", in "AAAI Conference on Artifical Intelligence", pp. 1121–1126 (Association for the Advancement of Artificial Intelligence, 2010).

Rasch, G., *Probabilistic models for some intelligence and attainment tests.* (ERIC, 1993).

Regoczei, S. B. and G. Hirst, "Knowledge and knowledge acquisition in the computational context", in "The psychology of expertise", pp. 12–25 (Springer, 1992).

Rivers, K. and K. R. Koedinger, "Automatic generation of programming feedback: A data-driven approach", in "The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)", vol. 50 (2013).

Rivers, K. and K. R. Koedinger, "Data-driven hint generation in vast solution spaces: a self-improving python programming tutor", International Journal of Artificial Intelligence in Education **27**, 1, 37–64 (2017).

Russell, S. and P. Norvig, *Artificial intelligence: a modern approach* (Prentice Hall, 2003).

Scardamalia, M. and C. Bereiter, "Knowledge building: Theory, pedagogy, and technology", The Cambridge Handbook of the Learning Sciences pp. 97–118 (2006).

Scheckler, R. K., "Virtual labs: a substitute for traditional labs?", International journal of developmental biology **47**, 2-3, 231–236 (2003).

Schulze, K. G., R. N. Shelby, D. J. Treacy, M. Wintersgill, K. VanLehn and A. Gertner, "Andes: An active learning, intelligent tutoring system for newtonian physics", THEMES in Education **1**, 2, 115–136 (2000).

Seely Brown, J. and R. Adler, "Open education, the long tail, and learning 2.0", Educause review **43**, 1, 16–20 (2008).

Sengupta, S., T. Chakraborti and S. Kambhampati, "Ma-radar-a mixed-reality interface for collaborative decision making", in "Proceedings of User Interfaces and Scheduling and Planning (uisp) workshop at 28th International Conference of on automated planning and scheduling (icaps), Delft, Netherlands", pp. 40–45 (2018).

Sengupta, S., T. Chakraborti, S. Sreedharan, S. G. Vadlamudi and S. Kambhampati, "RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning", in "Proceedings of User Interfaces and Scheduling and Planning (uisp) workshop at 27th International Conference of on automated planning and scheduling (icaps), Pittsburgh, USA", pp. 44–52 (2017).

Settles, B., "Active learning literature survey", Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (2009).

Sheridan, T. B. and R. Parasuraman, "Human-automation interaction", Reviews of human factors and ergonomics (2005).

Sheridan, T. B. and W. L. Verplank, "Human and computer control of undersea teleoperators", Tech. rep., Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab (1978).

Shneiderman, B. and P. Maes, "Direct manipulation vs. interface agents", interactions **4**, 6, 42–61 (1997).

Shultz, T. R., "Rules of causal attribution", Monographs of the society for research in child development pp. 1–51 (1982).

Shute, V. J., "A macroadaptive approach to tutoring", Journal of Interactive Learning Research **4**, 1, 61–93 (1993).

Smith, D. E., "Planning as an iterative process", in "Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, Ontario, Canada", pp. 2180–2185 (AAAI Press, 2012).

Sreedharan, S., T. Chakraborti and S. Kambhampati, "Handling model uncertainty and multiplicity in explanations via model reconciliation", in "ICAPS", (2018).

Sreedharan, S. and S. Kambhampati, "Handling model uncertainty and multiplicity in explanations via model reconciliation", in "Twenty-Eighth International Conference on Automated Planning and Scheduling", (2018).

Stamper, J., M. Eagle, T. Barnes and M. Croy, "Experimental evaluation of automatic hint generation for a logic tutor", International Journal of Artificial Intelligence in Education **22**, 1-2, 3–17 (2013).

Steinberg, E. R., *Computer-assisted instruction: A synthesis of theory, practice, and technology* (Routledge, 1991).

Stern, R. and B. Juba, "Efficient, safe, and probably approximately complete learning of action models", arXiv preprint arXiv:1705.08961 (2017).

Strimel, G. and S. Grover, "Compression of machine learned models", US Patent 10,558,738 (2020).

Svetlik, M., M. Leonetti, J. Sinapov, R. Shah, N. Walker and P. Stone, "Automatic curriculum graph generation for reinforcement learning agents", in "Thirty-First AAAI Conference on Artificial Intelligence", (2017).

Talamadupula, K., J. Benton, S. Kambhampati, P. Schermerhorn and M. Scheutz, "Planning for human-robot teaming in open worlds", ACM Transactions on Intelligent Systems and Technology (TIST) **1**, 2, 1–24 (2010a).

Talamadupula, K., J. Benton, P. Schermerhorn, S. Kambhampati and M. Scheutz, "Integrating a closed world planner with an open world robot: A case study", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 24, pp. 1561–1566 (2010b).

Tian, X., H. H. Zhuo and S. Kambhampati, "Discovering underlying plans based on distributed representations of actions", in "AAMAS", (2016).

Vadlamudi, S. G., T. Chakraborti, Y. Zhang and S. Kambhampati, "Proactive decision support using automated planning", CoRR **abs/1606.07841**, URL http://arxiv.org/abs/1606.07841 (2016).

Van Beek, P., *Principles and Practice of Constraint Programming-CP 2005* (Springer, 2005).

van De Sande, B., "Properties of the bayesian knowledge tracing model.", Journal of Educational Data Mining **5**, 2, 1–10 (2013).

Van Someren, M., Y. F. Barnard and J. Sandberg, "The think aloud method: a practical approach to modelling cognitive", London: Academic Press **11** (1994).

Vanlehn, K., "The behavior of tutoring systems", International Journal of Artificial Intelligence in Education (2006).

VanLehn, K., "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems", Educational Psychologist **46**, 4, 197–221 (2011).

VanLehn, K., "Model construction as a learning activity: A design space and review", Interactive Learning Environments **21**, 4, 371–413 (2013).

VanLehn, K., C. Banerjee, F. Milner and J. Wetzel, "Teaching algebraic model construction: a tutoring system, lessons learned and an evaluation", International Journal of Artificial Intelligence in Education **30**, 3, 459–480 (2020).

VanLehn, K., S. Cheema, S. Kang and J. Wetzel, "Auto-sending messages in an intelligent orchestration system: a pilot study", in "International Conference on Artificial Intelligence in Education", pp. 292–297 (Springer, 2019).

VanLehn, K., G. Chung, S. Grover, A. Madni and J. Wetzel, "Learning science by constructing models: can dragoon increase learning without increasing the time required?", International Journal of Artificial Intelligence in Education **26**, 4, 1033–1068 (2016).

VanLehn, K., P. W. Jordan, C. P. Rosé, D. Bhembe, M. Böttner, A. Gaydos, M. Makatchev, U. Pappuswamy, M. Ringenberg, A. Roque *et al.*, "The architecture of why2-atlas: A coach for qualitative physics essay writing", in "International conference on intelligent tutoring systems", pp. 158–167 (Springer, 2002).

VanLehn, K., C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein and M. Wintersgill, "The andes physics tutoring system: Lessons learned", International Journal of Artificial Intelligence in Education **15**, 3, 147–204 (2005).

VanLehn, K., B. Van De Sande, R. Shelby and S. Gershman, "The andes physics tutoring system: An experiment in freedom", in "Advances in intelligent tutoring systems", pp. 421–443 (Springer, 2010).

VanLehn, K., J. Wetzel, S. Grover and B. van de Sande, "Learning how to construct models of dynamic systems: An initial evaluation of the dragoon intelligent tutoring system", IEEE Transactions on Learning Technologies (2017).

Verma, P., S. R. Marpally and S. Srivastava, "Asking the right questions: Interpretable action model learning using query-answering", (2020).

Vinner, S. and R. Hershkowitz, "Concept images and common cognitive paths in the development of some simple geometrical concepts", in "Proceedings of the fourth international conference for the psychology of mathematics education", vol. 1, pp. 177–184 (1980).

Walker, E. and W. Burleson, "User-centered design of a teachable robot", in "International Conference on Intelligent Tutoring Systems", pp. 243–249 (Springer, 2012).

Warm, J. S., R. Parasuraman and G. Matthews, "Vigilance requires hard mental work and is stressful", Human Factors: The Journal of the Human Factors and Ergonomics Society (2008).

Webb, N. M., "Information processing approaches to collaborative learning", Routledge Handbooks Online (2013).

Wetzel, J., K. VanLehn, D. Butler, P. Chaudhari, A. Desai, J. Feng, S. Grover, R. Joiner, M. Kong-Sivert, V. Patade *et al.*, "The design and development of the dragoon intelligent tutoring system for model construction: Lessons learned", Interactive Learning Environments **25**, 3, 361–381 (2017).

Wickens, C. D., H. Li, A. Santamaria, A. Sebok and N. B. Sarter, "Stages and levels of automation: An integrated meta-analysis", in "Proceedings of the Human Factors and Ergonomics Society Annual Meeting, San fransisco, California, US", vol. 54.4, pp. 389–393 (Sage Publications Sage CA: Los Angeles, CA, 2010).

Williams, R. J. and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", Neural computation **1**, 2, 270–280 (1989).

Wilson, M., P. De Boeck and C. H. Carstensen, "Explanatory item response models: A brief introduction", Assessment of competencies in educational contexts pp. 91–120 (2008).

Xiong, X., S. Zhao, E. G. Van Inwegen and J. E. Beck, "Going deeper with deep knowledge tracing.", International Educational Data Mining Society (2016).

Xu, Y. and J. Mostow, "Using logistic regression to trace multiple sub-skills in a dynamic bayes net.", in "EDM", pp. 241–246 (Citeseer, 2011).

Xu, Y. and J. Mostow, "Comparison of methods to trace multiple subskills: Is lr-dbn best?.", International Educational Data Mining Society (2012).

Yang, Q., K. Wu and Y. Jiang, "Learning action models from plan examples using weighted max-sat", Artificial Intelligence **171**, 2-3, 107–143 (2007).

Zhang, L., *Biology question generation from a semantic network* (Arizona State University, 2015).

Zhang, L. and K. VanLehn, "How do machine-generated questions compare to human-generated questions?", Research and practice in technology enhanced learning **11**, 1, 1–28 (2016).

Zhang, L. and K. VanLehn, "Adaptively selecting biology questions generated from a semantic network", Interactive Learning Environments **25**, 7, 828–846 (2017).

Zhang, Y., V. Narayanan, T. Chakraborti and S. Kambhampati, "A human factors analysis of proactive support in human-robot teaming", in "2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany", pp. 3586–3593 (IEEE, 2015a).

Zhang, Y., V. Narayanan, T. Chakraborty and S. Kambhampati, "A human factors analysis of proactive assistance in human-robot teaming", in "IROS", (2015b).

Zhang, Y., S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo and S. Kambhampati, "Plan explicability and predictability for robot task planning", in "2017 IEEE international conference on robotics and automation (ICRA), Marina Bay Sands, Singapore", pp. 1313–1320 (IEEE, 2017).

Zhu, L. and R. Givan, "Landmark extraction via planning graph propagation", ICAPS Doctoral Consortium pp. 156–160 (2003).

Zhuo, H. H., Y. Zha, S. Kambhampati and X. Tian, "Discovering underlying plans based on shallow models", ACM Transactions on Intelligent Systems and Technology (TIST) **11**, 2, 1–30 (2020).

APPENDIX A

IPASS USER STUDY DOCUMENTS

This appendix shows all the documents that we provided to the students while conducting the `iPass` user study. These documents were available to the students throughout the user study. There were four different documents we shared –

- Consent form and receipt of payment.
- Small copy of M.S. CS handbook.
- Instruction manual describing the interface behavior.

Consent form and Payment form content below.

```
Study on Interactive Generation of CIDSE Plan of Study

I am Sachin Grover, a PhD student in the School of Computing, Informatics
    and Decision Systems Engineering at Arizona State University. I am
    conducting a research study to find out effective ways in which an
    interactive planning software can provide support for human decision-
    makers. The entire study will be conducted on a prototype interface
    which allows the user to interactively build a plan of study or iPOS.

Subjects will be compensated $10 for their participation. The entire study
    is likely to take under 30 mins to complete. The planning task itself
    is restricted to 15 mins.

We will need to record all the responses provided by the participants
    during the task, and each participant would additionally need to
    complete a questionnaire at the end to be eligible for your payment.
    You have the right not to answer any question and to stop participation
     at any time without penalty. Your participation in this study is
    completely voluntary.

Participating in this study will help us improve the quality of decision-
    making of human decision-makers supported by AI-based systems. The
    participant will also get to interact with and learn about the state-of
    -the-art human-in-the-loop planning techniques. There are no
    foreseeable risks or discomforts associated with your participation.

To protect your privacy, responses from participants will never be used
    individually while compiling or presenting results for the study. The
    results of this study may be used in reports, presentations, or
    publications only in aggregate form. There will be no audio / video
    recording.

If you have any questions concerning the study, please contact me at
    sgrover6@asu.edu.

By signing below you are agreeing to be part of the study -

Name:
```

Signature:

Date:

After completion of study -

I certify that I have been compensated to the amount of $10 in recognition of my participation in the study.

Name:

Signature:

Date:

MS C.S. Handbook , was designed to look like an original CS handbook for Masters degree. Please find the content of the document below.

I. Objective of the handbook

The purpose of this handbook is to provide guidance and information related to degree requirements, and general policies and procedures towards making the interactive Plan of Study or iPOS for a Master of Science in Computer Science with a thesis option.

Please read this document carefully. Note that in some cases you will find differences between the actual Graduate College policies and procedures in practice right now.

II. Deficiencies

Depending on prior academic preparation and accomplishments of an applicant , deficiency courses may be specified to ensure adequate background preparation.

Below is a list of prerequisites, along with the associated ASU course numbers:

- CSE 230 - Computer Organization and Assembly Language Programming
- CSE 310 - Data Structures and Algorithms
- CSE 330 - Operating Systems
- CSE 340 - Principles of Programming Languages
- CSE 355 - Introduction to Theoretical Computer Science
- CSE 360 - Introduction to Software Engineering

III. MS Degree Requirements

Degree requirements for the MS include 10 graduate level courses beyond deficiency courses. The MS is comprised of five major milestones, which all students are required to complete successfully prior to graduation :

a) Completion of all deficiency courses.
b) Completion of all coursework (including one for each area courses, i.e. foundaiton, applications and systems)
c) Selection of 1 committee chair and 2 committee members to oversee Thesis progress.
d) Specialization in one of three topics, i.e. AI, Big Data or Cybersecurity.
e) Successful oral defense of an approved written thesis.

a. Coursework

The iPOS must contain 10 graduate level courses amounting to a total of 30 semester hours. Of these, at least one course has to come from all of the three concentrations - Foundations, Systems and Applications (consult Appendix).
Culminating Experience: CSE 599a and 599b Thesis
Note: Thesis credits (which are included in the count of 10 graduate courses) can only be taken after appointment of a committee chair. CSE 599b has to be taken in the end.
In addition, you must complete a specialization in one of the following topics.

Specialization in Cyber Security
Requires 9 credit hours from
· CSE 543: Information Assurance and Security (3)
· CSE 545: Software Security (3)
· CSE 548: Advanced Computer Network Security (3)

Specialization in Big Data
Requires 9 credit hours from
· CSE 510 Database Management System Implementation (3)
· CSE 512 Distributed Database Systems (3)
· CSE 572 Data Mining (3)

Specialization in Artificial Intelligence
Requires 9 credit hours from
· CSE 575: Statistical Machine Learning (3)
· CSE 574: Planning and Learning Methods in AI (3)
· CSE 571: Artificial Intelligence (3)

b. Selection of Faculty Advisor - Thesis Option

You must select a faculty advisor from the list of professors. The faculty advisor must be from the same area of specialization. You cannot complete your Specialization and Thesis Credits until you have selected your advisor.

c. Thesis Supervisory Committee

```
In addition to the chair, you must appoint two more members of the Thesis
    Advisory committee which advises the student during the formulation of
    the research topic and during the completion of the research and thesis
    . The non-chair members need not be in your specialization area.

d. Thesis

The iPOS must end in defense instead of a normal semester end.

e. Continuous Enrollment

Students must enroll in at least one course every semester.

International students must register for at least three courses. Only
    Research Assistants (RA) or Teaching Assistants (TA) can register for
    four courses in a semester.

It is not possible to register for more than four courses in a single
    semester.

Every semester has a $1000 registration fee.
Each course enrollment requires a further $3000 fee.
```

This was followed by a table of courses and their specialization and the list of professors with their specialization.

```
List of Approved 500 (Graduate Level) Area Courses
```

Course	Course Title	Foundations	Systems	Applications
CSE 509	Digital Video Processing			X
CSE 510	Database Management System Implementation			X
CSE 511	Semi-Structured Data Management			X
CSE 512	Distributed Database Systems			X
CSE 515	Multimedia and Web Databases			X
CSE 520	Computer Architecture		X	
CSE 522	Real-Time Embedded Systems		X	
CSE 530	Embedded Operating Systems Internals		X	

CSE 531	Distributed and Multiprocessor Operating Systems		X	
CSE 534	Advanced Computer Networks		X	
CSE 535	Mobile Computing		X	
CSE 536	Advanced Operating Systems		X	
CSE 539	Applied Cryptography		X	
CSE 543	Information Assurance and Security		X	
CSE 545	Software Security		X	
CSE 546	Cloud Computing		X	
CSE 548	Advanced Computer Network Security		X	
CSE 550	Combinatorial Algorithms and Intractability	X		
CSE 551	Foundations of Algorithms	X		
CSE 552	Randomized and Approximation Algorithms	X		
CSE 555	Theory of Computation	X		
CSE 556	Game Theory with Applications to Networks	X		
CSE 561	Modeling and Simulation Theory and Applications		X	
CSE 563	Software Requirements and Specification		X	
CSE 564	Software Design		X	
CSE 565	Software Verification, Validation and Testing		X	
CSE 566	Software Project, Process and Quality Management		X	
CSE 569	Fundamentals of Statistical Learning and Pattern Recognition	X		

CSE 570	Advanced Computer Graphics			X
CSE 571	Artificial Intelligence			X
CSE 572	Data Mining			X
CSE 573	Semantic Web Mining			X
CSE 574	Planning and Learning Methods in AI			X
CSE 575	Statistical Machine Learning			X
CSE 576	Topics in Natural Language Processing			X
CSE 577	Advanced Geometric Modeling			X
CSE 578	Data Visualization			X
CSE 579	Knowledge Representation and Reasoning	X		

Table A.1: List of courses that were available for the students. Students could take all the courses except special courses such as CSE 591 and CSE 598 courses for their Masters degree.

We also presented another table to the students which described the specialization of different professor in the department.

Instruction Manual. Finally, we showed the student a printed instruction manual describing the ability of the interface. Picture A.1 shows the classification of the interface into three panels. After this we described every panel one by one. The actions described different abilities of the interface such as adding courses, choosing specialization etc. We also described the application of each button, such as to validating the plan, and plan suggestions.

Professor	Foundation	AI	Big Data	Cybersecurity
Charlie Coulborn	X			
Arunabha Sen	X			
Guoliang Xue			X	
Andrea Richa	X			
Adam Doupe				X
Dijiang Huang				X
Mohammad Sarwat			X	
Selchuk Candan				X
Subbarao Kambhampati		X		
Huan Liu			X	
Yu Zhang		X		
Heni Ben Amor		X		

Table A.2: List of professors and their specialization. The specialization should match the specialization a student is working towards in their thesis.

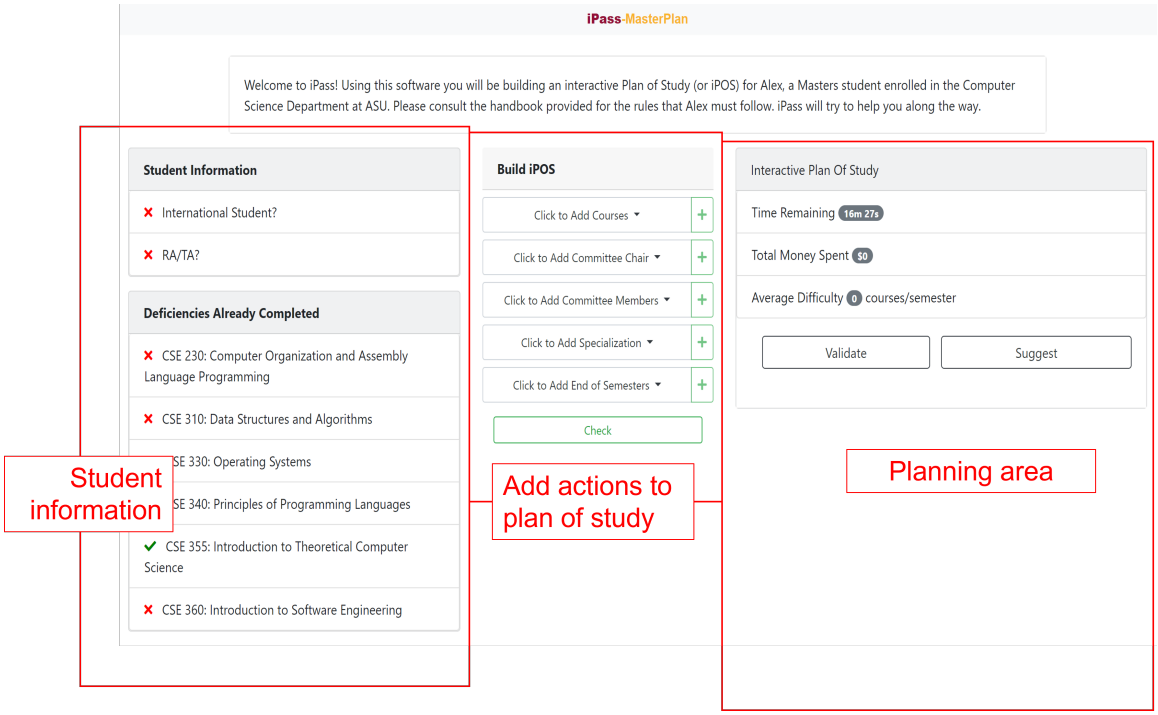


Figure A.1: Initial image of the instruction manual describing the three panel structure. Every condition had different image for the interface, and description of behavior.

APPENDIX B

MAY I ASK A QUESTION USER STUDY DOCUMENTS

This appendix shows all the documents that we provided to the students while conducting the `May I Ask a Question`user study. These documents were available to the students throughout the user study. There were four different documents we shared –

- Consent form.
- Study and warehouse storage description.
- Robot capabilities description.
- Rephrased pre-test rules and distractor rules given to students.
- Video describing the interface of the user study.

Consent form describes the initial study, it's possible consequences, data we record and precautions we will take with the private information.

```
I am Sachin Grover, a Ph.D. student in the School of Computing, Informatics
    and Decision Systems Engineering at Arizona State University. I am
   conducting a research study to find out effective ways to ask questions
    for teaming between human and a robot. The study will be conducted
   online using this interface.

You will be compensated $15 for your participation. The study will take
   around 45 minutes to an hour to complete. The study consists of five
   different screens after this consent form. First, you will be explained
    a human-robot teaming scenario, followed by the capabilities and
   security rules to be followed by the robot. After this we will test
   your understanding of rules through two different sets of question
   answers. This will be followed by a feedback form.

We will need to record your responses to the questions as well the feedback
    form presented after the study. All the results will be presented in
   aggregated form and no personal details (or any identifying information
   ) about you will be used anywhere. You have the right not to answer any
    question and to stop participation at any time without penalty. Your
   participation in this study is completely voluntary.

Through this study we want to see if a human-robot team can interact and
   work together. It will help us improve the way robot-human teams can
   work together by communicating with each other. The detailed scenario
   for the team will be explained on the next page. There are no
   foreseeable risks or discomforts associated with your participation.

To protect your privacy, your responses will never be used individually
   while compiling or presenting the results for the study. The final
   results may be used for reports, presentations and publications only in
    aggregate form. We only record your name for the consent form and your
    responses the questions and the feedback form. No audio or video is
   recorded (neither on Zoom or anywhere else).
```

```
If you have any questions concerning the study, please contact me at
    sgrover6@asu.edu. By writing your name and clicking the button you
    accept the conditions for the study. The compensation for your work
    will be done after completing the study.
```

Study description describes the three main actors in the user study – Sam, Squeaky and the Planner. Although, in the chapter we don't talk about the planner, however, we introduce it to create a story around the requirement of asking questions from Sam.

```
There is a Robot Squeaky that works in a warehouse storing vaccines. The
    vaccines are stored in sub-zero (freezing) temperatures, which are
    unsuitable for human workers. A robot expert Sam, monitors the area. He
     is responsible for managing the vaccine orders and ensuring security
    protocols for the warehouse. For the orders, Sam provides Squeaky
    commands that Squeaky follows. Since Sam is the expert, he understands
    how the robot functions and the steps to maintain security in the
    warehouse.

Warehouse environment and security protocols affect the behavior of Squeaky
    . For example, Squeaky can move freely around the warehouse. However,
    for security reasons, it should keep it's arm close to it's body while
    moving (called tucking the arm). Sam has to change the commands sent to
     the robot based on many such small rules.

The company is trying to increase productivity and decrease Sam's workload.
     Thus, Squeaky has been equipped with a planner that can issue
    sequential commands instead of Sam. The planner doesn't know the
    security rules that Sam knows, but it can learn them by asking
    questions from Sam.

As a participant in this study, you play Sam. You will first learn about
    the capabilities of Squeaky and become an expert, and then you will
    learn the set of security rules. As you will be the expert in handling
    Squeaky, you will need to know these rules well because you will need
    to answer the questions asked by the planner as it acquires your
    expertise. On the next page, we will describe the capabilities of
    Squeaky, the warehouse environment and the security rules. There will
    be a small test that you have to ace to prove your expertise before you
     start answering planner's questions. In the end, we will provide a
    feedback form, where you will describe your understanding of the study.
```

Robot capabilities and warehouse description, introduces the robot, describes the warehouse environment and the security rules that Sam follows. All the rules have been described in Chapter 6. As described earlier, students are shown only 4 out of 6 rules. We use a combination of text and images to provide the information to the student. The videos can be checked at the Youtube link provided with each video. Figure B shows the Figure used to show robot Squeaky.

Figure B.1: Image of the robot Fetch used to introduce Squeaky to the student.

```
<Figure> Hi I am Squeaky. I am capable of moving in the warehouse, picking
    up and moving stuff around the warehouse. Please check the videos below
     to see my capabilities.

<Video>(https://youtu.be/6IuEaiy75vA) I can tuck and extend my arm to reach
     far away stuff. The video shows the action of tucking the arm.
    Similarly, I can also extend it as you will check below.

<Video>(https://youtube.com/shorts/1oOLQOBeFWU) I also have an expandable
    torso which increases my height. It helps me reaching different objects
     and gives me space to move my arm. Video shows me expanding and
    contracting my torso.

<Video>(https://www.youtube.com/watch?v=OCGFhJaNtpo) I have capability to
    pick up stuff. In the video you can see that I can pickup as well as
    putdown the box.
```

<Video>(https://youtu.be/k15lKskCMWU) I can move using the wheels in my base. In the video I have been asked to move to a table.

I live at a Warehouse, which consists of serveral objects. To understand the safety rules, I will describe the objects in the warehouse and then describe the rules --

- Warehouse consists of -- racks and boxes.
- Racks where medicines are kept.
- For simplicity, assume that each rack has only one level.
- Boxes of medicine are of two types -- (1) large -- containing 10 bottles, (2) small -- containing 2 bottles
- I am responsible for making stacks of boxes of medicines and move the boxes around for storage purposes.

Security rules that I have to follow --

- While moving my torso should be in the contracted position, otherwise, I can topple on the small wheels while turning.
- While moving my arm should be tucked in, as extended arm is heavy and I can topple.
- While picking up I can pick up both kind of boxes. Although, I can only pick one box at a time.
- A large box can be stacked only on a large box, whereas a small box can be stacked on both large and small boxes.
- I can pick up a large box only when my torso is in expanded state, as I can hold the box vertically.
- I can tuck my arm while holding a small box and not while holding a large box. Thus, I can move while holding small box.

Pre-test consists of helping students memorize the security rules. The rules are rephrased and distractor rules are added based on the basic rules that the student can have. Below you can check the rephrased rules and the distractor rules for the core rules. For every test, based on the rules shown to the user, four rephrased rules, and four corresponding distractor rules are selected. The list is shown to the user in random order. A student gives the pre-test three times.

Rephrased security rules --
- Squeaky's torso should be in contracted position while moving.
- While moving squeaky should not extend it's arm.
- Squeaky can only pick up one box at a time.
- A large box can be stacked only on large boxes.
- Squeaky can pick up large boxes only when it's torso is expanded.
- Squeaky can not tuck it's arm while holding the large box.

Distractor rules for rule 1 & 2 --
- Squeaky can move with it's torso expanded.
- Squeaky can move without tucking it's arm.
- Squeaky can freely move in the warehouse.

Distractor rules for rule 3 & 4 --
· Squeaky can pick up many boxes at a time.
· Squeaky can pick up many large boxes at a time.
· A large box can be stacked at any place.
· A small box can not be stacked on top of a large box.
· A large box can be stacked on a small box.

Distractor rules for rule 5 & 6 --
· Squeaky can easily pick up large boxes.
· Squeaky can move while holding any kind of box.
· Squeaky can tuck it's arm while holding a large box.
· Squeaky can move while holding a large box

Feedback for distractor rules for rule 1 & 2 --
· Squeaky has small wheels and it should move with torso compressed.
· Squeaky has a heavy arm and it's arm should be tucked in while moving.
· Squeaky has small wheels and heavy arm, thus it needs to follow safety rules while moving in the environment.

Feedback for distractor rules for rule 3 & 4 --
· Squeaky can pick only one box at a time.
· Squeaky can pick only one large box at a time.
· A large box can only be stacked on a large box.
· A small box can be stacked on top of both large and a small box.
· A large can not be stacked on a small box.

Feedback for distractor rules for rule 5 & 6 --
· Due to extra weight Squeaky can pick up a large box after expanding it's torso.
· Squeaky can move while holding a small box, with it's arm tucked in.
· Squeaky can tuck it's arm while holding only a small box.
· Squeaky can move while holding only a small box.

User Interface details. Students are shown a video describing different parts of the interface and how robot can work in a warehouse. The videos are chosen based the query type and the rules that were shown to the students. Thus, we made four different videos for two conditions and 2 different set of rules that are shown to each user. The videos are –

· Validation query with rules 1, 2, 3 and 4 -- https://www.youtube.com/watch?v=NscMijSBlIk
· Validation query with rules 1, 2, 5 and 6 -- https://www.youtube.com/watch?v=CXn9CwhVSH8
· Construction query with rules 1, 2, 3 and 4 -- https://www.youtube.com/watch?v=2Mh92XQidsI
· Construction query with rules 1, 2, 5 and 6 -- https://www.youtube.com/watch?v=EhgYznhRT-c

Feedback Questions. After the study students are asked a set of objective and subjective questions. They are as follows –

- `Describe the rules that Squeaky has to follow in your own words.`
- `Describe in detail atleast 3 things you liked about the interaction initiated by the planner.`
- `Describe in detail atleast 3 things you did NOT like about the interaction initiated by the planner.`
- `Describe in detail what other features you would like to improve for the interaction with human teammates.`
- `On a scale of 1 -- 5 rate -- The questions above were pretty easy to solve for you.`
- `On a scale of 1 -- 5 rate -- The questions above were difficult to understand.`
- `On a scale of 1 -- 5 rate -- Intent of asking the questions is not clear for the planner.`

First question in the feedback acts as a post-test question. We also ask about some personal information, i.e. whether they have ever worked with a robot or not, and what is their Major and department of study.